Placing problems from phylogenetics and (quantified) propositional logic in the polynomial hierarchy

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Janosch Otto Döcker
aus Tübingen

Tübingen
2021

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	23.06.2021
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter:	Apl. Prof. Dr. Britta Dorn
2. Berichterstatter:	Prof. Dr. Peter Hauck
3. Berichterstatter:	Prof. Dr. Uwe Schöning

Contents

Acknowledgments

I feel great gratitude to everyone who supported me on my PhD journey. Huge thanks go to the supervisors of my dissertation: Britta Dorn and Peter Hauck. Britta Dorn inspired me to pursue a PhD after completing my master's degree and was tremendously supportive during the whole process. She also got me into contact with Simone Linz (University of Auckland) which resulted in two research stays in New Zealand and inspiring collaborations with several researchers from the phylogenetics community. Moreover, I would like to thank Britta Dorn for not only allowing me to fully pursue my research interests but also being incredibly supportive of it—Thank you Britta! I am very grateful to Peter Hauck for assessing my dissertation even though the submission of my thesis eventually fell within his retirement and for being really helpful in the process of getting my thesis submitted—Thank you!

Equally, I would like to thank Simone Linz and Charles Semple for their incredible hospitality during my two research stays in New Zealand.

In my time as a student I was fortunate to meet several great people who helped me grow scientifically and personally. Special thanks go to Britta Dorn, Simone Linz and Elke Wilkeit who I consider to be my mentors and my main academic influences.

Big thanks go to the co-authors of my papers: Britta Dorn, Simone Linz, Andreas Darmann, Charles Semple, Steven Kelk, Leo van Iersel, Ulle Endriss, Jérôme Lang, Ronald de Haan, Sebastian Schneckenburger and Dominikus Krüger. I learnt a lot from you guys and I enjoyed every collaboration!

Another huge "Thank you!" goes to Britta Dorn, Simone Linz and my partner Leonie for taking time out of their busy schedules and providing valuable feedback and helpful discussions on draft versions of my dissertation which benefited my thesis significantly.

Further, I'm grateful for the support of the secretary Renate Hallmayer who was always kind, patient and helpful when I had questions related to my studies or my employment at the university. Thank you Renate!

Last but not least, I'm infinitely grateful for the love and support of my family, friends and my beloved partner Leonie.

Summary

German

In der vorliegenden Arbeit betrachten wir die Komplexität von Entscheidungsproblemen aus zwei Themenbereichen und ordnen diese in der Polynomialzeithierarchie ein.

Zum einen befassen wir uns mit Problemen, die im Zusammenhang mit phylogenetischen Netzwerken auftreten. Ein phylogenetisches Netzwerk ist im Forschungsbereich der Phylogenetik ein etabliertes Modell, um die evolutionäre Beziehung zwischen verschiedenen Arten darzustellen. Es wurde als Verallgemeinerung von phylogenetischen Bäumen eingeführt, da letztere keine von einer Baumstruktur abweichenden biologischen Prozesse wie etwa Hybridisierung oder horizontalen Gentransfer abbilden können. Richtet man allerdings den Fokus auf einzelne Bereiche des *Genoms* der Arten, so lässt sich die evolutionäre Geschichte häufig durch einen einzigen Baum modellieren. Ein phylogenetisches Netzwerk kann daher auch als Zusammenfassung mehrerer solcher Bäume aufgefasst werden, es ist dann somit ein Modell, das gleichzeitig alle gegebenen baumartigen Entwicklungen von Teilen des Genoms erklärt (d.h. die zugehörigen Bäume einbettet). In diesem Zusammenhang betrachten wir die Frage, ob ein derartiges Netzwerk für eine gegebene Menge von phylogenetischen Bäumen existiert, wenn der Suchraum auf eine bestimmte Teilklasse von phylogenetischen Netzwerken (den sogenannten *temporalen* Netzwerken [MNW$^+$04]) beschränkt ist. Aufbauend auf einer von Humphries et al. [HLS13a] entwickelte Charakterisierung dieser Teilklasse zeigen wir, dass die Existenz eines solchen Netzwerks für jede nicht-triviale Anzahl (d.h. mindestens zwei) von gegebenen Bäumen schwierig zu entscheiden ist: das Entscheidungsproblem stellt sich als NP-vollständig heraus. Andererseits entwickeln wir unter Verwendung von Automatentheorie einen Algorithmus, der für eine beliebige, aber konstante Anzahl an Bäumen, die zusätzlich eine bestimmte topologische Eigenschaft erfüllen, die Existenzfrage in Polynomzeit löst. Anschließend untersuchen wir die Beziehungen von zwei gegebenen phylogenetischen Netzwerken im Hinblick auf die eingebetteten phylogenetischen Bäume. Zum Beispiel analysieren wir die folgenden Fragen hinsichtlich ihrer Komplexität:

Sind in beiden Netzwerken exakt die gleichen phylogenetischen Bäume eingebettet? Sind alle phylogenetischen Bäume, die im ersten Netzwerk eingebettet sind, auch im zweiten Netzwerk eingebettet? Für die beiden genannten Fragestellungen zeigen wir, dass diese Π_2^P-vollständig sind, also auf der zweiten Ebene der Polynomialzeithierarchie liegen. Im Kontrast zu diesen negativen Komplexitätsresultaten geben wir einen Algorithmus an, der das erste Entscheidungsproblem — d.h. den Test auf Gleichheit — in Polynomzeit löst, wenn die beiden Netzwerke jeweils zu einer bestimmten Klasse gehören.

Zum anderen beschäftigen wir uns mit Varianten des Erfüllbarkeitsproblems für (quantifizierte) Boolesche Formeln. Unsere Forschung zu diesem Thema ist inspiriert durch eine Vermutung für Monotone 3-SAT mit beschränkten Variablenvorkommen, die in der Mitschrift [DKY14] einer Vorlesung des Massachusetts Institute of Technology aufgestellt wurde: Die Vermutung war, dass Monotone 3-SAT NP-schwer ist, wenn jede Variable höchstens fünf mal vorkommt. Wir bestätigen die Vermutung und zeigen, dass das Problem sogar NP-schwer ist, wenn jede Variable genau zweimal nicht-negiert und zweimal negiert, insgesamt also exakt vier mal in der Formel vorkommt. Darüber hinaus erhalten wir ein allgemeineres Resultat für solche Formeln, wenn jede Variable exakt p mal nicht-negiert und q mal negiert vorkommt, wobei p und q beliebige, aber konstante natürliche Zahlen sind. Das Problem ist NP-vollständig, wenn $p \geq 2$ und $q \geq 2$ oder $p + q \geq 6$ gilt (oder beides). Abgesehen von den symmetrischen Fällen, sind die beiden einzigen verbleibenden offenen Fälle von der Form $p \in \{3, 4\}$ und $q = 1$. Wir befassen uns außerdem mit monotonen Formeln, deren Inzidenzgraph eine planare Darstellung besitzt. Der Fokus liegt hierbei wiederum auf beschränkten Variablenvorkommen. Zudem zeigen wir, dass eine besonders eingeschränkte Variante von Not-All-Equal 3-SAT (NAE-3-SAT) NP-vollständig ist. Schließlich betrachten wir die quantifizierten Varianten $\forall\exists$ 3-SAT und $\forall\exists$ NAE-3-SAT für Instanzen mit beschränkten Variablenvorkommen und ordnen diese in der Polynomialzeithierarchie ein. Am Schluss dieser Arbeit zeigen wir exemplarisch, wie diese eingeschränkten Varianten des Erfüllbarkeitsproblems Anwendung finden können, um Reduktionen zu vereinfachen oder stärkere Resultate für andere Entscheidungsprobleme abzuleiten.

English

In this thesis, we consider the complexity of decision problems from two different areas of research and place them in the polynomial hierarchy.

On the one hand, we are concerned with problems that arise in the context of phylogenetic networks. A phylogenetic network is a well-established model used in evolutionary biology to represent the evolutionary relationships between different species. In graph-theoretic terms, a phylogenetic network is a leaf-labeled rooted DAG[1]. It was introduced as a generalization of phylogenetic trees as the latter is not suitable to represent biological processes that cannot be represented in a tree-like fashion such as hybridization or lateral gene transfer. However, if the focus is laid on specific parts of the *genome* of the considered species, the evolutionary history can often be modeled by a single tree. Hence, a phylogenetic network can also be viewed as an amalgamation of several such trees in which case the network embeds the corresponding trees. In that regard, we consider the question whether such a network exists for a given set of phylogenetic trees if the search space is restricted to a certain subclass of phylogenetic networks, the so-called *temporal* networks [MNW+04]. Using a characterization of this subclass introduced by Humphries et al. [HLS13a], we show that deciding the existence of such a network is NP-complete for each non-trivial number (i.e., at least two) of given trees. On the positive side, by using automata theory, we present an algorithm solving the problem in polynomial time if the number of trees in the input is bounded by some constant and each tree satisfies a certain topological constraint. Subsequently, we investigate the relationship of two given phylogenetic networks with respect to the phylogenetic trees embedded by the networks. For instance, we consider the following questions regarding their computational complexity: Do both networks embed precisely the same phylogenetic trees? Is each phylogenetic tree that is embedded in the first network also embedded in the second network? We show that both decision problems are Π_2^P-complete and, thus, can be placed on the second level of the polynomial hierarchy. In contrast to these negative complexity results, we present an algorithm solving the former decision problem—i.e., checking for equality—in polynomial time if each network belongs to a particular class (i.e., the networks have to satisfy certain structural properties).

On the other hand, we consider variants of the satisfiability problem for (quantified) Boolean formulas. Our research on this topic is inspired by a conjecture for MONOTONE 3-SAT with bounded variable appearances stated in the scribe notes [DKY14] of a lecture

[1] directed acyclic graph

held at the Massachusetts Institute of Technology: The conjecture was that MONOTONE 3-SAT is NP-hard if each variable appears at most five times. We confirm the conjecture and show that the problem remains NP-hard even if each variable appears exactly twice unnegated and exactly twice negated (thus, each variable has a total number of four appearances in the formula). Further, we obtain a more general result for such formulas, if each variable appears exactly p time unnegated and q times negated, where p and q are arbitrary but constant positive integers. The problem is NP-complete if $p \geq 2$ and $q \geq 2$ or $p + q \geq 6$ holds (or both). Apart from the symmetric cases, the only two remaining open cases have the form $p \in \{3, 4\}$ and $q = 1$. We also consider monotone formulas, where the incidence graph is planar. The focus is again laid on bounded variable appearances. In addition, we show that a particularly restricted variant of NOT-ALL-EQUAL 3-SAT (NAE-3-SAT) is NP-complete. Finally, we consider the quantified variants $\forall\exists$ 3-SAT and $\forall\exists$ NAE-3-SAT for instances with bounded variable appearances and place them in the polynomial hierarchy. We close this thesis by showing in an exemplary way how these restricted variants of the satisfiability problem can be applied to simplify reductions or to infer stronger results for other decision problems.

List of Publications

Accepted Manuscripts

1. Janosch Döcker and Simone Linz. **On the existence of a cherry-picking sequence** [DL18]. Theoretical Computer Science, 714:36–50, 2018. The published version of this article can be found in Appendix 1.1 and the paper is also available online at the following URL: `https://doi.org/10.1016/j.tcs.2017.12.005`.

2. Janosch Döcker, Leo van Iersel, Steven Kelk, and Simone Linz. **Deciding the existence of a cherry-picking sequence is hard on two trees** [DvIKL19]. Discrete Applied Mathematics, 260:131–143, 2019. The published version of this article can be found in Appendix 1.2 and the paper is also available online at the following URL: `https://doi.org/10.1016/j.dam.2019.01.031`.

3. Janosch Döcker, Simone Linz, and Charles Semple. **Displaying trees across two phylogenetic networks** [DLS19]. Theoretical Computer Science, 796:129–146, 2019. The published version of this article can be found in Appendix 1.3 and the paper is also available online at the following URL: `https://doi.org/10.1016/j.tcs.2019.09.003`.

4. Janosch Döcker, Simone Linz, and Charles Semple. **The display sets of normal and tree-child networks** [DLS21]. The Electronic Journal of Combinatorics, 28(1):P1.8 (21 pages), 2021. The published version of this article can be found in Appendix 1.4 and the paper is also available online at the following URL: `https://doi.org/10.37236/9128`.

5. Andreas Darmann, Janosch Döcker, and Britta Dorn. **The Monotone Satisfiability Problem with Bounded Variable Appearances** [DDD18a]. International Journal of Foundations of Computer Science, 29(6):979–993, 2018. The author created version of this article that was accepted for publication can be

found in Appendix 1.5. The published version of this article is available online at the following URL: https://doi.org/10.1142/S0129054118500168.

6. Andreas Darmann and Janosch Döcker. **On simplified NP-complete variants of Monotone 3-Sat** [DD21]. Discrete Applied Mathematics, 292:45–58, 2021. The published version of this article can be found in Appendix 1.6 and the paper is also available online at the following URL: https://doi.org/10.1016/j.dam.2020.12.010.

7. Andreas Darmann and Janosch Döcker. **On a simple hard variant of Not-All-Equal 3-Sat** [DD20]. Theoretical Computer Science, 815:147–152, 2020. The published version of this article can be found in Appendix 1.7 and the paper is also available online at the following URL: https://doi.org/10.1016/j.tcs.2020.02.010.

8. Janosch Döcker, Britta Dorn, Simone Linz, and Charles Semple. **Placing quantified variants of 3-SAT and Not-All-Equal 3-SAT in the polynomial hierarchy** [DDLS20]. Theoretical Computer Science, 822:72–91, 2020. The published version of this article can be found in Appendix 1.8 and the paper is also available online at the following URL: https://doi.org/10.1016/j.tcs.2020.04.003.

Additional Manuscripts

We added the main result of the following article to the 6th accepted manuscript [DD21] during peer review (i.e., it was not part of the initial submission but included in the version accepted for publication). Since not all of the results of the former article—in particular the results for the quantified variants of the satisfiability problem—appear in the accepted manuscript, we decided to include both.

9. Janosch Döcker. **Monotone 3-SAT-(2,2) is NP-complete** [Döc19]. arXiv: 1912.08032 [cs.CC], 2019. This manuscript can be found in Appendix 2.1 and it is also available online at the following URL: https://arxiv.org/abs/1912.08032.

Personal Contribution

Accepted Manuscripts

1. **On the existence of a cherry-picking sequence** [DL18]: In 2017, I was a visiting student at the University of Auckland. The project was initiated by Simone Linz who worked with me on problems from phylogenetics during my stay. She introduced me to the then open question concerning the existence of a cherry-picking sequence given a set of phylogenetic trees. The main proof ideas leading to the results in the above manuscript were contributed by me. However, working out the finer details and writing up the paper was a shared effort by Simone Linz and myself.

2. **Deciding the existence of a cherry-picking sequence is hard on two trees** [DvIKL19]: Steven Kelk initiated this collaboration on the existence of a cherry-picking sequence on two trees and he provided the main proof ideas (i.e., a sketch of the reduction). I significantly contributed to the second direction of the proof of the main result by working out the finer details required to obtain a rigorous proof. Writing up the paper was a shared effort by all four authors, where Simone Linz and—to a slightly lesser extent—myself had a leading role.

3. **Displaying trees across two phylogenetic networks** [DLS19]: This manuscript is the result of a collaboration with Simone Linz and Charles Semple (in 2018, I was, for the second time, a visiting student at the University of Auckland and subsequently at the University of Canterbury, where the three of us worked together in person). The idea to consider the topic of the manuscript was already suggested in 2017 by Simone Linz when we were looking for possible problems to work on during my first visit. I came up with the idea to consider complexity beyond NP-hardness (specifically Π_2^P-completeness) in order to obtain a finer placement of these decision problems in the polynomial hierarchy. Also, I contributed the main ideas for the constructions leading to our Π_2^P-completeness result

for DISPLAY-SET-CONTAINMENT. The proof idea obtaining Π_2^P-completeness for DISPLAY-SET-EQUIVALENCE is shared by Simone Linz and myself (she came up with a reduction showing NP-hardness and I observed that it can be generalized to show Π_2^P-completeness using our result for DISPLAY-SET-CONTAINMENT). Apart from verifying correctness, I was not involved in obtaining the NP-completeness result for COMMON-TREE-CONTAINMENT. Writing up the paper was a shared effort by all three authors, where Simone Linz and myself had a leading role.

4. **The display sets of normal and tree-child networks** [DLS21]: These results were also obtained as a result of the collaboration with Simone Linz and Charles Semple alluded to above for the preceding manuscript. We developed many of the involved ideas together while collaborating in person which makes it hard to single out individual contributions. That being said, most of the crucial observations were made by Simone Linz and Charles Semple. My most significant contributions to the collaboration on the display sets of phylogenetic networks are part of the manuscript containing our hardness results (i.e., in the preceding manuscript above). Writing up the paper was a shared effort by all three authors, where Charles Semple had a leading role.

5. **The Monotone Satisfiability Problem with Bounded Variable Appearances** [DDD18a]: This manuscript is a collaboration with Andreas Darmann and Britta Dorn. The research was initiated by me and most of the scientific ideas and results were contributed by me. A significant contribution by Andreas Darmann is the collection of clauses used as a gadget in the proof showing that MONOTONE 3-SAT-4 is NP-complete. Writing up the paper was a shared effort by all three authors, where I had a leading role.

6. **On simplified NP-complete variants of Monotone 3-Sat** [DD21]: This manuscript is joint work with Andreas Darmann. The research was initiated by me and most of the scientific ideas and results were contributed by me. Andreas Darmann helped me with working out the finer details of the proofs and fixing minor errors that were present in the original draft. An anonymous referee provided the tool for increasing the number of literal appearances which benefited the manuscript significantly (Section 3 of our manuscript is dedicated to this tool). Writing up the paper was a shared effort by Andreas Darmann and myself, where I had the leading role.

7. **On a simple hard variant of Not-All-Equal 3-Sat** [DD20]: This manuscript is joint work with Andreas Darmann. The research was initiated by me and most of the scientific ideas and results were contributed by me. Andreas Darmann helped me with working out the finer details of the proofs, finding subtle inaccuracies, and improving the presentation of the original draft. Writing up the paper was a shared effort by Andreas Darmann and myself, where I had the leading role.

8. **Placing quantified variants of 3-SAT and Not-All-Equal 3-SAT in the polynomial hierarchy** [DDLS20]: This manuscript is joint work with Britta Dorn, Simone Linz and Charles Semple. The research was initiated by me and most of the scientific ideas and results were contributed by me. Simone Linz and Charles Semple contributed crucially in working out finer details of the proofs, thus making the proofs significantly more rigorous, as well as improving the overall presentation of the manuscript. Writing up the paper was a shared effort by all four authors, where I had a leading role.

Additional Manuscripts

9. **Monotone 3-SAT-(2,2) is NP-complete** [Döc19]: This manuscript is in its entirety my own work (with respect to both the scientific ideas and the writing).

1 Introduction

One important task in theoretical computer science is to classify decision problems regarding the computational effort[1] necessary to solve them. If we find an efficient algorithm for some decision problem, the benefit of such an analysis is immediately obvious—we can find an answer to the corresponding question in a reasonable amount of time. Here, we call an algorithm efficient if it returns the result within polynomial time (measured in the length of the input). The complexity class P contains all decision problems that can be solved by an efficient algorithm. However, there are many problems that arise in practical applications that sound simple at first glance—or at least are easy to understand—but withstood all efforts to find an efficient algorithm for them so far. A well-known example of such a decision problem is the Traveling Salesperson problem which asks whether a salesperson can make a round trip visiting a number of cities such that the traveled distance stays within a given limit. The latter problem belongs to the class NP which contains all decision problems for which a solution can be verified efficiently. Moreover, Traveling Salesperson is NP-hard which means that, in a certain sense, it is at least as hard to solve as any other problem in NP (see, e.g., Garey and Johnson [GJ79] for more details). Further, an NP-hard problem that is also in NP is called NP-complete. As indicated above, there are no polynomial-time algorithms available for any decision problem that is NP-hard. Intuitively, an NP-complete decision problem can be viewed as a representative of the class NP in the following sense: if a single NP-complete decision problem turns out to be in P it follows that $P = NP$. The question whether $P = NP$ or $P \neq NP$ is perhaps the most famous open problem in theoretical computer science and it is also one of the Millennium Prize Problems[2] established by the Clay Mathematics Institute which awards $1 million for the first correct solution to each of the problems. Now, what are the implications of showing that a decision problem is NP-complete? An important consequence is that there is no efficient algorithm for an NP-hard problem unless $P = NP$. In other words: An efficient algorithm for

[1] We assume the reader to be familiar with the basic concepts of computational complexity and refer to the book by Garey and Johnson [GJ79] for an introduction to this topic.

[2] See http://www.claymath.org/millennium-problems (last accessed: 19.12.2020)

an NP-complete problem would provide an answer to the P vs. NP question alluded to above. Hence, for an NP-complete problem, it is crucial to shift the attention to other algorithmic techniques. Thus, classifying decision problems regarding their computational complexity provides valuable information even if the problem cannot be placed in P. In fact, there are many complexity classes beyond NP. For instance, the *polynomial hierarchy* [Sto76] is a system of complexity classes that contains P and NP, but also classes with potentially harder problems (the use of the word "potentially" already indicates that there are open problems concerning the relationships of the classes in the polynomial hierarchy). We will come back to the polynomial hierarchy in Section 3.1, where we give the formal definition and provide the main ideas that are relevant to this thesis.

The goal of this thesis is to place decision problems in their respective complexity class. To this end, we use the classical technique of polynomial-time reductions to obtain hardness results for classes on the first and the second level of the polynomial hierarchy. If a decision problem—or a variant thereof—turns out to be in P, we present an efficient way to find the answer to a given instance of the problem, e.g., by designing an algorithm that solves the problem within polynomial time or by reducing the problem to another decision problem that is already known to be in P (e.g., we make use of known results from automata theory in order to obtain an efficient problem for a problem related to phylogenetics). In the following, we provide a brief introduction to the considered problems from phylogenetics and satisfiability. First, we turn our attention to phylogenetics.

Phylogenetics

Phylogenetics is a research area which studies questions related to the evolutionary relationships between extant species. A common way to represent these evolutionary relationships are phylogenetic trees and phylogenetic networks (for an extensive introduction to these topics see, e.g., Semple and Steel [SS03], Huson et al. [HRS10], and Steel [Ste16]).

Phylogenetic trees and networks

Phylogenetic trees are a tool to study the evolution of species and are used as such at least since 1837 when Charles Darwin considered this concept to understand the evolutionary relationships between species. More recently, phylogenetic trees and networks

are not only used in evolutionary biology but also to represent relationships between other entities such as languages, viruses, and cancer cells (cf. Bordewich et al. [BDLN20, p. 134f] and see, e.g., Willems et al. [WLL+16], Lam et al. [LHT10] and Schwartz and Schäffer [SS17], respectively).

In graph-theoretic terms, a *phylogenetic tree* is a rooted binary tree, where the leaves are bijectively labeled with species. The internal[3] vertices of a phylogenetic tree correspond to speciation events. A speciation event can be driven, e.g., by spatial separation of a species into two parts. However, phylogenetic trees are not sufficient to represent all processes that drive the evolution of species. In particular, processes like hybridization and lateral gene transfer cannot be represented in a tree-like fashion. A well-known example of hybridization is the evolutionary history of bread wheat which is the result of different species producing a common offspring (see, e.g., Petersen et al. [PSYB06] and Marcussen et al. [MSH+14]). An example of lateral gene transfer is the transmission of genes related to antibiotic resistance between different species of bacteria. The term "reticulation event" is used as an umbrella term for non-tree-like processes such as the two examples that we just alluded to. A *phylogenetic network* generalizes the concept of a phylogenetic tree, such that reticulation events can be represented. The latter is achieved by introducing vertices, referred to as *reticulations*, with in-degree at least 2 and out-degree 1. In this thesis, we focus on the *binary* case, where each reticulation has in-degree 2. Now, given a set $X = \{x_1, \ldots, x_n\}$ of extant species, a binary phylogenetic network on X is a rooted directed acyclic graph with the following properties:

- the root has in-degree 0 and out-degree 2,
- each non-root internal vertex has either in-degree 1 and out-degree 2 (*tree vertex*) or in-degree 2 and out-degree 1 (*reticulation*),
- each leaf has in-degree 1 and out-degree 0,
- the leaves are labeled bijectively with elements from X.

Each reticulation in a phylogenetic network corresponds to a reticulation event and each tree vertex corresponds to a speciation event. Moreover, a phylogenetic tree is a phylogenetic network with no reticulation. Figure 1.1 depicts an example of both a phylogenetic tree and a phylogenetic network. This figure and all following figures were created for this thesis, unless it is stated otherwise and a reference is provided. Throughout this work, we use the common convention that arcs are directed downwards

[3] An internal vertex is any vertex that is not a leaf.

Figure 1.1: A phylogenetic tree $\mathcal{T}$ (left-hand side) and a phylogenetic network $\mathcal{N}$ (right-hand side) on the leaf set $X = \{a, b, c, d\}$.

and omit the corresponding arrowheads in the visual representation. Further, we omit the term "binary" in the following since all considered networks are binary. The same applies to all trees throughout the thesis.

Now, even though the evolution of species may contain non-tree-like events as alluded to above, the ancestral history of different parts of a species' *genome* can often be represented by a single tree. Hence, the relationship between a phylogenetic network and its embedded phylogenetic trees is of interest in the study of certain biological questions. To further discuss this relationship, we need a definition. Let $\mathcal{N}$ be a phylogenetic network on a set X and let $\mathcal{T}$ be a phylogenetic tree on the same set. We say that $\mathcal{N}$ *embeds* (or *displays*) $\mathcal{T}$ if $\mathcal{T}$ can be obtained from $\mathcal{N}$ by deleting arcs and non-root vertices and then *suppressing* all resulting vertices of degree 2 (i.e., vertices of in-degree 1 and out-degree 1). Suppressing a vertex v of degree 2 means to delete v and both its incident arcs, and to create an arc from the parent of v to the child of v. An example of a tree displayed by a phylogenetic network is depicted in Figure 1.2 (suppressing vertices is done in the second step which is depicted by an accordingly labeled arc going from the middle to the right-hand side of the figure). The question whether, given a phylogenetic tree $\mathcal{T}$ and a phylogenetic network $\mathcal{N}$ on a set X, it is possible to embed $\mathcal{T}$ in the network $\mathcal{N}$ was shown to be NP-complete by Kanj et al. [KNTX08, Thm. 3.1]. On the positive side, the latter question is solvable in polynomial time for some classes of phylogenetic networks (see Section 3.2 for more details and references). Since reticulation events occur comparatively rarely (cf. Bordewich and Semple [BS07, p. 915]), another natural question related to the relationship between phylogenetic trees and networks is the following: Given a set of phylogenetic trees on a set X, what is the least amount of reticulation events required to construct a phylogenetic network that simultaneously embeds every tree in the input? Bordewich and Semple [BS07, Thm. 2.1] showed that this problem is APX-hard[4] and, consequently, NP-hard. The

[4]The class APX contains the optimization versions of decision problems in NP that admit an efficient algorithm computing an approximate solution which is bounded by a multiple of the optimum (in

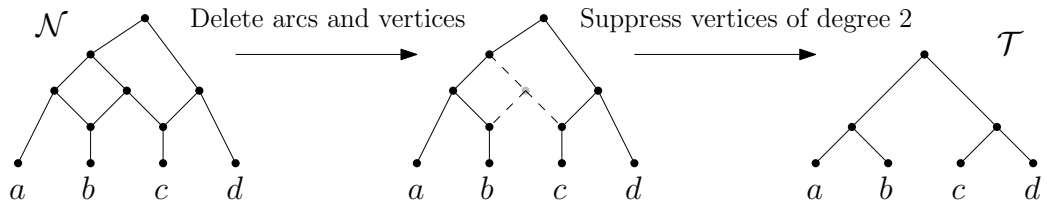


Figure 1.2: The network $\mathcal{N}$ displays the phylogenetic tree $\mathcal{T}$ on the right-hand side. This can be shown as follows: First, in $\mathcal{N}$, delete the vertex which is grayed out in the middle of the figure, and its incident arcs (depicted by dashed lines). Then, suppressing each resulting non-root vertex of degree 2 yields $\mathcal{T}$. In fact, the embedding of $\mathcal{T}$ in $\mathcal{N}$ is not unique (cf. Cordue et al. [CLS14]).

class of *temporal* networks[5] contains all phylogenetic networks of a particular class that additionally satisfy two temporal constraints which can be briefly summarized as follows: reticulation events occur instantaneously and speciation events occur successively (cf., e.g., [LSS10, Mor12, HLS13a]). Humphries et al. [HLS13a] characterized the existence of a temporal network that embeds a given collection $\mathcal{P}$ of phylogenetic trees on a set X by establishing a connection to the existence of a particular sequence on X. They called this sequence a *cherry-picking sequence* and showed that, in case of existence, a cherry-picking sequence also provides the minimum number of reticulations that are sufficient to construct a temporal network that displays each phylogenetic tree in $\mathcal{P}$.

Cherry-picking sequences

Let $\mathcal{T}$ be a phylogenetic tree on a set X. A cherry is a subset $\{x, y\} \subseteq X$ containing two distinct elements from X such that x and y have the same parent in $\mathcal{T}$ (e.g., $\{a, b\}$ and $\{c, d\}$ are cherries of the phylogenetic tree depicted in Figure 1.1). Now, let $\mathcal{P} = \{\mathcal{T}_1, \ldots, \mathcal{T}_m\}$ be a collection of phylogenetic trees on the same set $X = \{x_1, \ldots, x_n\}$. Then, a cherry-picking sequence is a permutation (or ordering)

$$(x_{i_1}, x_{i_2}, \ldots, x_{i_n})$$

of the elements of X which satisfies certain properties. We introduce these properties using the example shown in Figure 1.3 where $\mathcal{P}$ consists of two phylogenetic trees $\mathcal{T}_1$ and $\mathcal{T}_2$ on the set $X = \{a, b, \ldots, h\}$ (for the formal definition and an example with $m > 2$

other words, the found solution is bounded by the optimum multiplied with some constant factor). Hardness for this class is defined by reductions that preserve certain approximation properties (see, e.g., Alimonti and Kann [AK00, p. 124f] for more details regarding these reductions).

[5] We formally define temporal networks [MNW+04] and other network classes in Section 3.2

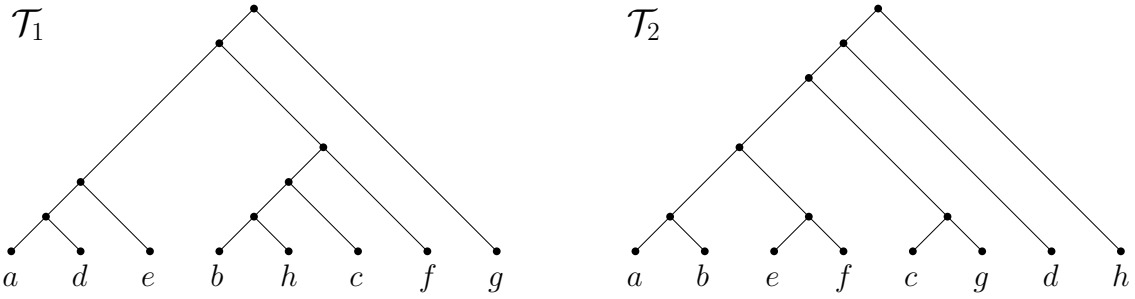


Figure 1.3: Two phylogenetic trees $\mathcal{T}_1$ and $\mathcal{T}_2$ on $X = \{a, b, c, d, e, f, g, h\}$ for which a cherry-picking sequence exists (e.g., the ordering $a < e < b < c < f < d < g < h$). These two trees were constructed in Döcker and Linz [DL18, Fig. 2] as a gadget enforcing that a never occurs between b and c in any cherry-picking sequence.

phylogenetic trees see Section 3.2). Let us consider the ordering

$$\mathcal{O} = (a,\, e,\, b,\, c,\, f,\, d,\, g,\, h)$$

of the elements in X. First, observe that the first element, i.e., a, is in a cherry of both $\mathcal{T}_1$ and $\mathcal{T}_2$. By deleting the leaf labeled with a and its incident arc in both phylogenetic trees and subsequently suppressing the two resulting vertices—one in each phylogenetic tree—of in-degree 1 and out-degree 1, we obtain two phylogenetic trees $\mathcal{T}_1'$ and $\mathcal{T}_2'$ on $X' = X \setminus \{a\}$ (this reduction step is illustrated in Figure 1.4). Now, the second element in $\mathcal{O}$, i.e., e, is in a cherry of both $\mathcal{T}_1'$ and $\mathcal{T}_2'$. Again, we reduce both networks in the same way as in the previous reduction step (delete e instead of a) to obtain two phylogenetic trees $\mathcal{T}_1''$ and $\mathcal{T}_2''$, where the third element b is in a cherry of both $\mathcal{T}_1''$ and $\mathcal{T}_2''$. If we can continue in this way until both phylogenetic trees are reduced to a cherry, then $\mathcal{O}$ is a cherry-picking sequence. It is now straightforward to verify that $\mathcal{O}$ is indeed a cherry-picking sequence for $\mathcal{T}_1$ and $\mathcal{T}_2$.

Comparing display sets

Let $\mathcal{N}$ and $\mathcal{N}'$ be two phylogenetic networks on the same set X. The set consisting of all phylogenetic trees that are displayed by $\mathcal{N}$ (resp. $\mathcal{N}'$) is called the *display set*[6] of $\mathcal{N}$ (resp. $\mathcal{N}'$) and we write $T(\mathcal{N})$ (resp. $T(\mathcal{N}')$) to denote this set. An open question stated in Gunawan et al. [GDZ17] concerns the computational complexity of deciding whether $T(\mathcal{N}) = T(\mathcal{N}')$ for a restricted class of phylogenetic networks—we provide more details on this in Section 3.2. Now, Figure 1.5 illustrates how the display set

[6] We introduced the term "display set" in Döcker et al. [DLS19, DLS21].

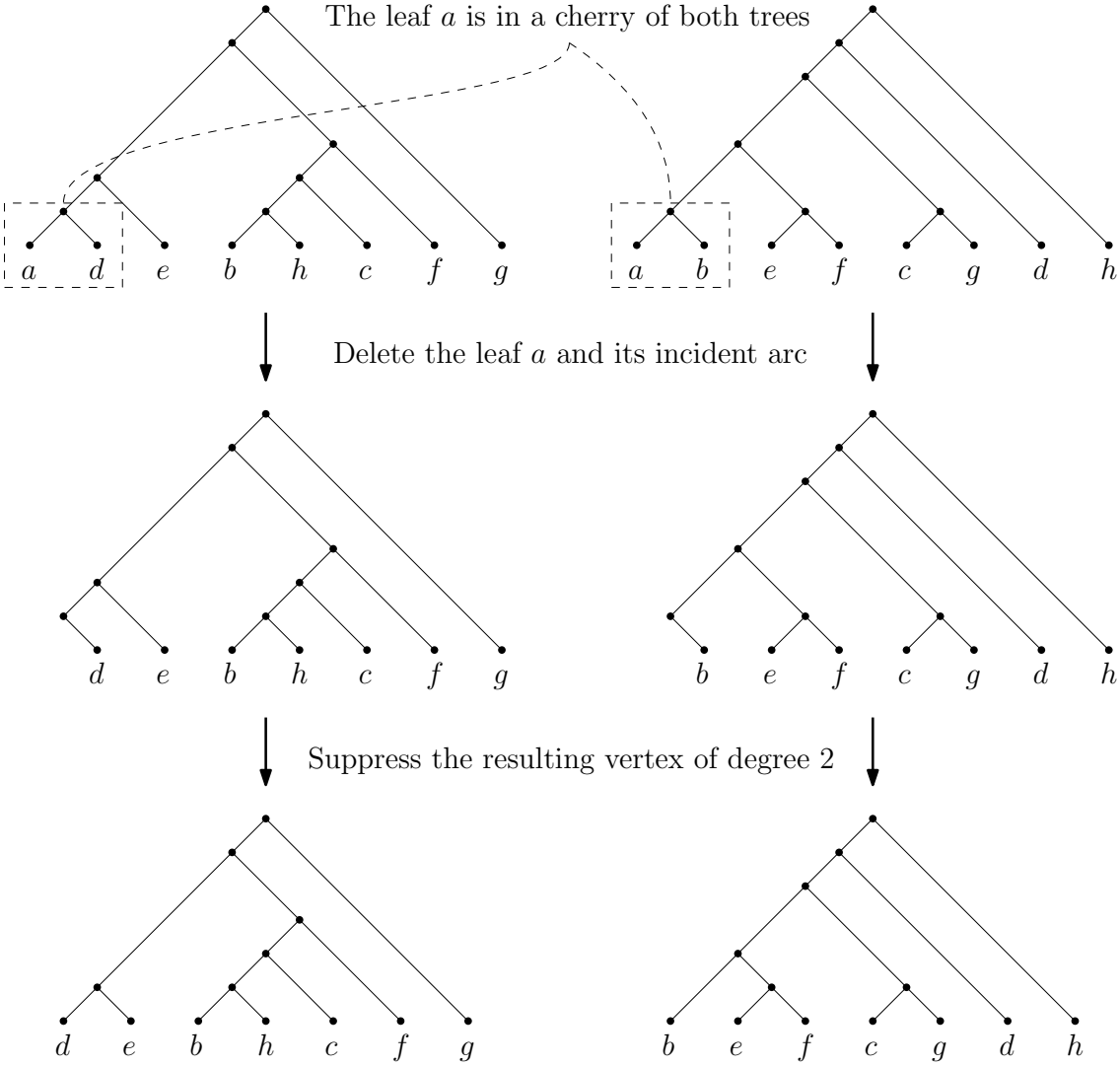


Figure 1.4: Reducing phylogenetic trees with respect to a cherry-picking sequence.

of the phylogenetic network depicted in Figure 1.2 can be obtained. Intuitively, each reticulation in a phylogenetic network introduces a choice, since, in order to obtain a phylogenetic tree, we must delete at least one of the incoming arcs of each reticulation. In Section 3.2.3 we present our algorithmic and complexity results for several ways to compare the display sets of two phylogenetic networks $\mathcal{N}$ and $\mathcal{N}'$ on X. Specifically, we analyze the computational complexity of deciding the following:

- $T(\mathcal{N}) \cap T(\mathcal{N}') \neq \emptyset$ (Is there a phylogenetic tree displayed by both $\mathcal{N}$ and $\mathcal{N}'$?)
- $T(\mathcal{N}) \subseteq T(\mathcal{N}')$ (Is each phylogenetic tree displayed by $\mathcal{N}$ also displayed by $\mathcal{N}'$?)
- $T(\mathcal{N}) = T(\mathcal{N}')$ (Do $\mathcal{N}$ and $\mathcal{N}'$ display the same phylogenetic trees?)

Figure 1.6 shows an example of two phylogenetic networks $\mathcal{N}$ and $\mathcal{N}'$ on $X = \{x_1, x_2, x_3\}$ for which the answer to all three of the above questions is "yes". Note that a positive answer to the latter question implies a positive answer to the former two questions.

Satisfiability

The Boolean satisfiability problem concerns particular logical expressions—which are called Boolean formulas—that can be formed over a set of variables $V = \{x_1, x_2, \ldots, x_n\}$ taking values in $\{\text{True}, \text{False}\}$ using logical connectives (including the unary negation operator) and parentheses. For instance, the expression

$$\varphi = x_1 \wedge (x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3)$$

is a Boolean formula, where "$\vee$" is the "logical or" connective, "$\wedge$" is the "logical and" connective, and $\overline{x_i}$ denotes the negation of x_i. The elements in $\mathcal{L} = \{x_i, \overline{x_i} \mid x_i \in V\}$ are called *literals*; and a *clause* is a disjunction of literals (e.g., $\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}$). A Boolean formula which is a conjunction of clauses is in *conjunctive normal form* (CNF). Observe that the Boolean formula φ above is in CNF. For Boolean formulas in CNF, it is also common to use set notation, where a clause is a subset of $\mathcal{L}$ and a Boolean formula is then written as a set of clauses. For instance, φ in set notation reads

$$\varphi = \{\{x_1\}, \{x_2, \overline{x_3}\}, \{\overline{x_1}, \overline{x_2}, \overline{x_3}\}, \{\overline{x_1}, x_2, x_3\}\}.$$

Noting that all Boolean formulas considered in this thesis are in CNF, we omit stating this each time (i.e., if not mentioned otherwise, a Boolean formula is always assumed

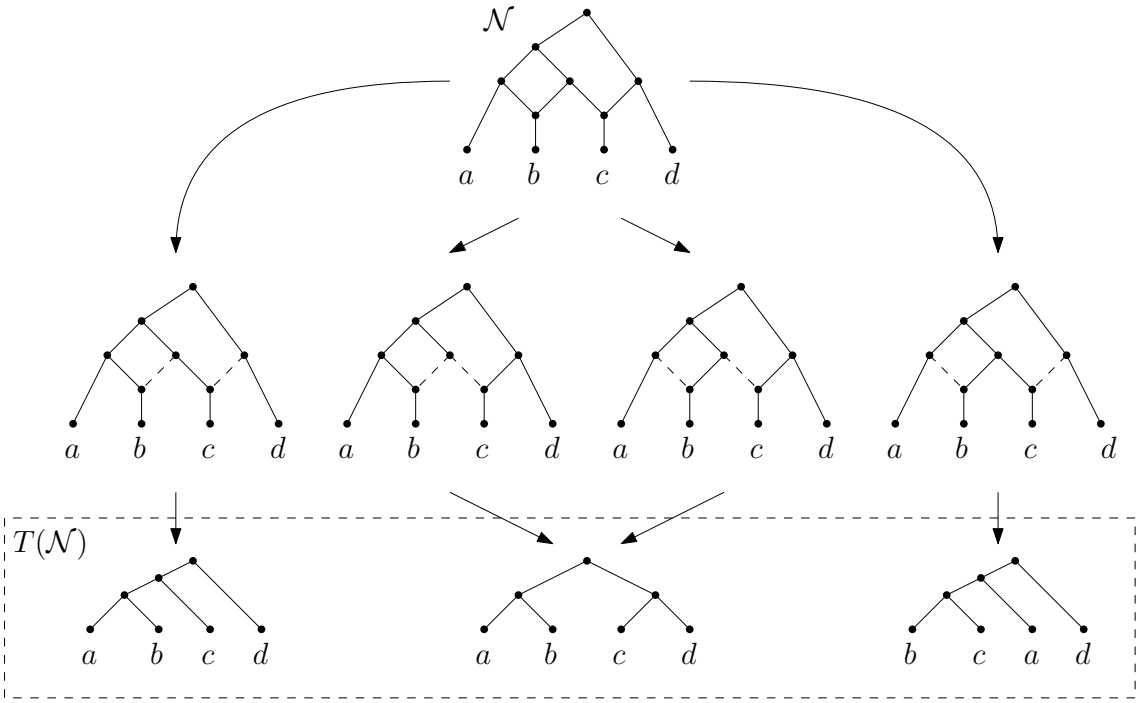


Figure 1.5: The display set $T(\mathcal{N})$ of the phylogenetic network $\mathcal{N}$ on $X = \{a, b, c, d\}$ that was used in Figure 1.2 to illustrate the embedding of a phylogenetic tree in a phylogenetic network. First, we obtain the $2^2 = 4$ graphs depicted in the second row of the above figure by deleting precisely one incoming arc of each reticulation (the deleted arc is indicated by a dashed line). Then, deleting the vertex with two dashed outgoing arcs along with its incident arc and subsequently suppressing vertices of in-degree 1 and out-degree 1 yields $T(\mathcal{N})$.

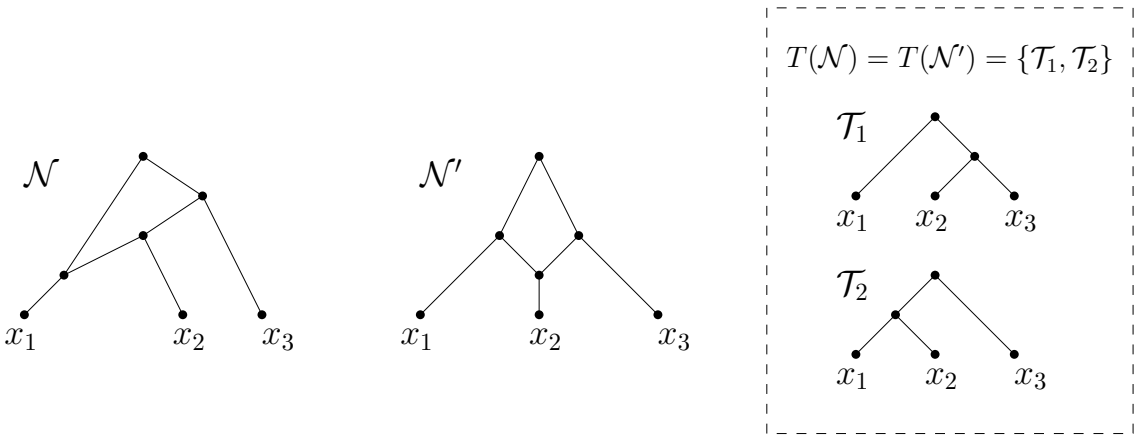


Figure 1.6: Two phylogenetic networks $\mathcal{N}$ and $\mathcal{N}'$ that have the same display set $\{\mathcal{T}_1, \mathcal{T}_2\}$.

to be in CNF). Now, a Boolean formula is satisfiable if there is a truth assignment $\beta\colon V \to \{\text{True}, \text{False}\}$ such that in each clause, at least one literal evaluates to True (a literal $\overline{x_i}$ evaluates to True if and only if $\beta(x_i) = \text{False}$). The satisfiability problem (SATISFIABILITY) for Boolean formulas can now be stated as follow: Given a Boolean formula φ over a set of variables V, is there a truth assignment $\beta\colon V \to \{\text{True}, \text{False}\}$ that satisfies φ? This question turns out to be very hard to solve in general as Cook [Coo71] showed completeness of SATISFIABILITY for the complexity class NP. Further, this hardness result turned out to be particularly useful since NP-hardness results for many other decision problems were later derived either directly or indirectly from the established NP-completeness of SATISFIABILITY by means of polynomial-time reductions (see, e.g., Karp [Kar72] for many classical results using this transformation technique). For more details on the theory of NP-completeness see the book by Garey and Johnson [GJ79]; we also provide a brief historical overview of the satisfiability problem in Section 3.3.2.

A lot of research was done on the computational complexity of variants of SATISFIABILITY (see Filho [Fil19] for an extensive collection of related results). For instance, SATISFIABILITY remains NP-complete [GJ79, p. 48f] if each clause contains precisely three distinct literals formed over pairwise distinct variables—this popular variant of SATISFIABILITY is called 3-SAT—but it can be solved in linear time if each clause contains at most two literals [EIS76, p. 696f]. Several natural restrictions for 3-SAT were considered in the literature that further refined this complexity analysis, e.g.,

- monotone clauses, i.e., a clause has the form $\{x_i, x_j, x_k\}$ or $\{\overline{x_i}, \overline{x_j}, \overline{x_k}\}$ (the corresponding variant MONOTONE 3-SAT is known to be NP-complete [Gol78, Li97]),
- bounded variable appearances (see, e.g., Tovey [Tov84] and Berman et al. [BKS03]),
- planar incidence graph, i.e., a certain graph—such as the one shown in Figure 1.7—associated with a Boolean formula is required to admit a planar drawing (the corresponding variant PLANAR 3-SAT is known to be NP-complete [Lic82, Man83]).

Note that for the complexity result for MONOTONE 3-SAT and PLANAR 3-SAT, respectively, we included two different references. This is due to the fact that Gold [Gol78] and Lichtenstein [Lic82] obtained the first results for these restrictions, but for a slightly less strict definition of 3-SAT where clauses contain *at most* three distinct literals. Another possibility to obtain variants of SATISFIABILITY is to redefine which properties of a truth assignment are desirable, e.g., asking whether there is a truth assignment such that at least one but not all literals contained in each clause evaluate to True. The

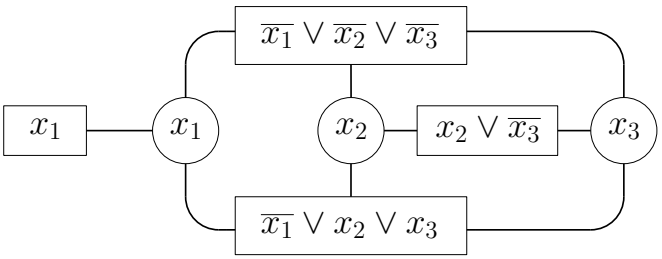


Figure 1.7: A planar drawing of the incidence graph for the Boolean formula $\varphi = x_1 \wedge (x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3)$. In the incidence graph, each variable x_i (resp. clause c_j) is depicted as a circular (resp. rectangular) vertex, and there is an edge connecting x_i and c_j if and only if x_i appears in c_j. It is also common to consider the additional requirement that vertices representing the variables are connected by a cycle (e.g., the graph associated with a Boolean formula in the first publication [Lic82] on the planar variants has this property).

latter variant goes by the name Not-All-Equal Satisfiability and was shown by Schaefer [Sch78] to be NP-complete even if each clause contains at most three literals (we provide more known results for this variant in Section 3.3).

Generalizations of Satisfiability—specifically, quantified variants of the Boolean satisfiability problem—turned out to be useful to obtain canonical complete problems for complexity classes beyond NP, i.e., for classes that contain NP but may also contain further decision problems (see, e.g., Meyer and Stockmeyer [MS72], Stockmeyer [Sto76], Wrathall [Wra76] and Garey and Johnson [GJ79, Sec. 7.2]). We illustrate a quantified Boolean formula and the corresponding decision problem using the following example:

$$\Phi = \forall x_1 \exists x_2 \exists x_3 [(x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3)].$$

The quantified Boolean formula Φ is considered to be true if for both possibilities to assign a truth value to x_1, there are truth values for x_2 and x_3 such that the unquantified part between the brackets evaluates to True; the task is then to decide whether Φ is true. Now, consider the example Φ above. First, observe that setting all variables to False satisfies each clause. Second, for $\beta(x_1) =$ True, we can satisfy all clauses by setting $\beta(x_2) =$ True and $\beta(x_3) =$ False. Hence, the quantified Boolean formula Φ is true. In contrast, the quantified Boolean formula

$$\Phi' = \forall x_1 \forall x_2 \exists x_3 [(x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3)]$$

is false since, for $\beta(x_1) =$ True and $\beta(x_2) =$ False, there is no assignment of a truth value to x_3 that satisfies the first and the third clause simultaneously. The computational com-

plexity class in which solving such quantified Boolean formulas can be placed depends on the type of the first (i.e., leftmost) quantifier and the number of allowed alternations between an existential and a universal quantifier (for more details on the corresponding system of complexity classes, i.e., the polynomial hierarchy, see Section 3.1).

Structure of the thesis

The remainder of this thesis is structured as follows: Chapter 2 is dedicated to the objectives of the conducted research and also provides some of the background story regarding the development of the thesis over time. Chapter 3 consists of four parts: First, we provide necessary preliminaries on the polynomial hierarchy in Section 3.1. Second, in Section 3.2, we present our complexity results for the considered decision problems from phylogenetics which either concern the existence of a cherry-picking sequence for a collection of phylogenetic trees or the comparison of the display sets of two phylogenetic networks. Third, in Section 3.3, we present our classification results for several restricted variants of the (quantified) satisfiability problem where bounded variable appearances are a common theme. In this section, and also in Section 3.2, we provide preliminaries on the respective topics including formal definitions of some concepts that we already introduced in an intuitive manner in the introduction. Fourth, in Section 3.4, we close the thesis with concluding remarks including comments on possible directions for future research. Finally, the appendix contains the manuscripts that are part of this thesis.

2 Objectives

Initially, this thesis was planned as a continuation and broadening of the research I conducted in my master's thesis [Dö14] on combinatorial auctions (here, broadening means the additional consideration of other problems arising in the context of computational social choice). While I contributed to several articles on this topic (specifically, on combinatorial auctions [DDEK16], tool auctions [DDE+18] and the simplified group activity selection problem [DDD+17, DDD+18b, DDDS21]) during my PhD studies, these papers are not included in my dissertation as the focus of my thesis changed quite early on in the process and it did not seem feasible to combine all papers in a natural way.

My research was pushed in a new direction when my PhD advisor Britta Dorn got me in touch with Simone Linz (University of Auckland) which sparked a fruitful collaboration resulting in five of the nine manuscripts included in this thesis. We set out to tackle the problem of deciding whether two tree-child networks display precisely the same phylogenetic trees which was my first exposure to the fascinating world of phylogenetics (the computational complexity of deciding this question also appears as an open problem for reticulation-visible networks in Gunawan et al. [GDZ17]). In 2017, I followed the invitation from Simone Linz to come to Auckland, New Zealand, and jointly work with her on this problem. Despite significant efforts, the problem withstood all our attempts to approach it. Fortunately, Simone Linz had the idea to consider another problem from phylogenetics—related to the reconstruction of phylogenetic networks—which asks whether a cherry-picking sequence exists for two phylogenetic trees (the computational complexity of this problem was stated as an open problem in Humphries et al. [HLS13a]). Our first goal was to consider this question for a constant number of phylogenetic trees as we conjectured that this problem would turn out to be NP-complete for some constant. In the following year, following another kind invitation from Simone Linz, I came to New Zealand for a second time as a visiting student, where we—together with Charles Semple (University of Canterbury)—revisited the former problem on the display sets of two phylogenetic networks. In summary, the objective was to classify the computational complexity of the problems related to the display sets and the existence of a

cherry-picking sequence in a precise way by placing them in their respective level on the polynomial hierarchy (specifically, we considered the classes P, NP, co-NP and Π_2^P).

The second part of my thesis was inspired by a conjecture [DKY14] on the computational complexity of MONOTONE 3-SAT with at most five appearances of each variable. After confirming this conjecture (joint work with Andreas Darmann and my PhD advisor Britta Dorn), we extended the focus on other restrictions for variants of the satisfiability problem while keeping the bounded variable appearances a common theme. We also considered restricted variants of the Π_2^P-complete quantified counterparts of 3-SAT and NOT-ALL-EQUAL 3-SAT. The goal was again to place all these decision problems in the polynomial hierarchy (as for the problems from phylogenetics, the relevant complexity classes for the problems we considered turned out to be P, NP, co-NP and Π_2^P).

3 Discussion of Results

In this chapter, we present and discuss our results that place problems from phylogenetics and satisfiability in the polynomial hierarchy. First, we provide relevant preliminaries on the polynomial hierarchy. Second, we turn our attention to decision problems from phylogenetics, specifically to the problem of deciding the existence of a cherry-picking sequence and questions related to the display sets of phylogenetic networks. Third, we present our complexity results for several variants of the Boolean satisfiability problem. Finally, we make concluding remarks including possible directions for future research.

3.1 Preliminaries on the polynomial hierarchy

The polynomial hierarchy [Sto76] consists of a hierarchy of complexity classes which are defined recursively as follows:

$$\Sigma_0^P = \Pi_0^P = \mathrm{P},$$

and, for all $k \geq 0$,

$$\Sigma_{k+1}^P = \mathrm{NP}^{\Sigma_k^P} \quad \text{and} \quad \Pi_{k+1}^P = \text{co-NP}^{\Sigma_k^P},$$

where a problem is in Σ_{k+1}^P (resp. Π_{k+1}^P) if the problem is in NP (resp. co-NP) when we are given access to an *oracle* for Σ_k^P that outputs the answer to any problem in Σ_k^P in constant time. By definition, we have $\Sigma_1^P = \mathrm{NP}$ and $\Pi_1^P = \text{co-NP}$. Recall that NP (resp. co-NP) is the class of decision problems where an appropriate certificate of a yes-instance (resp. no-instance) can be verified in polynomial time and P is the class of decisions problems that have polynomial-time decision procedures (i.e., we can find the answer in polynomial-time with respect to the input size). Σ_{k+1}^P- and Π_{k+1}^P-hardness are defined with respect to polynomial-time many-one reductions, which are also called Karp reductions, and there are canonical complete problems—quantified variants of the satisfiability problem [Wra76]—for each of the classes Σ_k^P (resp. Π_k^P), $k \geq 1$. There are several interesting and important open problems related to the polynomial hierarchy. In particular, it is unknown whether $\Sigma_{k+1}^P \neq \Sigma_k^P$ or $\Pi_{k+1}^P \neq \Pi_k^P$ for any $k \geq 0$. Note that,

for $k = 0$, the question whether $\Sigma_{k+1}^P \neq \Sigma_k^P$ is the fundamental P versus NP problem. The results in this work concern the classes P, NP, co-NP and Π_2^P. For an in-depth introduction into the theoretical foundations of the polynomial hierarchy we refer to Stockmeyer [Sto76], Wrathall [Wra76], Garey and Johnson [GJ79], and de Haan [dH19].

Canonical Π_2^P-complete problems

In the following, we use notation related to the Boolean satisfiability problem that is formally introduced in Section 3.3 (e.g., a truth assignment β *nae-satisfies* a clause if the clause has both a true and a false literal under β). The following problems are canonical problems for the complexity class Π_2^P. They are quantified variants of 3-SAT and Not-All-Equal 3-SAT and have both been shown to be Π_2^P-complete (see Stockmeyer [Sto76] and Eiter and Gottlob [EG95], respectively).

∀∃ 3-Sat [DDLS20, p. 74]
Instance. A quantified Boolean formula

$$\forall x_1 \cdots \forall x_p \exists x_{p+1} \cdots \exists x_n \bigwedge_{j=1}^{m} c_j$$

over a set $V = \{x_1, x_2, \ldots, x_n\}$ of variables, where each clause c_j is a disjunction of at most three literals.
Question. For every truth assignment for $\{x_1, x_2, \ldots, x_p\}$, does there exist a truth assignment for $\{x_{p+1}, x_{p+2}, \ldots, x_n\}$ such that each clause of the formula is satisfied?

∀∃ Not-All-Equal 3-SAT (∀∃ NAE-3-SAT) [DDLS20, p. 75]
Instance. A quantified Boolean formula

$$\forall x_1 \cdots \forall x_p \exists x_{p+1} \cdots \exists x_n \bigwedge_{j=1}^{m} c_j$$

over a set $V = \{x_1, x_2, \ldots, x_n\}$ of variables, where each clause c_j is a disjunction of at most three literals.
Question. For every truth assignment for $\{x_1, x_2, \ldots, x_p\}$, does there exist a truth assignment for $\{x_{p+1}, x_{p+2}, \ldots, x_n\}$ such that each clause of the formula is nae-satisfied?

An important characteristic of these two problems is that there are two types of variables, namely *universal* variables and *existential* variables. Further, there is precisely

one alternation between universal and existential variables. In particular, the first p variables in the list of quantifiers are universal, which are followed by $n - p$ existential variables. As already mentioned in the first paragraph of Section 3.1, for all $k \geq 0$, canonical complete problems exist for Σ_{k+1}^P and Π_{k+1}^P, respectively. Specifically, the quantified variant of 3-SAT with k alternations between universal and existential quantifiers and where the list of quantifiers starts with a universal variable (resp. existential variable) is Π_{k+1}^P-complete (resp. Σ_{k+1}^P-complete) [Wra76]. Note that, for $k = 0$, this states that the quantified variant of 3-SAT is co-NP-complete if all variables are universal variables and NP-complete if all variables are existential variables. Indeed, this is just another way to define the classical 3-SAT problem and its co-NP counterpart (in the first problem, a certificate of a yes-instance is a truth assignment that satisfies the corresponding formula and in the latter problem, a satisfying truth assignment certifies a no-instance).

3.2 Phylogenetics

In this section, we give an overview of our results concerning different problems arising in phylogenetics. As alluded to in the introduction, a phylogenetic network is a common formalism to represent ancestral relationships between different taxa (e.g., species). Now, each phylogenetic network $\mathcal{N}$ embeds a collection of phylogenetic trees $T(\mathcal{N})$ which we call the *display set* of $\mathcal{N}$. A natural and well-studied problem is the following one:

Tree Containment [DLS19, p. 131]
Instance. A phylogenetic tree $\mathcal{T}$ and a phylogenetic network $\mathcal{N}$ on X.
Question. Is $\mathcal{T} \in T(\mathcal{N})$?
(In words: Is $\mathcal{T}$ contained in the display set of $\mathcal{N}$?)

In other words, given a phylogenetic tree $\mathcal{T}$ and a phylogenetic network $\mathcal{N}$ on X, the question is whether $\mathcal{T}$ is embedded in $\mathcal{N}$. Tree Containment was shown to be NP-complete by Kanji et al. [KNTX08]. While this problem is known to remain NP-complete in restricted settings (i.e., certain classes of phylogenetic networks) [vISS10], it can be solved in polynomial time on some popular classes of phylogenetic networks such as tree-child networks and reticulation-visible networks [vISS10, BS16, GDZ17]. In two of our articles [DLS19, DLS21] (which we discuss in Section 3.2.3), we analyze the computational complexity of problems related to Tree Containment. Specifically, we compare the display sets of two phylogenetic networks $\mathcal{N}$, $\mathcal{N}'$ with respect to the following questions:

- $T(\mathcal{N}) \cap T(\mathcal{N}') \neq \emptyset$? (Common-Tree-Containment)
- $T(\mathcal{N}) \subseteq T(\mathcal{N}')$? (Display-Set-Containment)
- $T(\mathcal{N}) = T(\mathcal{N}')$? (Display-Set-Equivalence)

We note that Common-Tree-Containment (resp. Display-Set-Containment) is a generalization of Tree Containment. Further, Gunawan et al. [GDZ17] previously asked the question whether $T(\mathcal{N}) = T(\mathcal{N}')$ can be tested in polynomial time for two reticulation visible networks. While this problem is—to the best of our knowledge—still open as of writing this thesis, we were able to show that the question is Π_2^P-complete for two general networks and can be decided in polynomial time if the input consists of a normal network and a tree-child network [DLS19, DLS21]. It is worthwhile to go the extra mile and prove Π_2^P-completeness instead of stopping at NP-hardness because the former result has negative implications for the usefulness of encoding such a problem as

an NP-complete problem for which powerful solvers exist (e.g., SAT) since there is no polynomial transformation from a Π_2^P-complete problem to a problem in NP unless the polynomial hierarchy collapses (cf., e.g., de Haan and Szeider [dHS19, Sec. 2.3]). It is also worth pointing out that TREE CONTAINMENT being in P for some restricted class of phylogenetic networks does not necessarily imply that any of the comparisons described above can be done in polynomial time for two networks of this class (e.g., we show that COMMON-TREE-CONTAINMENT is NP-complete [DLS19, Thm. 3.2] for a class that has a known polynomial-time algorithm for TREE CONTAINMENT).

Further, we present several complexity results for a problem related to the reconstruction of phylogenetic networks given a set of phylogenetic trees. In the reconstruction problem, the task is to construct a phylogenetic network $\mathcal{N}$ (possibly of a certain class) such that the display set $T(\mathcal{N})$ contains all phylogenetic trees of the input. While such a network always exists if we do not restrict the search space to a particular class of phylogenetic networks (in fact, for any given set X of taxa with $|X| \geq 2$, there exists a phylogenetic network that displays all rooted binary phylogenetic trees on X [FS15, Prop. 4]), this is not necessarily the case otherwise. For instance, for temporal networks [MNW+04] where some natural assumptions with respect to the flow of time—such as the co-existence of two species that generate a hybrid offspring—are represented, such a network may or may not exist depending on the input (see, e.g., Humphries et al. [HLS13b, Fig. 2] for an example of two phylogenetic trees that cannot both be displayed by the same temporal network). Hence, for such classes the natural decision problem arises that asks whether a network of the considered class exists for a given set of phylogenetic trees. Specifically, for temporal networks, Humphries et al. [HLS13a] established a characterization[1] based on the existence of a cherry-picking sequence which is a certain elimination order on the taxa constrained by the given phylogenetic trees. In particular, they show that a cherry-picking sequence exists for a set of phylogenetic trees $\mathcal{P}$ if and only if there is a temporal network $\mathcal{N}$ with $\mathcal{P} \subseteq T(\mathcal{N})$ [HLS13a, Thm. 1]. Furthermore, the authors left the computational complexity of deciding the existence of a cherry-picking sequence—we call this decision problem CPS-EXISTENCE—as an interesting open question to investigate for the case $|\mathcal{P}| = 2$.

As a first step, we showed that CPS-EXISTENCE is NP-complete [DL18] if $|\mathcal{P}| = m$ for each fixed $m \geq 8$. In the same article, we presented a polynomial-time algorithm for CPS-EXISTENCE if the number of phylogenetic trees (i.e., m) and the number of

[1] To be precise, the characterization established by Humphries et al. [HLS13a] is concerned with the computation of a solution with a minimum number of reticulation events.

cherries, a certain structure that we will define later, in each tree of the input are bounded by a constant. In order to obtain the latter result, we explored connections to automata theory, an approach which has rarely been used so far to solve problems from phylogenetics (for instance, in Hall and Klein [HK10]). In a subsequent paper [DvIKL19], we were able to answer the open question of Humphries et al. [HLS13a] alluded to in the preceding paragraph by showing that CPS-EXISTENCE is NP-complete even if $|\mathcal{P}| = 2$. Hence, even for two phylogenetic trees it is NP-complete to decide whether there exists a temporal network that embeds both trees.

3.2.1 Preliminaries

In the following, we provide preliminaries and notation for the subsequent discussion of our results for several decision problems from phylogenetics. Here, we focus on the aspects that are crucial to the discussion of our results. For an extensive introduction to phylogenetics see, e.g., the book by Semple and Steel [SS03] and the book by Huson et al. [HRS10]. Moreover, we assume the reader to be familiar with basic concepts from graph theory and refer to Bondy and Murty [BM08] for an introduction to this topic.

Phylogenetic trees and networks

A *rooted binary phylogenetic X-tree* on a set X is a rooted directed tree with the following properties:

(1) there is a unique vertex (the *root*) with in-degree 0 and out-degree 2,

(2) each non-root internal[2] vertex has in-degree 1 and out-degree 2,

(3) all other vertices (called *leaves*) have in-degree 1 and out-degree 0, and

(4) the leaves are labeled with the elements from X (the labeling can be represented by a bijective mapping from the *leaf-set*, i.e., the set containing all leaves, to X).

A *phylogenetic network* is an acyclic digraph (a digraph is a directed graph), which has properties (1), (3) and (4) of a rooted binary phylogenetic X-tree and, additionally, each non-root internal vertex has in-degree 1 and out-degree 2 or in-degree 2 and out-degree 1 (vertices with the latter property are called *reticulations*). Note that a phylogenetic network without reticulations satisfies all properties of a rooted binary phylogenetic X-tree. Thus, phylogenetic networks generalize the latter concept. To improve readability,

[2]Recall that an internal vertex is any vertex that is not a leaf.

we will sometimes write phylogenetic tree instead of rooted binary phylogenetic X-tree as all trees considered in this thesis are rooted and binary (the set X is omitted if it is clear from the context or not relevant).

Network classes

Let $\mathcal{N}$ be a phylogenetic network. A vertex of $\mathcal{N}$ is called a *tree vertex* if it has in-degree 1 and out-degree 2. We say that $\mathcal{N}$ is a *tree-child network* if each non-leaf vertex has a child that is a tree vertex (hence the name tree-child) or a leaf. Tree-child networks were introduced by Cardona et al. [CRV09]. An arc (u, v) of $\mathcal{N}$ is a *shortcut* if there is a directed path from u to v that does not contain (u, v). The class of *normal networks* which was introduced by Willson [Wil10] consists of tree-child networks without shortcuts. Moret et al. [MNW$^+$04] introduced the class of *temporal networks*: An arc (u, v) of a phylogenetic network $\mathcal{N}$ is a *reticulation arc* if v is a reticulation, otherwise we say that (u, v) is a *tree arc*. Then, $\mathcal{N}$ is a temporal network if $\mathcal{N}$ is a tree-child network and there is a mapping t that assigns time-stamps (i.e., positive real numbers) to the vertices of $\mathcal{N}$ such that the following conditions are satisfied:

(i) $t(u) = t(v)$ if (u, v) is a reticulation arc and

(ii) $t(u) < t(v)$ if (u, v) is a tree arc.

A mapping t with these properties is called a *temporal labeling*.

Display sets

A phylogenetic X-tree $\mathcal{T}$ is *displayed* by a phylogenetic network $\mathcal{N}$ (we also say $\mathcal{N}$ *embeds* $\mathcal{T}$) if $\mathcal{T}$ can be obtained from $\mathcal{N}$ by deleting vertices and arcs and repeatedly *suppressing* vertices with in-degree 1 and out-degree 1. Here, suppressing a vertex v with a single parent p_v and a single child c_v means to replace v and its two incident arcs with the arc (p_v, c_v). The *display set* $T(\mathcal{N})$ of a phylogenetic network $\mathcal{N}$ is the set containing all phylogenetic trees that are displayed by $\mathcal{N}$. Specifically, a tree $\mathcal{T}$ is displayed by $\mathcal{N}$ if $\mathcal{T} \in T(\mathcal{N})$.

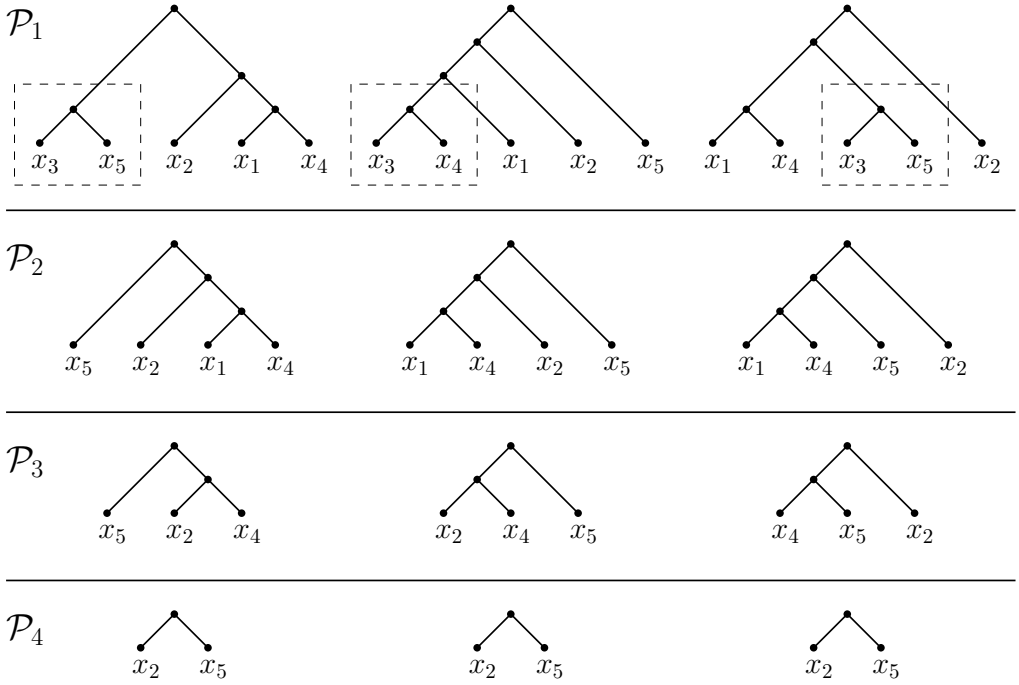


Figure 3.1: The permutation $(x_3, x_1, x_4, x_5, x_2)$ of $X = \{x_1, \ldots, x_5\}$ is a cherry-picking sequence for the three phylogenetic trees depicted in the first row. The rows below are obtained by successively reducing the trees according to the definition of a cherry-picking sequence. Note that, for each phylogenetic tree in $\mathcal{P}_1$, the leaf x_3 is in a cherry (marked by a dashed box in the first row of the figure above). Analogously, x_1 (resp. x_4) is in a cherry for each phylogenetic tree in $\mathcal{P}_2$ (resp. $\mathcal{P}_3$).

Cherry-picking sequences

A *cherry* of a tree $\mathcal{T}$ on X is a set containing two leaves of X that have the same parent. Let $\mathcal{P} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_m\}$ be a collection of phylogenetic X-trees. Note that each phylogenetic tree in $\mathcal{P}$ is formed over the same set of taxa X. A *cherry-picking sequence* for $\mathcal{P}$ is a permutation $(x_1, x_2, \ldots, x_{|X|})$ of X with the following properties:

(i) x_1 is contained in a cherry of each phylogenetic tree in $\mathcal{P}_1 = \mathcal{P}$ and,

(ii) for $2 \leq j \leq |X| - 1$, x_j is in a cherry of each phylogenetic tree in $\mathcal{P}_j$ where $\mathcal{P}_j$ is obtained from $\mathcal{P}_{j-1}$ by deleting x_{j-1} in each phylogenetic tree in $\mathcal{P}_{j-1}$ and suppressing the resulting vertex with in-degree 1 and out-degree 1.

For an example of a cherry-picking sequence for three phylogenetic trees see Figure 3.1.

Automata and formal languages

An *alphabet* is a finite set Σ and a *word* is a finite string that can be formed by concatenating symbols of Σ (repetitions of symbols within a word are allowed). The set Σ^* containing all possible words that can be formed over Σ is expressed using the Kleene star operator $*$. Now, a *language* is a subset of Σ^* (i.e., a language is a subset of the words that can be formed over an alphabet). A *deterministic finite automaton* (abbreviated DFA) is an abstract model used to recognize languages of a certain class (regular languages). Formally, a DFA is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_{\text{ini}}, F)$, where

(i) Q is a finite set of states,

(ii) Σ is a finite alphabet,

(iii) $\delta\colon Q \times \Sigma \to Q$ is a transition relation,

(iv) q_{ini} is the initial state, and

(v) $F \subseteq Q$ are final states.

Given a word $w \in \Sigma^*$, a DFA $\mathcal{A}$ reads the symbols of w from left to right and changes its state according to the transition relation δ. If $\mathcal{A}$ is in a final state after processing all symbols of w, we say that $\mathcal{A}$ *accepts* w. The language consisting of all words accepted by $\mathcal{A}$ is denoted with $\mathcal{L}(\mathcal{A})$ and we say that $\mathcal{A}$ *recognizes* the language $\mathcal{L}(\mathcal{A})$. For more details on automata theory and languages we refer to Hopcroft and Ullman [HU79].

Problem statements

In the following paragraph, we define the main decision problems that we discuss in the remainder of this section (specifically, in Section 3.2.2 and Section 3.2.3, respectively). First, we state the decision problem related to cherry-picking sequences.

CPS-Existence [DL18, p. 38]
Instance. A collection $\mathcal{P}$ of rooted binary phylogenetic X-trees.
Question. Does there exist a cherry-picking sequence for $\mathcal{P}$?

Furthermore, we are interested in the following three decision problems related to the display sets of two phylogenetic networks $\mathcal{N}$ and $\mathcal{N}'$.

Common-Tree-Containment [DLS19, p. 132]
Instance. Two phylogenetic networks $\mathcal{N}$ and $\mathcal{N}'$ on X.
Question. Is $T(\mathcal{N}) \cap T(\mathcal{N}') \neq \emptyset$?
(In words: Is there a phylogenetic X-tree displayed by both $\mathcal{N}$ and $\mathcal{N}'$?)

Display-Set-Containment [DLS19, p. 132]
Instance. Two phylogenetic networks $\mathcal{N}$ and $\mathcal{N}'$ on X.
Question. Is $T(\mathcal{N}) \subseteq T(\mathcal{N}')$?
(In words: Is each phylogenetic X-tree displayed by $\mathcal{N}$ also displayed by $\mathcal{N}'$?)

Display-Set-Equivalence [DLS19, p. 132]
Instance. Two phylogenetic networks $\mathcal{N}$ and $\mathcal{N}'$ on X.
Question. Is $T(\mathcal{N}) = T(\mathcal{N}')$?
(In words: Do $\mathcal{N}$ and $\mathcal{N}'$ display the same set of phylogenetic X-trees?)

As already briefly mentioned in the beginning of Section 3.2, the decision problems Common-Tree-Containment and Display-Set-Containment are generalizations of Tree Containment. To this end, consider Common-Tree-Containment if $\mathcal{N}' = \mathcal{T}$ where $\mathcal{T}$ is a phylogenetic X-tree (resp. $\mathcal{N} = \mathcal{T}$ for Display-Set-Containment).

3.2.2 Deciding the existence of a cherry-picking sequence

Let $\mathcal{P}$ be a collection of rooted binary phylogenetic X-trees. In this section, we present our results [DL18, DvIKL19] regarding the computational complexity of CPS-Existence.

First, we sketch a proof of hardness if the number of phylogenetic trees in $\mathcal{P}$ is not bounded as this proof already contains some of the key ingredients used in establishing our result [DL18, Thm. 4.2] for $|\mathcal{P}| = 8$ trees while being less technical. To this end, we obtain NP-hardness of CPS-Existence by a reduction[3] from the following problem.

Betweenness
Instance. A finite set A and a collection C of triples of distinct elements in A.
Question. Does there exist a total linear ordering on the elements in A such that $a_i < a_j < a_k$ or $a_k < a_j < a_i$ for each $(a_i, a_j, a_k) \in C$?

[3] This reduction was not previously published as we were quickly able to show hardness if the number of phylogenetic trees is constant.

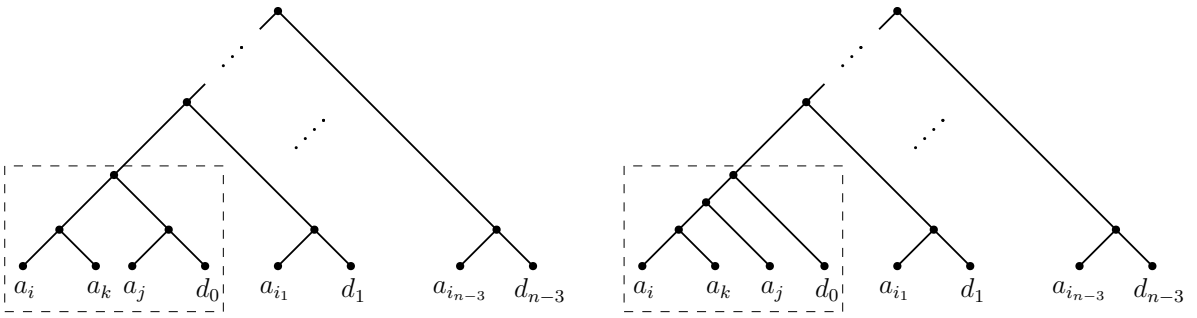


Figure 3.2: The two phylogenetic trees constructed for each triple (a_i, a_j, a_k) of an instance of the Betweenness problem. The two subtrees that are enforcing the betweenness constraint are marked with dashed rectangles. The elements with indices i_j, $1 \leq j \leq n-3$, are the elements not appearing in the triple.

Betweenness was shown to be NP-complete by Opatrny [Opa79]. Intuitively speaking, we are looking for an ordering of the elements in A such that a number of constraints, each given by a triple, are satisfied. For each triple (a_i, a_j, a_k), a solution, i.e., an ordering of the elements in A, must place a_j *between* a_i and a_k. Now, let us consider an instance of this problem with $A = \{a_1, \ldots, a_n\}$ and a collection $C \subseteq A \times A \times A$ of triples. Then, we construct the two phylogenetic trees shown in Figure 3.2 for each $(a_i, a_j, a_k) \in C$ (except for the part in the dashed rectangle all constructed trees are identical), where $D = \{d_0, \ldots, d_{n-3}\}$ are dummy variables. Thus, we construct $2 \cdot |C|$ phylogenetic trees in total. Let us consider the two phylogenetic trees—in particular the parts shown in the dashed rectangle—constructed for a triple (a_i, a_j, a_k). Observe that $\{a_i, a_k\}$ is a cherry in both trees, but a_j is not in a cherry in the right tree in Figure 3.2. Hence, either a_i or a_k precedes a_j in any cherry-picking sequence for these phylogenetic trees. Moreover, the right phylogenetic tree enforces that at least two of a_i, a_j, a_k must be placed before d_0 and, by the left phylogenetic tree, the second element in this subsequence must be a_j. Therefore, a cherry-picking sequence for all constructed trees yields a solution to the instance of Betweenness (by ignoring the dummy variables or, more precisely, taking the subsequence of the elements in A). Since the two phylogenetic trees constructed for a triple (a_i, a_j, a_k) do not restrict the placement of the elements in $A \setminus \{a_i, a_j, a_k\}$ (these elements are each in a cherry with some dummy variable), it is straightforward to construct a cherry-picking sequence given a solution to the instance of Betweenness. Noting that the described transformation is polynomial and a cherry-picking sequence can be verified in polynomial time, we conclude that CPS-Existence is NP-complete. Let us now come back to the results established in our papers [DL18, DvIKL19] that show hardness even if the number of phylogenetics trees is bounded by a constant.

Hardness for eight trees

Knowing that CPS-Existence is NP-complete in general, the next obvious question is whether or not the problem remains hard for a constant number of phylogenetic trees. We started with the idea that we may be able to represent multiple triples in the same two phylogenetic trees if the triples are pairwise disjoint (i.e., each of the two phylogenetic trees would have additional subtrees as the ones shown in the dashed rectangle in Figure 3.2). Hence, to consider this approach, we need an NP-complete decision problem similar to Betweenness such that the set of constraints (e.g., represented by triples) can be partitioned into a constant number of subsets such that elements of each subset are pairwise disjoint. To this end, we introduced the following restricted variant of the Intermezzo problem (the general version was introduced and shown to be NP-complete by Guttmann and Maucher [GM06]).

N-Disjoint-Intermezzo [DL18, p. 39]
Instance. A finite set A, collections $B_1, B_2, \ldots, B_N$ of pairs consisting of distinct elements from A, and collections $C_1, C_2, \ldots, C_N$ of triples consisting of distinct elements from A such that, for each $\ell \in \{1, 2, \ldots, N\}$, the elements in $B_\ell \cup C_\ell$ are pairwise disjoint.
Question. Does there exist a total linear ordering on the elements in A such that

$$a_i < a_j \text{ for each } (a_i, a_j) \in \bigcup_{1 \leq \ell \leq N} B_\ell,$$

and

$$a_i < a_j < a_k \text{ or } a_j < a_k < a_i \text{ for each } (a_i, a_j, a_k) \in \bigcup_{1 \leq \ell \leq N} C_\ell?$$

The proof of Guttmann and Maucher [GM06, Lem. 1] yields the following result if a restricted variant of 3-SAT is used in the reduction.

Theorem 1 ([DL18, Thm. 3.1])**.** 4-Disjoint-Intermezzo *is* NP*-complete.*

Reducing from 4-Disjoint-Intermezzo, we obtained the following result. The main idea is to enforce the constraints similar to the approach taken for the proof of the general case—the subtrees are a bit more involved for the Intermezzo constraints—and to represent all elements in $B_\ell \cup C_\ell$ (as defined in the above problem definition) within the same pair of phylogenetic trees.

Theorem 2 ([DL18, Thm. 4.2]). *Let $\mathcal{P} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_m\}$ be a collection of rooted binary phylogenetic X-trees.* CPS-Existence *is* NP*-complete for $m = 8$.*

In an instance of 4-Disjoint-Intermezzo, we can split sets of constraints which contain at least two elements (i.e., $B_\ell \cup C_\ell$ for all ℓ with $|B_\ell \cup C_\ell| \geq 2$) into multiple non-empty subsets. Using this, it is straightforward to show NP-completeness of N-Disjoint-Intermezzo for each fixed $N \geq 4$. Moreover, we can increase the number of phylogenetic trees in the reduction from N-Disjoint-Intermezzo to CPS-Existence by one if we introduce a phylogenetic tree corresponding to an empty set of constraints, say $B_{N+1} \cup C_{N+1} = \emptyset$ (in this case the two phylogenetic trees constructed similar to the ones shown in Figure 3.2 are identical), and obtain the following corollary.

Corollary 1 ([DL18, Cor. 4.3]). *Let $\mathcal{P} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_m\}$ be a collection of rooted binary phylogenetic X-trees.* CPS-Existence *is* NP*-complete for any fixed m with $m \geq 8$.*

Hardness for two trees

In the above presented results, we established that CPS-Existence is hard for a constant number of trees. Using the strategy described for $m = 8$ phylogenetic trees, the question of Humphries et al. [HLS13a] regarding the complexity for $m = 2$ seemed out of reach since a straightforward adaption of the reduction would have required NP-hardness of 1-Disjoint-Intermezzo (this variant can be solved in polynomial time though, since all constraints are disjoint and, thus, can always be simultaneously satisfied). However, by a rather complex reduction from 3-SAT, we obtained the following theorem.

Theorem 3 ([DvIKL19, Thm. 1]). *Let $\mathcal{P} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_m\}$ be a collection of rooted binary phylogenetic X-trees.* CPS-Existence *is* NP*-complete for $m = 2$.*

The reduction and the involved gadgets used in the proof of Theorem 3 are more complex compared to the sketched reduction from Betweenness for the general case (for an unbounded number of phylogenetic trees). The following two lemmas concern structural properties of cherry-picking sequences that turned out to be very useful in the proof of Theorem 3. The first lemma makes it possible to consider cherry-picking sequences for a collection $\mathcal{P}$ of phylogenetic trees which are defined on different sets X_i, $1 \leq i \leq |\mathcal{P}|$, instead of requiring that they are all phylogenetic X-trees on the same set X. Intuitively, we want that a phylogenetic X_i-tree $\mathcal{T}_i$ does not constrain the ordering of any element $x_j \notin X_i$. To this end, we simply define x_j to be in a cherry of each

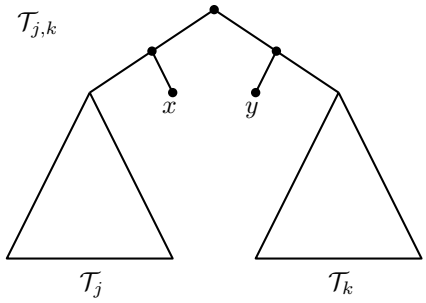


Figure 3.3: The compound tree $\mathcal{T}_{j,k}$ of $\mathcal{T}_j$ and $\mathcal{T}_k$ on disjoint sets X_j and X_k, respectively. We omitted some details from the original figure in Döcker et al. [DvIKL19, Fig. 3]. The two leaves x and y are introduced to ensure that the last appearance of an element from X_j (resp. X_k) ends up in a cherry of the tree obtained by reducing the compound tree according to the preceding elements.

phylogenetic $\mathcal{T}_i$. Thus, we obtain a straightforward generalization of a cherry-picking sequence to phylogenetic trees which are not necessarily on the same set.

Lemma 1 ([DvIKL19, Lem. 1]). *Let $\mathcal{P} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_m\}$ be a collection of rooted binary phylogenetic trees such that, for each $i \in \{1, \ldots, m\}$, $\mathcal{T}_i$ is a X_i-tree (i.e., the trees in $\mathcal{P}$ are not necessarily on the same set of taxa). Then we can construct in polynomial time a collection $\mathcal{P}'$ of m rooted binary phylogenetic trees all on the same set of taxa, such that $\mathcal{P}$ has a cherry-picking sequence if and only if $\mathcal{P}'$ does.*

Note that Lemma 1 could also be used to simplify our reduction from Betweenness as the two phylogenetic trees shown in Figure 3.2 can be replaced by the two subtrees shown in the dashed rectangles. Hence, the lemma improves readability since it enables us to focus on the crucial parts of the constructions. Now, the following lemma formalizes the idea that two phylogenetic trees $\mathcal{T}_j$ and $\mathcal{T}_k$ on disjoint sets can be represented in a single phylogenetic tree called the *compound tree* of $\mathcal{T}_j$ and $\mathcal{T}_k$ (see Figure 3.3). For a formal definition of the compound tree, we refer to Döcker et al. [DvIKL19]. Intuitively, we merge several phylogenetic trees on disjoint sets into one larger phylogenetic tree while preserving the existence and non-existence of a cherry-picking sequence. A similar idea was also used in our previous approach for $m = 8$ phylogenetic trees.

Lemma 2 ([DvIKL19, Lem. 2]). *Let $\mathcal{P} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_m\}$ be a collection of rooted binary phylogenetic trees such that, for each $i \in \{1, \ldots, m\}$, $\mathcal{T}_i$ is a X_i-tree, and let $\mathcal{T}_j$ and $\mathcal{T}_k$ be two trees in $\mathcal{P}$ such that $X_j \cap X_k = \emptyset$. Let $\mathcal{T}_{j,k}$ be the compound tree of $\mathcal{T}_j$ and $\mathcal{T}_k$. Then $\mathcal{P}$ has a cherry-picking sequence if and only if $(\mathcal{P} \setminus \{\mathcal{T}_j, \mathcal{T}_k\}) \cup \{\mathcal{T}_{j,k}\}$ has a cherry-picking sequence.*

An interesting application of Lemma 1 that is not mentioned in Döcker et al. [DvIKL19] is the following: Given a collection $\mathcal{P}$ of m phylogenetic X-trees, we can add any phylogenetic X'-tree with $X \cap X' = \emptyset$ to $\mathcal{P}$ and apply Lemma 1 to construct a set $\mathcal{P}'$ of $m+1$ phylogenetic X''-trees that has a cherry-picking sequence if and only if the original set $\mathcal{P}$ of m phylogenetic X-trees has a cherry-picking sequence. Hence, we obtain the following corollary which closes the gap between Theorem 3 and Corollary 1.

Corollary 2. *Let $\mathcal{P} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_m\}$ be a collection of rooted binary phylogenetic X-trees.* CPS-Existence *is* NP*-complete for any fixed m with $m \geq 2$.*

Bounding the number of cherries

Faced with the hardness of CPS-Existence for a constant number of phylogenetic trees—as we just alluded to, it is NP-complete even for two phylogenetic trees—a natural follow-up question is whether there are topological properties of the input trees that alleviate the computational complexity of the problem. Since cherries play an important role in the problem, bounding the number of cherries in each phylogenetic tree seems like the obvious approach. Indeed, the main result [DL18, Thm. 5.1] presented in this section states that CPS-Existence is solvable in polynomial time if both the number of phylogenetic trees in the input $\mathcal{P}$ and the number of cherries in each phylogenetic tree $\mathcal{T} \in \mathcal{P}$ are bounded by, not necessarily the same, constants.

We need some more notation: A *cherry-picked tree* of a phylogenetic X-tree $\mathcal{T}$ is a phylogenetic X'-tree $\mathcal{T}'$ with $X' \subset X$ that can be obtained from $\mathcal{T}$ by repeatedly deleting a leaf in a cherry and suppressing vertices of in-degree 1 and out-degree 1 (intuitively, $\mathcal{T}'$ is a phylogenetic tree that can occur as an intermediate step when reducing $\mathcal{T}$ according to a cherry-picking sequence). Now, $\mathcal{C}(\mathcal{T})$ is defined as the set containing all cherry-picked trees of $\mathcal{T}$ (including $\mathcal{T}$ itself and the completely reduced tree with no vertices). For instance, each phylogenetic tree in the first column of Figure 3.1 is a cherry-picked tree of the phylogenetic tree shown in the top left corner of the figure. For the rigorous definition of $\mathcal{C}(\mathcal{T})$, see our paper [DL18, Sec. 5] in the appendix. Furthermore, we denote the number of cherries in a phylogenetic tree $\mathcal{T}$ by $c_{\mathcal{T}}$. For a phylogenetic X-tree, we consider the language

$$\mathcal{L}_X(\mathcal{T}) = \{x_1 x_2 \ldots x_{|X|} \mid (x_1, x_2, \ldots, x_{|X|}) \text{ is a cherry-picking sequence for } \mathcal{T}\}.$$

Note that $\mathcal{L}_X(\mathcal{T})$ is not empty since each phylogenetic X-tree $\mathcal{T}$ has a cherry-picking sequence (in particular, $\mathcal{T}$ has a cherry if $|X| \geq 2$). Further, unless P = NP, it follows

that it is not possible to construct a deterministic finite automaton (DFA) in time bounded by a polynomial in $|X|$ that recognizes $\mathcal{L}_X(\mathcal{T})$ in general (i.e., if $c_\mathcal{T}$ is not constant). Suppose that we can construct such a DFA in polynomial time. Then, a cherry-picking sequence for two phylogenetic X-trees $\mathcal{T}_1, \mathcal{T}_2$ exist if and only if

$$\mathcal{L}_X(\mathcal{T}_1) \cap \mathcal{L}_X(\mathcal{T}_2) \neq \emptyset.$$

Let $\mathcal{A}_1$, $\mathcal{A}_2$ be two DFAs that recognize $\mathcal{L}_X(\mathcal{T}_1)$ and $\mathcal{L}_X(\mathcal{T}_2)$, respectively. Then, checking for non-emptiness of $\mathcal{L}_X(\mathcal{T}_1) \cap \mathcal{L}_X(\mathcal{T}_2)$ can be done in polynomial time using the well-known construction of a product automaton [Koz97] which also works for more than two automata (see Döcker and Linz [DL18] for more details).

In order to make use of the presented ideas, we need a way to construct an automaton that recognizes $\mathcal{L}_X(\mathcal{T})$ for a phylogenetic X-tree $\mathcal{T}$. A useful result [DL18, Lem. 5.5] which we obtained with the help of a perhaps surprisingly complex representation of $\mathcal{T}$ is that $|\mathcal{C}(\mathcal{T})| \leq (|X|+1)^{4c_\mathcal{T}-2}$. In particular, if $c_\mathcal{T}$ is constant, the number of cherry-picked trees of $\mathcal{T}$ is polynomial in the number of leaves $|X|$. Hence, in this case, it is computationally feasible to represent $\mathcal{T}$ and each cherry-picked tree of $\mathcal{T}$ by a state in a DFA $\mathcal{A}_\mathcal{T}$. Then, we introduce a transition $\delta(q, a) = q'$ if a is a leaf in a cherry of the phylogenetic tree, say $\mathcal{T}_q$, represented by the state q and the phylogenetic tree represented by q' can be obtained from $\mathcal{T}_q$ by deleting the leaf a (and suppressing the resulting vertex of in-degree 1 and out-degree 1 if $\mathcal{T}_q$ contains at least three leaves). Otherwise, if a is not in a cherry of $\mathcal{T}_q$, the automaton $\mathcal{A}_\mathcal{T}$ transitions into a special state q_e where $\mathcal{A}_\mathcal{T}$ stays forever (i.e., $\delta(q, a) = q_e$ and $\delta(q_e, a') = q_e$ for all $a' \in X$). The initial state of $\mathcal{A}_\mathcal{T}$ is the one corresponding to $\mathcal{T}$ and we mark the state corresponding to the empty phylogenetic tree (i.e., without any vertices) as the final state. For the technical details and a proof of the following lemma, we refer to our paper [DL18].

Lemma 3 ([DL18, Lem. 5.6])**.** *Let $\mathcal{T}$ be a rooted binary phylogenetic X-tree. There is a deterministic finite automaton $\mathcal{A}_\mathcal{T}$ with $O(|X|^{4c_\mathcal{T}-2})$ states that recognizes the language*

$$\mathcal{L}_X(\mathcal{T}) = \{x_1 x_2 \dots x_{|X|} \mid (x_1, x_2, \dots, x_{|X|}) \text{ is a cherry-picking sequence for } \mathcal{T}\}.$$

Moreover, the automaton $\mathcal{A}_\mathcal{T}$ can be constructed in time $f(|X|, c_\mathcal{T}) \in |X|^{O(c_\mathcal{T})}$.

By observing the equivalence of solving CPS-Existence for phylogenetic X-trees $\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_m$ and deciding whether $\bigcap_{1 \leq i \leq m} \mathcal{L}_X(\mathcal{T}_i) \neq \emptyset$, and using the product automaton construction for the DFAs $\mathcal{A}_{\mathcal{T}_i}$, $1 \leq i \leq m$, we obtained the following theorem.

Theorem 4 ([DL18, Thm. 5.1]). *Let $\mathcal{P} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_m\}$ be a collection of rooted binary phylogenetic X-trees. Let c be a maximum element in $\{c_{\mathcal{T}_1}, c_{\mathcal{T}_2}, \ldots, c_{\mathcal{T}_m}\}$. Then solving* CPS-Existence *for $\mathcal{P}$ takes time*

$$O\left(|X|^{m(4c-2)+1} + \sum_{i=1}^{m} f_i(|X|, c_{\mathcal{T}_i})\right),$$

where $f_i(|X|, c_{\mathcal{T}_i}) \in |X|^{O(c_{\mathcal{T}_i})}$. In particular, the running time is polynomial in $|X|$ if c and m are constant.

It would be interesting to know whether solving CPS-Existence for phylogenetic X-trees $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_m$ can be done in polynomial time if the maximum element $c \in \{c_{\mathcal{T}_1}, c_{\mathcal{T}_2}, \ldots, c_{\mathcal{T}_m}\}$ is constant but the number m of phylogenetic trees in the input is not constant (e.g., if m is allowed to depend on $|X|$). We leave the analysis of this case as an open question for future work. Note that, by Theorem 3, the reverse case—i.e., m is constant and c is unbounded—is NP-complete even for $m = 2$.

3.2.3 Displaying trees across two phylogenetic networks

Let $\mathcal{N}$ and $\mathcal{N}'$ be phylogenetic networks on a set X. In this section, we present our results for several questions related to the display sets of $\mathcal{N}$ and $\mathcal{N}'$. First, we consider Common-Tree-Containment where the task is to check whether the two networks display a common tree (i.e., we need to decide whether $T(\mathcal{N}) \cap T(\mathcal{N}') \neq \emptyset$).

Hardness of Common-Tree-Containment

We obtained the following theorem by a reduction from 3-SAT.

Theorem 5 ([DLS19, Thm. 3.2]). Common-Tree-Containment *is* NP*-complete when the input consists of two temporal normal networks.*

The following corollary is an immediate consequence of Theorem 5.

Corollary 3 ([DLS19, Cor. 3.3]). *Let $\mathcal{N}$ and $\mathcal{N}'$ be two temporal normal networks on X. It is* co-NP*-complete to decide if $T(\mathcal{N}) \cap T(\mathcal{N}') = \emptyset$.*

Recall that a normal network is a tree-child network without shortcuts. Hence, Common-Tree-Containment remains NP-hard for a strict subclass of phylogenetic networks for which we can decide Tree Containment in polynomial time. In the proof of Theorem 5, we showed that the problem is in NP using the polynomial algorithm of van Iersel et al. [vISS10] solving Tree Containment for tree-child networks (i.e., guess a phylogenetic tree and verify in polynomial time whether it is contained in both display sets). Now, since Tree Containment is NP-hard in general [KNTX08], we cannot automatically place Common-Tree-Containment in NP for general networks (indeed, the other problems we consider later with respect to the display set turn out to be complete for Π_2^P and, thus, are unlikely to be contained in NP). By using a different certificate, we obtained the following proposition showing that Common-Tree-Containment remains in NP for two unrestricted phylogenetic networks.

Proposition 1 ([DLS19, Cor. 5.3]). Common-Tree-Containment *is* NP*-complete for two arbitrary phylogenetic networks.*

Computational complexity of Display-Set-Equivalence

The main result in this section is that it is Π_2^P-complete to decide whether $T(\mathcal{N}) = T(\mathcal{N}')$ for two phylogenetic networks $\mathcal{N}$ and $\mathcal{N}'$. We obtained this result by a polynomial reduction from Display-Set-Containment (i.e., the question whether $T(\mathcal{N}) \subseteq T(\mathcal{N}')$). For this sake, we first needed to show Π_2^P-completeness of Display-Set-Containment. Since this problem is a generalized variant of Tree Containment, we already knew that it is NP-hard at the least. Kanj et al. [KNTX08, Thm. 3.1] showed NP-hardness of Tree Containment by a reduction from the problem of finding vertex-disjoint paths between given pairs of vertices of an acyclic digraph (this problem is called the Node-disjoint Paths problem in their article). Another name used for the latter problem—where the graph is not necessarily directed and acyclic—is Disjoint Connecting Paths (see Garey and Johnson [GJ79, p. 217]). Hence, starting the chain of reductions leading to Π_2^P-completeness of Display-Set-Containment and finally Display-Set-Equivalence with a Π_2^P variant of Disjoint Connecting Paths seemed like a promising approach. To this end, we introduced the following decision problem where π_i denotes a directed path from s_i to t_i for a pair of vertices (s_i, t_i) in an acyclic digraph.

∀∃ Directed-Disjoint-Connecting-Paths [DLS19, p. 136]
Instance. A directed graph G and two collections

$$P^{\forall} = \{(s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p)\},$$
$$P^{\exists} = \{(s_{p+1}, t_{p+1}), (s_{p+2}, t_{p+2}), \ldots, (s_k, t_k)\}$$

of disjoint pairs of vertices in G such that $1 \leq p < k$ and, for each $(s_i, t_i) \in P^{\forall}$, there exists a directed path from s_i to t_i in G.
Question. For each set $\Pi^{\forall} = \{\pi_1, \pi_2, \ldots, \pi_p\}$ of directed paths, does there exist a set $\Pi^{\forall} \cup \{\pi_{p+1}, \pi_{p+2}, \ldots, \pi_k\}$ of mutually vertex-disjoint directed paths in G?

By a reduction from ∀∃ 3-SAT, we obtained the following result regarding a restricted variant of the above problem, where the input graph is a particular phylogenetic network (the structure of an instance of this restricted variant is shown in Figure 3.4).

Theorem 6 ([DLS19, Thm. 4.1]). *The decision problem* ∀∃ Phylo-Directed-Disjoint-Connecting-Paths *is* Π_2^P*-complete.*

As a consequence of Theorem 6, we obtained the following corollary.

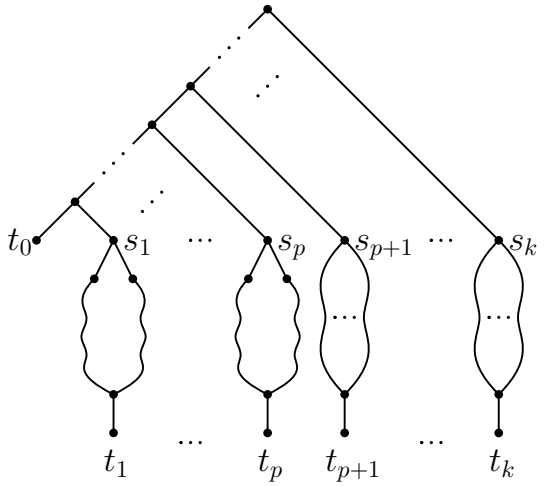


Figure 3.4: Structure of an instance of ∀∃ Phylo-Directed-Disjoint-Connecting-Paths. Each squiggly line is a directed path. For each $i \in \{1, \ldots, p\}$, there are precisely two paths from s_i to t_i which only have the first vertex (i.e., s_i) and the last two vertices (t_i and its unique parent) in common (moreover, for $j, k \in \{1, \ldots, p\}$ with $j \neq k$, no path from s_j to t_j intersects with a path from s_k to t_k). We called this property the *two-path property relative to* p in Döcker et al. [DLS19, Sec. 4.1]. Depending on the given instance, these paths may intersect otherwise (e.g., a path from s_1 to t_1 and a path from s_{p+1} to t_{p+1}) and for $i > p$, there may be more than two paths from s_i to t_i.

Corollary 4 ([DLS19, Cor. 4.2]). *The decision problem* ∀∃ Directed-Disjoint-Connecting-Paths *is* Π_2^P*-complete.*

By a reduction from ∀∃ Phylo-Directed-Disjoint-Connecting-Paths to Display-Set-Containment, we obtained the following theorem.

Theorem 7 ([DLS19, Thm. 4.3]). Display-Set-Containment *is* Π_2^P*-complete.*

In the proof of Theorem 7, we constructed two networks $\mathcal{N}_1$ and $\mathcal{N}_2$ (depicted in Figure 3.5) for which $T(\mathcal{N}_1) \subseteq T(\mathcal{N}_2)$ if and only if the given instance, say $\mathcal{I}$, of ∀∃ Phylo-Directed-Disjoint-Connecting-Paths is a yes-instance. Let G be the graph corresponding to the instance $\mathcal{I}$. Then, the network $\mathcal{N}_2$ is obtained from G by, for each $i \in \{1, \ldots, p\}$, subdividing both of the two outgoing arcs of s_i with two new vertices and attaching a new leaf (t_i' and t_i'', respectively) to each of the introduced vertices (compare the generic instance shown in Figure 3.4 and the network $\mathcal{N}_2$ in Figure 3.5). Now, each phylogenetic tree in the display set of $\mathcal{N}_1$ encodes a selection of paths connecting the pairs in $P^\forall = \{(s_1, t_2), \ldots, (s_p, t_p)\}$. By the two-path property (alluded to in the caption of Figure 3.4) there are precisely two paths from s_i to t_i for each $(s_i, t_i) \in P^\forall$. Let $(s_i, t_i) \in P^\forall$ be a pair of vertices and let π_i, π_i' be the two directed paths from s_i to t_i.

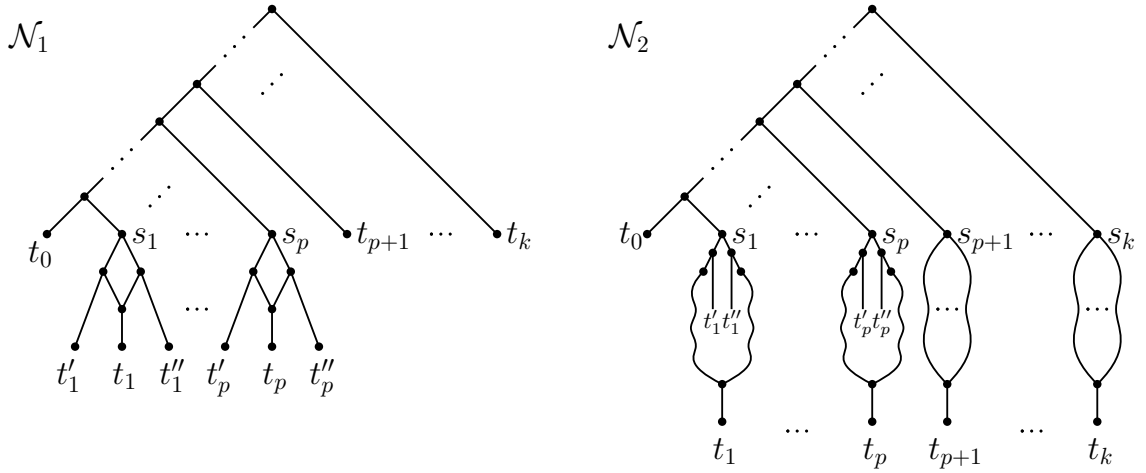


Figure 3.5: Networks constructed in the reduction from ∀∃ Phylo-Directed-Disjoint-Connecting-Paths to Display-Set-Containment (some details of our original figure [DLS19, Fig. 6] were omitted or revised to keep the presentation consistent with the rest of this thesis, respectively). Each squiggly line is a directed path. Depending on the given instance, these paths may intersect and for $i > p$, there may be more than two paths from s_i to t_i.

Intuitively, for $i \in \{1, \ldots, p\}$, a phylogenetic tree $\mathcal{T} \in T(\mathcal{N}_1)$ encodes the selection of π_i if $\{t_i, t'_1\}$ is a cherry of $\mathcal{T}$ and the selection of π'_i if $\{t_i, t''_1\}$ is a cherry of $\mathcal{T}$, respectively. Shifting the attention back to $\mathcal{N}_2$, we can embed $\mathcal{T}$ in $\mathcal{N}_2$ if and only if there is a selection of mutually vertex-disjoint paths in G connecting all pairs (s_i, t_i) of the instance and using the paths encoded by $\mathcal{T}$ for $i \leq p$. In Figures 3.6 and 3.7, we illustrate the direction for which the path problem is a yes-instance (see Döcker et al. [DLS19] for a detailed proof of both directions).

By a reduction from Display-Set-Containment, we obtained the following theorem (for the proof involving rather complex constructions, we refer to our article [DLS19, Thm. 4.5]).

Theorem 8 ([DLS19, Thm. 4.5])**.** Display-Set-Equivalence *is* Π_2^P*-complete.*

In contrast, if an instance of Display-Set-Equivalence consists of a normal and a tree-child network, we obtained the following result by comparing local structures and successively reducing the two networks in a way that preserves the answer to the decision problem.

Theorem 9 ([DLS21, Thm. 1.1])**.** *Let $\mathcal{N}$ and $\mathcal{N}'$ be normal and tree-child networks on X, respectively. Then deciding if $T(\mathcal{N}) = T(\mathcal{N}')$ can be done in time quadratic in the size of X.*

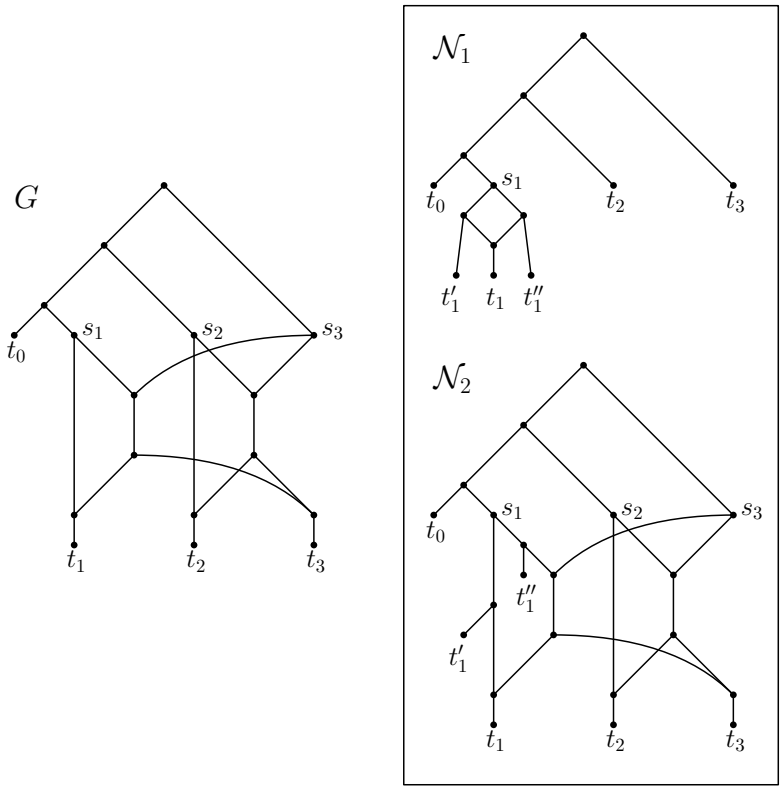


Figure 3.6: Example showing the transformation of the instance of ∀∃ Phylo-Directed-Disjoint-Connecting-Paths defined by the graph G depicted on the left, $P^{\forall} = \{(s_1, t_1)\}$ and $P^{\exists} = \{(s_2, t_2), (s_3, t_3)\}$ into the instance of Display-Set-Containment depicted on the right.

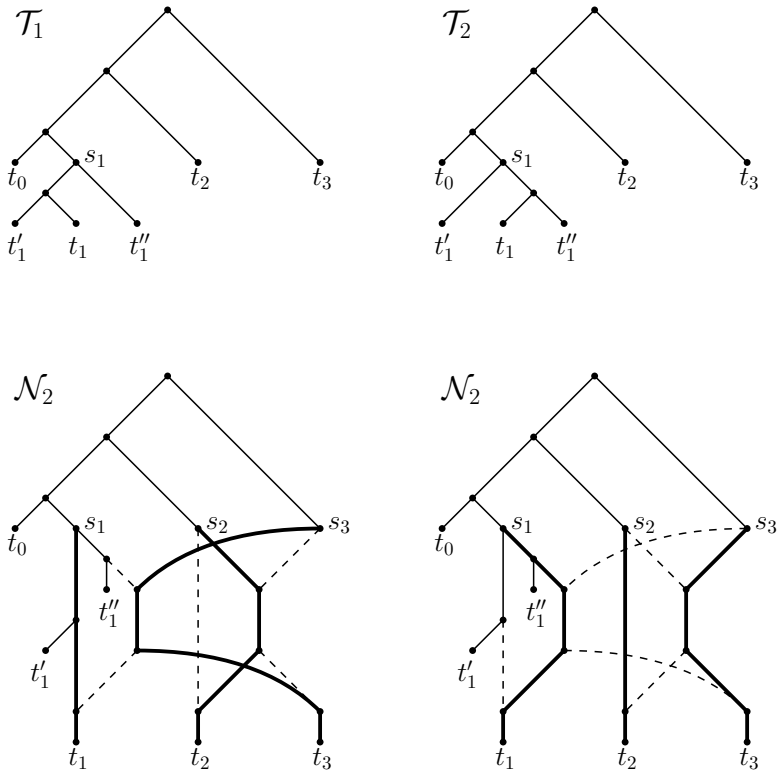


Figure 3.7: Continuation of the example shown in Figure 3.6. The display set of $\mathcal{N}_1$ contains precisely the two phylogenetic trees $\mathcal{T}_1$ and $\mathcal{T}_2$ depicted above. In the lower part of the figure, we illustrated how these trees can be embedded into $\mathcal{N}_2$ using a selection of mutually vertex-disjoint paths connecting each pair in $\{(s_1, t_1), (s_2, t_2), (s_3, t_3)\}$ (the selection is indicated by bold arcs). Note that deleting the dashed arcs and suppressing vertices of in-degree 1 and out-degree 1 yields $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively.

Observe that there is a gap between the results obtained in Theorem 8 and Theorem 9 leaving plenty of room for future research regarding the computational complexity of Display-Set-Equivalence. In particular, the question of Gunawan et al. [GDZ17] for two reticulation-visible networks—which we alluded to in the beginning of Section 3.2—remains open. Our results make progress towards an answer to their question as follows.

First, Theorem 9 shows that Display-Set-Equivalence can be solved in polynomial time if one network is normal and the other one is tree-child, respectively. Since both of these classes are well-known to be properly contained in the class of reticulation-visible networks, we provided a polynomial-time algorithm for a more restricted setting. An obvious follow-up question is whether the computational complexity of the problem changes if both networks are only required to be tree-child (cf. Döcker et al. [DLS21]).

Second, we showed that Display-Set-Equivalence is very hard, i.e., Π_2^P-complete, for the more general setting where the phylogenetic networks in the input are not required to be of a certain class. As noted in the conclusion in Döcker et al. [DLS19], it is unlikely[4] that Display-Set-Equivalence is Π_2^P-complete for two phylogenetic networks $\mathcal{N}$ and $\mathcal{N}'$ for which we can decide Tree Containment in polynomial time (deciding whether $T(\mathcal{N}) = T(\mathcal{N}')$ is in co-NP in this case). Hence, it would be interesting to know whether there are relevant network classes (e.g., reticulation-visible networks) for which Display-Set-Equivalence is co-NP-complete.

[4]The word unlikely is used a bit loosely here. To be precise, the result would lead to a collapse of the polynomial hierarchy to the first level. While this possibility cannot be completely discarded, it would be surprising in a similar way to P being equal to NP (though the latter has even stronger consequences, i.e., a complete collapse of the polynomial hierarchy).

3.3 Satisfiability

In this section, we give an overview of the main results we obtained for several variants of the satisfiability problem for Boolean formulas—most of these results are sharp with respect to the chosen problem parameters. In particular, our results place these variants in one of the complexity classes P, NP, co-NP or Π_2^P of the polynomial hierarchy (see Section 3.1 for a brief introduction into the polynomial hierarchy). Our results were already used to analyze the computational complexity of decision problems from different areas such as computational social choice (specifically, group activity selection [Dar18, Thm. 5]), graph theory (see, e.g., Chudnovsky et al. [CHR+20, Thm. 16] or Krysta et al. [KMZ20, p. 39]) and graph drawing (see Alam et al. [AKM17, Sec. 4]).

First, we consider Monotone 3-SAT for which hardness was shown by Gold [Gol78]. In this variant, each clause is required to be *monotone* (a clause is monotone if either all of the contained variables are negated or none of them are). Further, Monotone 3-SAT is also known to be NP-complete if each clause contains exactly three distinct literals [Li97] (if not explicitly mentioned otherwise, we consider this variant in the following). Our research interest for restricted variants of this problem was sparked by the conjecture that Monotone 3-SAT remains NP-hard if each variable appears at most five times which was stated, and attributed to Eisenstat, in the scribe notes [DKY14] of lecture held by Demaine at the Massachusetts Institute of Technology (Eisenstat was one of the teaching assistants for the corresponding course with the title "Algorithmic Lower Bounds: Fun with Hardness Proofs"). In fact, a stronger version of this conjecture is true: In a working paper (see Darmann and Döcker [DD16]), we proved that Monotone 3-SAT is NP-complete if each variable appears at most four times. For the published journal version [DDD18a], we combined these results with another working paper [DDD16], in which we obtained NP-hardness results for bounded (with respect to variable appearances) variants of Planar Monotone $(2,3)$-SAT, where each clause contains either two or three distinct literals and the associated bipartite incidence graph[5] is planar. We note that the unbounded version was already known to be NP-complete by a result from de Berg and Khosravi [dBK12]. In subsequent articles [DD21, Döc19], we refined our previous results for Monotone 3-SAT. For example, we showed that Monotone 3-SAT remains NP-complete if each variable appears exactly twice unnegated and exactly twice negated.

Second, we present our hardness result [DD20] for Not-All-Equal 3-SAT (NAE-3-

[5]Each variable and each clause is represented by a vertex and there is an edge between a variable vertex, say v, and a clause vertex, say c, if v appears in c (unnegated or negated).

SAT), where a satisfying truth assignment sets at least one literal in each clause to True and at least one to False, respectively. Specifically, we showed that the problem remains NP-hard if there are no negations present in the formula, each clause consists of exactly three distinct variables, each variable appears in exactly four clauses and each pair of distinct clauses shares at most one variable (Boolean formulas with the latter property are called linear). The novelty of our contribution in relation to a previous result from Porschen et al. [PSSW14, Thm. 3] for linear instances of NAE-3-SAT is the bound on the variable appearances.

Finally, we outline our results [DDLS20] for the Π_2^P-complete decision problems $\forall\exists$ 3-SAT and $\forall\exists$ NAE-3-SAT (see Section 3.1 for a formal description of these problems). Both of these problems are canonical problems for the complexity class Π_2^P and they are frequently used to obtain hardness results for this class (similar to the role of the unquantified counterparts for obtaining NP-hardness results). Hence, showing Π_2^P-completeness of $\forall\exists$ 3-SAT and $\forall\exists$ NAE-3-SAT in restricted settings is important as it helps to

- simplify the design of polynomial transformations that reduce from these problems,
- obtain stronger results since restrictions may carry over in some way (cf. Döcker et al. [DDLS20, p. 72]).

Note that showing NP-completeness of restricted variants of 3-SAT and NAE-3-SAT is helpful in the same way (in obtaining NP-hardness results). Furthermore, obtaining sharp results also leads to a better understanding of Π_2^P as we gain insights under which restrictions the canonical problems remain Π_2^P-hard and at what point the problem becomes easier (i.e., we obtain membership in P, NP or co-NP).

In this thesis, a result is considered sharp with respect to a restriction if a minimal deviation allows the problem to be placed on a lower level of the polynomial hierarchy. For instance, we showed that Monotone 3-SAT is NP-complete if each variable appears k times unnegated and k times negated for each fixed $k \geq 2$ (see Theorem 11). The restriction $k = 2$ is sharp, since all instances of Monotone 3-SAT where $k = 1$ are satisfiable by a well-known result from Tovey [Tov84, Thm. 2.4]. We say that a hardness result R is *stronger* than some other result R' if one the following conditions holds:

(1) both results are obtained for the same decision problem and R shows hardness of the problem for a complexity class that is on a higher level of the polynomial hierarchy, or

(2) both results obtain hardness for the same complexity class and the decision problem considered by R is a special case of the decision problem considered by R' (i.e., the instances of the former problem satisfy additional properties).

Throughout the thesis, we have the second condition in mind when speaking of "stronger results" unless it is explicitly mentioned otherwise. For instance, showing NP-hardness of Monotone 3-SAT if each variable appears *at most* four times is a stronger result than establishing this classification result if each variable appears *at most* five times.

3.3.1 Formal framework

Let $V = \{x_1, x_2, \ldots, x_n\}$ be a set of variables. A *literal* is an element of the set $\mathcal{L} = \{x_i, \overline{x_i} \mid 1 \leq i \leq n\}$, where $\overline{x_i}$ denotes the negation of the variable x_i. A *clause* c_j is a disjunction of literals, which we describe as a subset of $\mathcal{L}$. Another common way to describe clauses uses the symbol $\vee$ (read "or"). For instance, the clause $\{x_1, \overline{x_2}, x_3\}$ is written as $x_1 \vee \overline{x_2} \vee x_3$ in this notation. Further, a k-clause contains exactly k distinct literals. We say that a clause is *positive* (resp. *negative*) if the clause is a subset of V (resp. a subset of $\mathcal{L} \setminus V$); otherwise we say that the clause is *mixed*. A Boolean *formula* $\varphi = \bigcup_{j=1}^{m} \{c_j\}$ in *conjunctive normal form* (CNF) is a conjunction of clauses $c_1, \ldots, c_m$. If clauses are described using the symbol $\vee$, then we use the symbol $\wedge$ (read "and") to express a conjunction of clauses, i.e., a formula is written as $\varphi = \bigwedge_{j=1}^{m} c_j$. We say that a formula φ is *monotone*[6] if it does not contain a mixed clause, and *linear* if each pair of distinct clauses in φ shares at most one variable (e.g., the two clauses $\{x_1, \overline{x_2}, x_3\}$ and $\{x_2, x_4, x_5\}$ share the variable x_2). We are also interested in *planar* formulas, where the following associated bipartite graph (*incidence graph*) is required to be planar: Each variable and each clause is represented by a vertex which has the same label as its corresponding variable (e.g., the vertex introduced for variable x_1 has the label x_1) or clause, respectively, and we have an edge $\{x_i, c_j\}$ if and only if x_i or its negation is contained in c_j. A *truth assignment* $\beta \colon V \to \{T, F\}$ assigns a truth value to each variable, where T and F represent the truth values True and False, respectively. A truth assignment β *satisfies* a clause if at least one contained literal evaluates to T (e.g., a literal $\overline{x_i}$ evaluates to T if $\beta(x_i) = F$). Further, a truth assignment β *nae-satisfies* a clause if at least one contained literal evaluates to T and at least one evaluates to F (i.e., the truth values of the contained literals are "not all equal"). Now, a Boolean

[6] However, in the context of Not-All-Equal 3-SAT, a formula is called monotone if each clause is positive.

formula $\varphi = \bigcup_{j=1}^{m}\{c_j\}$ over a set of variables V is *satisfiable* (resp. *nae-satisfiable*) if there is a truth assignment $\beta\colon V \to \{T, F\}$ such that β satisfies (resp. nae-satisfies) each clause c_j, $1 \leq j \leq m$.

Problem statements

In the following, we formally state the considered variants of the Boolean satisfiability problem and provide notation to describe restrictions of these problems (similar to the notation used in Darmann et al. [DDD18a]). First, we present the definition of SAT which was the first problem shown to be NP-complete by Cook [Coo71].

SAT
Instance. A Boolean formula $\varphi = \bigcup_{j=1}^{m}\{c_j\}$ over a set V of variables, where $c_j = \{\ell_{j,1}, \ell_{j,2}, \ldots, \ell_{j,i_j}\}$ is a disjunction of literals formed over variables in V.
Question. Is there a truth assignment for V such that φ is satisfied?

The problem SAT is also well-known to be NP-complete if each clause contains exactly three distinct literals (see Garey and Johnson [GJ79, p. 48f] for a proof). The corresponding decision problem can be formally stated as follows.

3-SAT
Instance. A Boolean formula $\varphi = \bigcup_{j=1}^{m}\{c_j\}$ over a set V of variables, such that each clause $c_j \in \varphi$ contains $|c_j| = 3$ distinct literals formed over pairwise distinct variables.
Question. Is there a truth assignment for V such that φ is satisfied?

We also obtained some results for formulas where we allow variables and literals to appear multiple times in the same clause. Whenever a clause is not required to consist of *distinct* literals, we use multisets to represent clauses and denote the corresponding decision problems by SAT* or 3-SAT*. The variants of SAT considered in this work can be described by a prefix belonging to the set

$$\{\text{PLANAR}, \lambda\} \times \{\text{MONOTONE}, \lambda\} \times \{(2,3), 3\},$$

where λ denotes the empty word, and a suffix belonging to the set

$$\{*, \lambda\} \times \{k, \text{E}k \mid k \geq 1\} \cup \{(\leq p, \leq q),\ (p, q) \mid p \geq 1 \text{ and } q \geq 1\}.$$

Here, PLANAR and MONOTONE refer to the incidence graph being planar and each clause being monotone, respectively. Further, if $(2, 3)$ is part of the prefix, each clause contains

exactly two or exactly three literals (otherwise 3 is part of the prefix which means that each clause contains exactly three literals). The suffix k (resp. Ek) indicates that each variable appears at most (resp. exactly) k times. Finally, $(\leq p, \leq q)$ (resp. (p, q)) refine the just mentioned suffixes: Specifically, each variable appears at most (resp. exactly) p times unnegated and at most (resp. exactly) q times negated. For example, the decision problem Planar Monotone $(2,3)$-SAT-E3 concerns monotone formulas, where the incidence graph is planar, each clause contains exactly two or exactly three distinct literals and each variable appears exactly three times (if we use 3 instead of E3 as the suffix, then each variable appears *at most* three times). As another example consider Monotone 3-SAT-$(\leq 2, \leq 2)$, where the formula is monotone, each clause contains exactly three distinct literals formed over pairwise distinct variables and each variable appears at most twice unnegated and at most twice negated (for the suffix $(2,2)$, each variable appears *exactly* twice unnegated and *exactly* twice negated).

We now turn to the variant of 3-Sat that asks whether a given formula is nae-satisfiable.

Not-All-Equal 3-Sat (NAE-3-Sat)
Instance. A Boolean formula $\varphi = \bigcup_{j=1}^{m} \{c_j\}$ over a set V of variables, such that each clause $c_j \in \varphi$ contains $\lvert c_j \rvert = 3$ distinct literals formed over pairwise distinct variables.
Question. Is there a truth assignment for V such that φ is nae-satisfied?

Schaefer [Sch78] established NP-completeness of NAE-3-SAT*. Recall that the $*$ symbol refers to Boolean formulas where literals may appear more than once within the same clause. To be precise, Schaefer considered the restriction of NAE-SAT, where a clause contains *at most* three distinct literals (but simply duplicating literals within clauses having less than three literals yields an instance of NAE-3-SAT*). Moreover, NAE-3-SAT* remains NP-hard if the literals in each clause are distinct (see, e.g., the proof of Porschen et al. [PSSW14, Thm. 3] showing that NAE-3-SAT is NP-complete if the input formula is linear and all variable appearances are unnegated).

We consider variants of NAE-3-Sat that can be described by a prefix belonging to the set

$$\{\text{Linear}, \lambda\} \times \{\text{Monotone}, \lambda\},$$

and a suffix belonging to the set

$$\{k, \text{E}k \mid k \geq 1\}.$$

Here, LINEAR and MONOTONE refer to the formula being linear and each clause being positive[7], respectively. Further, the suffix k (resp. Ek) indicates that each variable appears at most (resp. exactly) k times. In particular, the decision problem LINEAR MONOTONE NAE-3-SAT-E4 concerns linear formulas, where negations are completely absent, each clause contains exactly three distinct variables and each variable appears in exactly four clauses.

3.3.2 Related work and a brief historical overview

As alluded to above, the Boolean satisfiability problem SAT was the first decision problem shown to be NP-complete (Cook [Coo71] established this result in 1971). Karp [Kar72] demonstrated in the following year how powerful the tool of polynomial-time reductions is by using this technique to show that many classical problems are NP-complete (among these problems are INTEGER PROGRAMMING, CLIQUE, CHROMATIC NUMBER, EXACT COVER, 3-DIMENSIONAL MATCHING and HAMILTONIAN CIRCUIT[8], just to name a few). Moreover, Karp introduced the notion of reducibility for such problems as it is used to this day in the theory of NP-completeness. For this reason, polynomial-time reductions are sometimes called *Karp reductions*. For other ways to define NP-hardness, e.g. by using Turing reductions, we refer the interested reader to the book by Garey and Johnson [GJ79].

Another milestone in the history of the Boolean satisfiability problem is the Dichotomy Theorem obtained by Schaefer [Sch78] in 1978 which considers generalizations of this problem described by finite sets of logical relations and places these problems in the complexity classes P or NP (i.e., in the latter case the problem is classified as NP-complete). As a consequence, many variants of SAT (e.g., NAE-3-SAT*) turn out to be NP-complete. Since the Dichotomy Theorem cannot be used to infer the computational complexity of SAT for certain types of constraints (e.g. bounded variable appearances), proving hardness by traditional means—specifically, polynomial-time reductions from known NP-complete problems—remains a necessity for such variants.

It is well-known that 3-SAT is NP-complete (a proof can be found in Garey and Johnson [GJ79, p. 48f]). Recall that each clause in a 3-SAT formula contains exactly three distinct literals. Adding to that, Tovey [Tov84, Thm. 2.3] showed that 3-SAT-4,

[7] Note that the prefix MONOTONE has a different meaning for 3-SAT and NAE-3-SAT, respectively.

[8] HAMILTONIAN CIRCUIT is closely related to the TRAVELING SALESPERSON problem mentioned in the introduction. NP-completeness of the latter problem can be obtained by a reduction from the former problem [GJ79, p. 211].

where each variable appears in at most four clauses, remains NP-complete. Further, he also provided a proof that this bound is optimal in the sense that each 3-SAT-3 formula is satisfiable [Tov84, Thm. 2.4]. Berman et al. [BKS03, Thm. 1] further refined the hardness result from Tovey by establishing NP-completeness of 3-SAT if each variable appears exactly twice unnegated and exactly twice negated. As alluded to in the second paragraph of Section 3.3, the decision problem 3-SAT is also known to be NP-complete if each clause is monotone [Gol78, Li97].

An interesting and useful variant of the Boolean satisfiability problem is concerned with formulas with a planar incidence graph. Lichtenstein [Lic82] showed that Planar 3-SAT* is NP-complete[9] and Mansfield [Man83] added to this result by establishing NP-completeness of Planar 3-SAT. Using the latter result as a starting point for a reduction, Kratochvíl [Kra94] proved that Planar 3-SAT-4, where each variable appears at most four times, is NP-complete even if the incidence graph is vertex 3-connected. De Berg and Khosravi [dBK12] showed that Planar Monotone 3-SAT* is NP-complete[10] as well. In contrast, Pilz [Pil19] proved that all instances of Planar Monotone 3-SAT are satisfiable, thereby answering our open question in Darmann et al. [DDD18a]. For a detailed survey of satisfiability problems with a planar incidence graph we refer to the master's thesis of Tippenhauer [Tip16].

We already mentioned NAE-3-SAT* in the context of the Dichotomy Theorem. In fact, this problem is also known to be NP-complete if the number of appearances per variable is bounded by 4 (a proof can be found in the extended arXiv version [KP17] of an article by Karpiński and Piecuch [KP18]). Furthermore, Porschen et al. [PSSW14, Thm. 3] showed that Linear Monotone NAE-3-SAT—the variant of NAE-3-SAT where each variable appearance is unnegated and each pair of distinct clauses shares at most one variable—is NP-complete. Interestingly, planarity of the incidence graph alleviates the complexity of NAE-3-SAT* as Moret [Mor88] proved that this variant is in P.

As alluded to in Section 3.1, each of the complexity classes Σ_k^P (resp. Π_k^P) of the polynomial hierarchy contain quantified variants of the Boolean satisfiability problem that are Σ_k^P-complete (resp. Π_k^P-complete). Schaefer and Umans [SU02] compiled a useful list containing many problems that are known to be complete for some class in

[9] In fact, Lichtenstein [Lic82] proved a stronger result where the variables are additionally linked by edges in the incidence graph (such that they form a cycle).

[10] Actually, de Berg and Khosravi [dBK12] obtained a stronger result where the variables are linked (as mentioned in the footnote regarding the result of Lichtenstein [Lic82]) and the corresponding graph admits a particular rectilinear drawing.

the polynomial hierarchy (with a focus on $k \in \{2, 3\}$).

As mentioned in Section 3.1, the decision problems $\forall\exists$ 3-SAT and $\forall\exists$ NAE-3-SAT are both Π_2^P-complete by results from Stockmeyer [Sto76] and Eiter and Gottlob [EG95], respectively. Gutner [Gut96] showed that the variant of $\forall\exists$ 3-SAT where each clause contains exactly three distinct variables and the incidence graph is planar, remains Π_2^P-complete. Furthermore, Haviv et al. [HRTS07] showed hardness of approximation for $\forall\exists$ 3-SAT with bounded variable appearances (even if each universal variable appears at most twice and each existential variable appears at most three times).

For an extensive survey of different variants of the Boolean satisfiability problem and a detailed list of the known complexity results, we refer to Filho [Fil19]. Further, we refer to the book by Schöning and Torán [ST13] for an introduction into algorithms that can be used to decide whether a given Boolean formula is satisfiable.

3.3.3 Monotone 3-SAT with bounded variable appearances

We now turn to our research inspired by a conjecture for Monotone 3-SAT with bounded variable appearances which was stated in an MIT lecture (see the beginning of Section 3.3 for more details). As already mentioned earlier, we not only confirmed that the conjecture is true, but also provided results for a family of related problems including stronger versions of the conjecture that sparked this research. To this end, we consider the computational complexity of Monotone 3-SAT if each variable has only a bounded number of appearances. More specifically, we consider bounds on the number of unnegated and on the number of negated appearances per variable which are not necessarily equal. An anonymous referee provided the following result (including a proof) which significantly improved the presentation in Darmann and Döcker [DD21]. In particular, the provided general method to increase the number of literal appearances, which the proof of the following theorem is based on, resulted in more concise and easier to read constructions.

Theorem 10 ([DD21, Thm. 1]). *Let p, q be fixed integers with $p \geq 1$ and $q \geq 1$. Then,* Monotone 3-Sat-$(\leq p, \leq q)$ *is NP-complete if and only if* Monotone 3-Sat-(p, q) *is NP-complete.*

Observe that Theorem 10 implies the following corollary.

Corollary 5 ([DD21, Cor. 1]). *Let r, s be fixed positive integers such that* Monotone 3-Sat-$(\leq r, \leq s)$ *is NP-complete. Then,* Monotone 3-Sat-(p, q) *is NP-complete for all pairs of fixed integers p, q with $p \geq r$ and $q \geq s$.*

The following theorem summarizes two results from Darmann and Döcker [DD21, Cor. 5 and Cor. 7] (here, we explicitly include the case $(1, t)$ which is symmetric to $(t, 1)$).

Theorem 11. Monotone 3-Sat-(p, q) *is NP-complete for each pair of fixed integers*

$$(p, q) \in \{(r, s) \mid r \geq 2, s \geq 2\} \cup \{(1, t), (t, 1) \mid t \geq 5\}.$$

Noting that Monotone 3-Sat-(p, q) is trivial if $p+q \leq 3$ by a result from Tovey [Tov84], the only remaining cases with unknown status are $(p, q) \in \{(1, t), (t, 1) \mid 3 \leq t \leq 4\}$. By symmetry, it is sufficient to determine the computational complexity of $(p, q) \in \{(3, 1), (4, 1)\}$ in order to close the gap towards a complete dichotomy. Using a probabilistic argument, one can show that each Monotone 3-Sat-(p, q) formula with less than 21 variables (resp. 27 variables) is satisfiable if $(p, q) = (4, 1)$ (resp. $(p, q) = (3, 1)$) [DD21]. Moreover, we were able to prove the following theorem which relaxes the constraint on the variables, i.e., we only require that each variable either satisfies the $(3, 1)$ or the $(1, 3)$ constraint (in other words, the variables can be partitioned into two sets V', V'' such that each variable in V' appears exactly three times unnegated and once negated and each variable in V'' appears exactly once unnegated and three times negated).

Theorem 12 ([DD21, Thm. 6])**.** Monotone 3-Sat-E4 *is NP-complete even if each variable appears three times unnegated and once negated or three times negated and once unnegated.*

We note that both Theorem 11 (specifically NP-completeness of Monotone 3-Sat-(2,2)) and Theorem 12 improve upon the result from Darman et al. [DDD18a, Cor. 4] that established NP-completeness of Monotone 3-Sat-E4. Furthermore, Theorem 12 implies NP-completeness of 3-Sat-(3,1) and, by symmetry, of 3-Sat-(1,3) which complements a result from Berman et al. [BKS03, Thm. 1] showing NP-completeness of 3-Sat-(2,2) (cf. Darmann and Döcker [DD21, p. 57]).

Towards settling the two remaining cases

In this section, we discuss a possible approach to tackle the question whether Monotone 3-Sat-(p, q) is NP-complete for $(p, q) \in \{(3, 1), (4, 1)\}$ (we presented this question as a challenge for future research in Darmann and Döcker [DD21]). By Theorem 10, we may consider the relaxed version Monotone 3-Sat-$(\leq p, \leq q)$ instead. For the

relaxed version, the existence of an unsatisfiable formula implies the existence of a *minimal* formula $\mathcal{U}_{\min}$ with this property (minimal in the sense that removing any clause yields a satisfiable formula). Hence, given a minimal unsatisfiable formula $\mathcal{U}_{\min}$, we can do the following: By removing any positive clause, say $c = \{x_i, x_j, x_k\}$, from $\mathcal{U}_{\min}$, we obtain three variables that have the forced truth value F if the clauses $\mathcal{U}_{\min} \setminus \{c\}$ are present (variations of this approach belong to the established techniques used to force truth values; see, e.g., Darmann and Döcker [DD21, Lem. 4] for one such example). Analogously, we can obtain three variables which have the forced truth value T by removing any negative clause from $\mathcal{U}_{\min}$. Then, we can use the reduction in Darmann and Döcker [DD21, e.g., Thm. 2] to show NP-completeness of Monotone 3-Sat-$(\leq p, \leq q)$ for $(p, q) \in \{(3, 1), (4, 1)\}$. Hence, all we need to do to settle the two remaining cases is, for each case, to either find an unsatisfiable instance (to be precise, showing existence of an unsatisfiable instance would be sufficient) or to show that all such formulas are satisfiable. While we did not find an unsatisfiable instance of either case so far, we will shed light on some necessary conditions for unsatisfiability of such formulas. To this end, consider a formula

$$\varphi = \{\{\overline{x_1}, \overline{x_2}, \overline{x_3}\}, \{\overline{x_4}, \overline{x_5}, \overline{x_6}\}, \ldots, \{\overline{x_{n-2}}, \overline{x_{n-1}}, \overline{x_n}\}\} \cup \varphi^+,$$

over a set of variables $V = \{x_1, \ldots, x_n\}$, where φ^+ contains only positive clauses and φ is an instance of Monotone 3-Sat-$(\leq p, 1)$ with $p \in \{3, 4\}$. Note that we can always rename the variables in a Monotone 3-Sat-$(\leq p, 1)$ formula such that the negative clauses have the structure shown above. Without loss of generality, we can restrict our focus on truth assignments that set exactly one variable in each negative clause to F (setting multiple variables to F in a negative clause is never necessary since each variable appears in precisely one negative clause).

In order to continue, we need a definition: We say that the formula φ^+ *covers* the set

$$\mathcal{K} = \{x_1, x_2, x_3\} \times \{x_4, x_5, x_6\} \times \ldots \times \{x_{n-2}, x_{n-1}, x_n\}$$

if for each $(y_1, y_2, \ldots, y_{n/3}) \in \mathcal{K}$ there is a clause $c \in \varphi^+$ with $c \subseteq \{y_1, y_2, \ldots, y_{n/3}\}$.

Note that each element $(y_1, y_2, \ldots, y_{n/3}) \in \mathcal{K}$ represents a truth assignment that sets exactly one variable to F in each negative clause (set a variable x_i to F if $x_i = y_j$ for some $j \in \{1, 2, \ldots, n/3\}$ and to T otherwise). Moreover, each truth assignment of this kind is represented by a tuple in $\mathcal{K}$. Now, if we have $c \subseteq \{y_1, y_2, \ldots, y_{n/3}\}$ for some clause $c \in \varphi^+$, then c evaluates to F under the truth assignment represented by

$(y_1, y_2, \ldots, y_{n/3})$. It follows that φ^+ covers $\mathcal{K}$ if and only if φ is unsatisfiable.

Hence, if we find a set of positive 3-clauses, where each variable appears in at most p clauses, that covers $\mathcal{K}$, then we obtain an unsatisfiable instance of Monotone 3-Sat-$(\leq p, 1)$. Finally, by a construction in Darmann and Döcker [DD21], we can even allow up to four (resp. five) 2-clauses for the case $p = 3$ (resp. $p = 4$) to be used in the cover since several copies of the resulting formula, say ψ, can be used to construct an unsatisfiable instance of Monotone 3-Sat-$(\leq p, 1)$. Specifically for the case $p = 3$, if $\psi(z_1, z_2, z_3, z_4)$ denotes the formula where we add z_k to the k-th 2-clause of ψ, we can force the truth value of a variable y to be true in any satisfying truth assignment using

$$\mathcal{F}(y) = \psi(y, u_1, u_1, u_1) \cup \psi'(y, u_2, u_2, u_2) \cup \psi''(y, u_3, u_3, u_3) \cup \{\{\overline{u_1}, \overline{u_2}, \overline{u_3}\}\}$$

where u_1, u_2, u_3 are new variables and the formulas ψ' and ψ'' are obtained from ψ by replacing each variable x_i that appears in ψ by x_i' and x_i'', respectively. Now, it is straightforward to obtain an unsatisfiable instance of Monotone 3-Sat-$(\leq 3, 1)$. The construction can be adapted for the case $p = 4$ with five 2-clauses using $\psi(z_1, z_2, z_3, z_4, z_5)$ and setting $z_5 = u_i$, $i \in \{1, 2, 3\}$, in the definition of $\mathcal{F}(y)$ (e.g., $\psi(y, u_1, u_1, u_1, u_1)$).

Note that the 2-clauses make it easier to find a "good" cover, since a 2-clause covers more elements of $\mathcal{K}$ than a 3-clause does (to be precise, three times as much). However, our efforts to find such a cover with the help of a computer were not successful so far. We repeatedly generated a random monotone Boolean formula that satisfies the required bounds on the variable appearances and then traversed the search space by swapping literals between clauses with the goal to minimize the number of satisfying truth assignments for the formula (similar to the approach described in Döcker [Döc19]).

Planar variants

In this section, we consider variants of Monotone $(2, 3)$-SAT where the incidence graph is planar. The following theorem shows that this problem remains computationally hard in a very restricted setting.

Theorem 13 ([DDD18a, Cor. 2])**.** Planar Monotone $(2, 3)$-SAT-E3 *is NP-complete even if all 3-clauses are positive and each variable appears negated exactly once.*

The result is tight (resp. sharp) with respect to the bound on the number of variable appearances, since satisfiability of any Boolean formula in CNF with at most two appearances per variable can be decided in polynomial time using a result from Tovey [Tov84,

Sec. 3]. By Tovey [Tov84, Thm. 2.4], the problem is in P if we do not allow 2-clauses, i.e., Planar Monotone 3-SAT-E3 is in P. In Darmann et al. [DDD18a], we presented the following natural follow-up question as a research challenge:

> If we require every clause to contain exactly three *distinct* literals, is there a number $s \in \mathbb{N}$ such that Planar Monotone 3-SAT-s is NP-hard and if so, what is the smallest number with this property (clearly, $s \geq 4$)?

In the meantime, this question has been answered by Pilz [Pil19] who showed that all such instances are satisfiable. In order to prove this result, he used a planarity-preserving transformation of the incidence graph and then obtained a satisfying truth assignment via a 4-coloring (which exists by the Four Color Theorem [AH89]).

In contrast, if we allow variables to appear multiple times in the same clause, we obtain the following theorem.

Theorem 14 ([DDD18a, Thm. 5])**.** Planar Monotone 3-SAT*-E4 *is NP-complete.*

A useful property of the NP-complete decision problems considered in Theorem 13 and 14, respectively, is that a planar and orthogonal (edges have only horizontal and vertical segments, i.e., they always bend at a right angle) drawing of the corresponding graphs can be computed in polynomial time by a result from Biedl and Kant [BK98]. Actually, a planar orthogonal drawing was already used along the way to obtain Theorem 13 (to be precise, in the proof [DDD18a, Thm. 1] that Planar Monotone $(2,3)$-SAT-3 is NP-complete). Another example of this approach can be found in Alam et al. [AKM17, Sec. 4] where one of our results (a slightly weaker version of Theorem 13 that we established in our working paper [DDD16, Thm. 2] on planar variants of the monotone satisfiability problem) was used along with a planar orthogonal drawing of the corresponding incidence graph as a starting point for a reduction showing that a particular decision problem related to orthogonal graph drawing is NP-complete.

3.3.4 On a simple hard variant of Not-All-Equal 3-SAT

In this section, we present the main result from our paper [DD20] in which we established NP-completeness of NAE-3-SAT in a very restricted setting. It is well-known that there is a strong connection between Monotone NAE-3-SAT, Set Splitting and Hypergraph 2-Colorability (cf. Porschen et al. [PSSW14, p. 3] and Garey and Johnson [GJ79, p. 221]). In particular, Monotone NAE-3-SAT, Set Splitting where each subset of the input has size 3 and Hypergraph 2-Colorability

where each hyperedge has size 3 are simply different views of the same problem. Hence, the following theorem implies NP-completeness of restricted variants of all three problems. For instance, as mentioned in our paper [DD20], it follows that Hypergraph 2-Colorability for linear, 3-uniform, 4-regular hypergraphs is NP-complete.

Theorem 15 ([DD20, Thm. 2])**.** Linear Monotone NAE-3-SAT-E4 *is NP-complete.*

3.3.5 Quantified variants of the satisfiability problem

In the following, we consider variants of ∀∃ 3-Sat that can be described by a prefix belonging to the set

$$\{\text{Balanced}, \lambda\} \times \{\text{Monotone}, \lambda\},$$

where λ is the empty word, and a suffix belonging to the set

$$\{(s_1, s_2, t_1, t_2) \mid s_1, s_2, t_1, t_2 \in \mathbb{N}\}.$$

Here, Balanced means that the number of universal variables is equal to the number of existential variables and Monotone refers to each clause being monotone. The suffix (s_1, s_2, t_1, t_2), for some fixed values $s_1, s_2, t_1, t_2 \in \mathbb{N}$, indicates that each universal (resp. existential) variable appears exactly s_1 (resp. t_1) times unnegated and exactly s_2 (resp. t_2) times negated. For instance, the decision problem Balanced Monotone ∀∃ 3-SAT-$(1, 1, 2, 2)$ concerns ∀∃ 3-SAT formulas, where the number of universal and existential variables is *balanced* (i.e., there is an equal number of universal and existential variables), each clause is positive or negative (i.e., there is no mixed clause) and each universal variable (resp. existential variable) appears exactly once unnegated and exactly once negated (resp. exactly twice unnegated and exactly twice negated).

Regarding ∀∃ NAE-3-SAT, we consider variants that can be described by a prefix belonging to the set

$$\{\text{Linear}, \lambda\} \times \{\text{Monotone}, \lambda\},$$

and a suffix belonging to the set $\{(s, t) \mid s, t \in \mathbb{N}\}$. Analogously to the notation for NAE-3-SAT, Linear and Monotone refer to the formula being linear and each clause being positive[11], respectively. Further, the suffix (s, t), for some fixed values $s, t \in \mathbb{N}$, indicates that each universal (resp. existential) variable appears exactly s (resp. t) times. For example, Linear Monotone ∀∃ NAE-3-SAT-$(1, 3)$ concerns ∀∃ NAE-3-SAT

[11] Analogously to the unquantified setting, the prefix Monotone has a different meaning for ∀∃ 3-SAT and ∀∃ NAE-3-SAT, respectively.

formulas, where each pair of distinct clauses shares at most one variable, each clause is positive, and each universal variable (resp. existential variable) appears exactly once (resp. exactly three times).

Restrictions of ∀∃ 3-SAT

For the variant of ∀∃ 3-SAT where the number of universal variables is equal to the number of existential variables, we have the following theorem which summarizes two of our results established in Döcker et al. [DDLS20, Thm. 3.4 and Thm. 3.5].

Theorem 16. Balanced ∀∃ 3-SAT-(s_1, s_2, t_1, t_2) *is* Π_2^P*-complete if*

$$(s_1, s_2, t_1, t_2) \in \{(2, 2, 2, 2), (1, 1, 2, 2)\}.$$

In Döcker [Döc19], we further improved upon this result by showing that it also holds if each clause is monotone. Thus, we can state the following theorem.

Theorem 17. Balanced Monotone ∀∃ 3-SAT-(s_1, s_2, t_1, t_2) *is* Π_2^P*-complete if*

$$(s_1, s_2, t_1, t_2) \in \{(2, 2, 2, 2), (1, 1, 2, 2)\}.$$

In order to provide some insight into the gadget constructions used to obtain the hardness results presented in Section 3.3, we now discuss one crucial gadget used in the proof of Theorem 17 (additionally, the following comments complement the rather technical proof sketch in Döcker [Döc19] by focusing on the main construction idea). The latter result was obtained with the help of a gadget that we originally designed for the construction of an unsatisfiable instance of Monotone 3-SAT-(2,2) (see Döcker [Döc19] and Darmann and Döcker [DD21]). This gadget has the form

$$\mathcal{M} = \{\{u_1, u_2\}, \{\overline{u_2}, \overline{u_3}\}, \{\overline{u_2}, \overline{u_4}\}\} \cup \mathcal{U}$$

and consists of three 2-clauses and the set $\mathcal{U}$ that contains 39 monotone 3-clauses. The set of clauses $\mathcal{M}$ satisfies four important properties:

- $\mathcal{M}$ is unsatisfiable,
- $\mathcal{M} \setminus \{C\}$ is satisfiable for each $C \in \{\{u_1, u_2\}, \{\overline{u_2}, \overline{u_3}\}, \{\overline{u_2}, \overline{u_4}\}\}$,
- each clauses is monotone, and
- each variable appears at most twice unnegated and at most twice negated.

The difficult part was finding the set $\mathcal{U}$ such that $\mathcal{M}$ has these properties (for details on the construction of $\mathcal{U}$, we refer to our two above mentioned papers). Now, consider a mixed clause of the form $\{x_i, \overline{x_j}, \overline{x_k}\}$. Using $\mathcal{M}$, we can replace such a clause with

$$\{\{u_1, u_2, x_i\}, \{\overline{u_2}, \overline{u_3}, \overline{x_j}\}, \{\overline{u_2}, \overline{u_4}, \overline{x_k}\}\} \cup \mathcal{U}.$$

Observe that now each clause is monotone and each satisfying truth assignment for the above set of clauses sets at least one literal in $\{x_i, \overline{x_j}, \overline{x_k}\}$ to T. By negating each literal in $\mathcal{M}$, we obtain a gadget that can be used to transform mixed 3-clauses containing only one negated variable. Now, the results in Theorem 17 can be obtained by a reduction from the corresponding non-monotone variant (which is Π_2^P-complete by Theorem 16). Here, we omitted the technical parts of these reductions (see Döcker [Döc19]), e.g., on the construction of a formula with an equal number of universal and existential variables.

If we are only interested in finding the strongest bound with respect to the total number of appearances per variable (i.e., if we do not require the number of unnegated and negated appearances to be balanced for the universal and existential variables), we obtain the following result.

Theorem 18 ([DDLS20, Thm. 3.7]). $\forall\exists$ 3-SAT-$(1, 1, t_1, t_2)$ *is* Π_2^P*-complete if*

$$(t_1, t_2) \in \{(1, 2), (2, 1)\}.$$

Noting that $\forall\exists$ 3-SAT-(s_1, s_2, t_1, t_2) is in NP if $s_1 + s_2 = 1$ and in co-NP if $t_1 + t_2 \leq 2$ (see Haviv et al. [HRTS07, p. 55]), our results mentioned above are in a sense the best possible ones. In particular, the decision problem $\forall\exists$ 3-SAT-$(1, 1, 1, 1)$ is not Π_2^P-complete unless the polynomial hierarchy collapses (cf. Döcker et al. [DDLS20, p. 81]). Now, for the case $s_1 + s_2 = 1$ the following theorem shows that $\forall\exists$ 3-SAT-(s_1, s_2, t_1, t_2) is, in fact, NP-complete in a restricted setting.

Theorem 19 ([DDLS20, Thm. 3.6]). $\forall\exists$ 3-SAT-(s_1, s_2, t_1, t_2) *is NP-complete if* $s_1 + s_2 = 1$ *and*

$$(t_1, t_2) \in \{(1, 2), (2, 1)\}.$$

Again, this result is sharp since $\forall\exists$ 3-SAT-(s_1, s_2, t_1, t_2) is in P for

- $s_1 = s_2 = 0$ (i.e., there are no universal variables) and $(t_1, t_2) \in \{(1, 2), (2, 1)\}$ since such formulas are always satisfiable by a result from Tovey [Tov84, Thm. 2.4], and

- $s_1 + s_2 = 1$ and $t_1 + t_2 \leq 2$ since the universal variables can be omitted in this case (as noted by Haviv et al. [HRTS07, p. 55]), such that each existential variable appears at most twice in the resulting formula. Hence, we can use the algorithm of Tovey [Tov84, Sec. 3] to decide satisfiability in polynomial time (cf. Döcker et al. [DDLS20, p. 82]).

Restrictions of ∀∃ NAE-3-SAT

In this section, we consider variants of ∀∃ NAE-3-SAT. The following theorem shows that this problem remains Π_2^P-complete if each pair of distinct clauses shares at most one variable, there are no negations in the formula, each universal variable appears exactly once, each existential variable appears exactly three times and each clause contains at most one universal variable.

Theorem 20 ([DDLS20, Thm. 4.7])**.** Linear Monotone ∀∃ NAE-3-SAT-$(1, 3)$ *is* Π_2^P*-complete if each clause contains at most one universal variable.*

Interestingly, as mentioned in our paper [DDLS20, p. 73], only one appearance of each universal variable is sufficient to make the problem Π_2^P-hard (recall that ∀∃ 3-SAT becomes easier, i.e., can be shown to be in NP, in this setting). We remark that the bounds given for the universal and existential variable appearances stated in Theorem 20 are the best possible ones, since:

- ∀∃ NAE-3-SAT-$(0, t)$ is equivalent to NAE-3-SAT-Et and, thus, in NP (cf. Döcker et al. [DDLS20, p. 89]). Moreover, for $t = 3$ the problem can be solved in linear time by a result from Porschen et al. [PRS04, Thm. 4].

- Monotone ∀∃ NAE-3-SAT-$(s, 2)$ is in co-NP for each fixed $s \geq 1$ [DDLS20, Cor. 4.9] and turns out to be in P for $s = 1$ [DDLS20, Cor. 4.11]. Furthermore, the decision problem Monotone ∀∃ NAE-3-SAT-$(s, 2)$, for each $s \geq 1$, becomes trivial if each clause contains at most one universal variable—all such formulas are satisfiable [DDLS20, Cor. 4.10].

3.3.6 Using restricted SAT variants in hardness proofs

In this section, we consider for two of our hardness results presented in Section 3.2 how a restricted SAT variant can help to simplify the corresponding proof or to obtain a stronger result, respectively. First, we show how the gadget simulating a truth assignment to a variable in the proof of Theorem 3 could be simplified by reducing from a

variant of the Boolean satisfiability problem with bounded variable appearances. Then, we explain in a less problem-specific manner the benefits of using one of our strongest results regarding restrictions of Monotone 3-SAT as a starting point for a reduction in an NP-hardness proof. Finally, we sketch for the path problem ∀∃ Directed-Disjoint-Connecting-Paths that using a restricted quantified SAT variant—specifically, a restricted variant of ∀∃ 3-SAT—can additionally lead to stronger hardness results.

As alluded to above, we first consider the gadget [DvIKL19, Fig. 4] simulating a truth assignment to a variable in the proof of Theorem 3. Recall that Theorem 3 concerns NP-completeness of deciding the existence of a cherry-picking sequence for two phylogenetic trees (for the corresponding definitions see Section 3.2) and it is obtained by a reduction from 3-Sat. Now, given a conjunction of clauses $c_1, \ldots, c_m$ over a set of variables V, the gadget simulating a truth assignment to a variable $v_k \in V$ is depicted in Figure 3.8 (i.e., such a gadget is introduced for each variable in V). The gadget consists of three phylogenetic trees, where the leaf labels have the following intuitive meaning:

- $v_T^{(k)}, v_F^{(k)}$: the variable v_k is set to T (resp. to F),
- $v_{p_1}^{(k)}, \ldots, v_{p_{|\phi_k|}}^{(k)}$: the unnegated appearances (i.e., positive literals) of v_k,
- $\neg v_{q_1}^{(k)}, \ldots, \neg v_{q_{|\nu_k|}}^{(k)}$: the negated appearances (i.e., negative literals) of v_k, and
- $b_{2k-1}, b_{2k}, b_{2(n+k)-1}, b_{2(n+k)}$: elements ("blocking taxa") that are not in a cherry of the formula gadget [DvIKL19, Fig. 6] (which is omitted here) until a satisfying truth assignment for the given instance has been simulated.

Without going into the technical details of our proof [DvIKL19, Thm. 1], the two elements $v_T^{(k)}, v_F^{(k)}$ are part of precisely the cherries depicted in Figure 3.8 (the three phylogenetic trees are embedded in one of two larger phylogenetic trees at a later stage of the construction which is indicated by the solid and dashed frame; note that the two phylogenetic trees in the dashed frame are defined on disjoint sets). By deleting $v_T^{(k)}$ (resp. $v_F^{(k)}$) and suppressing the resulting vertex of in-degree 1 and out-degree 1, the leaf representing the first unnegated (resp. negated) appearance of v_k ends up in a cherry in the middle (resp. right) phylogenetic tree of the gadget. Intuitively, either the left or the right phylogenetic tree in the dashed frame can be unlocked. For instance, if $v_T^{(k)}$ is deleted, the former phylogenetic tree is consistent with a cherry-picking sequence having the following subsequence (where b is the first appearance of a blocking taxa):

$$v_{p_1}^{(k)} < v_{p_2}^{(k)} < \ldots < v_{p_{|\phi_k|}}^{(k)} < b < \neg v_{q_r}^{(k)}, \quad \forall r \in \{1, \ldots, |\nu_k|\}.$$

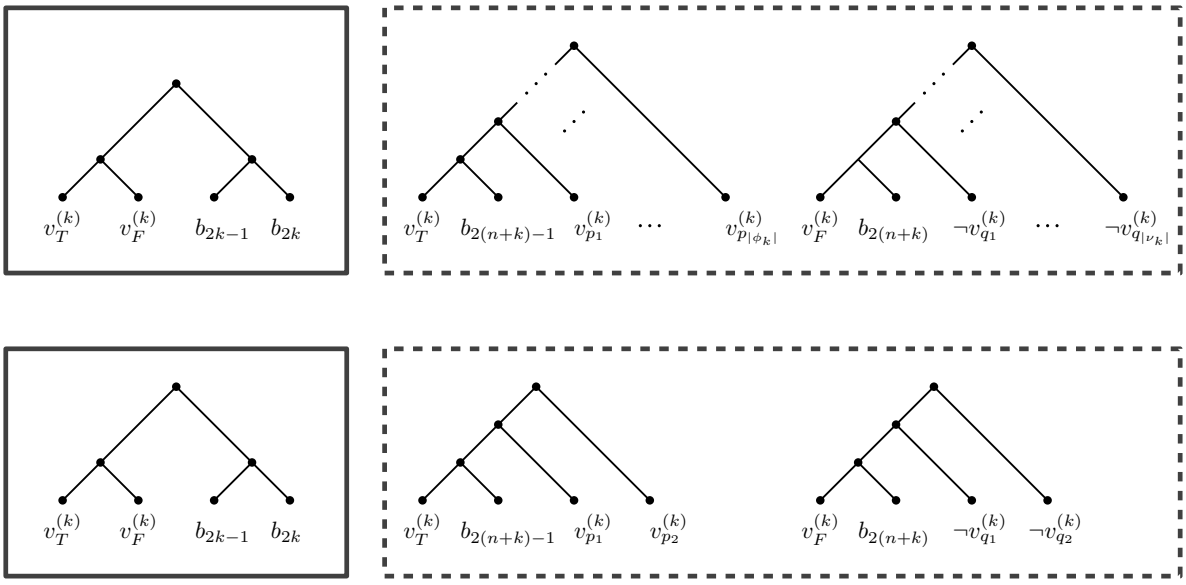


Figure 3.8: Simplification of the gadget [DvIKL19, Fig. 4] consisting of the three phylogenetic trees depicted above which simulates a truth assignment to a variable v_k in the proof of Theorem 3. Recall that the latter result establishes NP-completeness of deciding whether a cherry-picking sequence exists for two phylogenetic trees. To this end, instead of 3-SAT, we may use the still NP-complete [BKS03, Thm. 1] variant 3-SAT-(2,2), where each variable appears exactly twice unnegated and twice negated, respectively. The resulting simplified gadget is depicted below.

Moreover, none of the elements $v_F^{(k)}, \neg v_{q_1}^{(k)}, \ldots, \neg v_{q_{|\nu_k|}}^{(k)}$ precedes b in any cherry-picking sequence where $v_T^{(k)}$ precedes $v_F^{(k)}$. Further, the clause gadget (see Döcker et al. [DvIKL19, Fig. 5]) for a clause c_j requires that at least one of the elements representing a literal in c_j precedes b such that it is consistent with the existence of a cherry-picking sequence.

Now, if we reduce from 3-SAT-(2,2), where each variable appears exactly twice unnegated and exactly twice negated, the variable gadget has constant size (see Figure 3.8). In fact, each phylogenetic tree that is part of a variable gadget has precisely four leaves. Further, since $|\phi_k| = |\nu_k| = 2$, we can omit the explicit references to these index sets in the figure. Thus, the resulting simplified version of the variable gadget removes some of the formalism which was originally present and, thereby, improves readability (i.e., the construction idea becomes more obvious). While the constant size of the variable gadgets does not seem to imply any further meaningful hardness results here, such a bound may very well be useful in obtaining results for other problems even if it does not translate to interesting restrictions of the considered problem (e.g., computer search may be a viable option if we are looking for constructions that have constant size).

To further elaborate on properties of restricted SAT variants that have the potential to simplify reductions or lead to stronger results, let us consider MONOTONE 3-SAT-(2,2) for which NP-completeness follows by Theorem 11 (resp. Döcker [Döc19, Thm. 1]). Say we want to reduce from SAT in order to show NP-hardness of some decision problem by designing gadgets for variables and clauses, respectively, that in a certain sense simulate truth assignments and check whether clauses are satisfied (this approach is called *component design* in the book by Garey and Johnson [GJ79, Sec. 3.2.3]). Then, by our result alluded to above, we can assume all of the following properties to be simultaneously satisfied by the given Boolean formula over variables $V = \{x_1, \ldots, x_n\}$:

- each clause is formed over exactly three distinct variables,
- each clause is monotone,
- there are exactly $\frac{2n}{3}$ negative (resp. positive) clauses, and
- each variable appears exactly twice unnegated and exactly twice negated.

Figure 3.9 shows the structure of an instance of MONOTONE 3-SAT-(2,2). Note that it is sufficient to design a gadget for a variable with exactly two unnegated and two negated appearances, respectively. Therefore, we do not have to pay attention whether or not such a gadget works, e.g., if a variable has only unnegated appearances or whether the construction idea scales to an arbitrary number of variable appearances (as alluded to

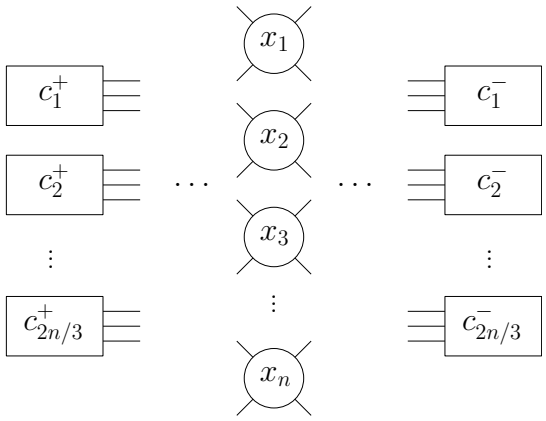


Figure 3.9: Structure of the incidence graph of an instance of Monotone 3-SAT-(2,2). The variables (resp. clauses) are depicted as circular (resp. rectangular) vertices. Further, the positive (resp. negative) clauses are drawn on the left-hand side (resp. right-hand side) of the variables. We note that one can assume that all edges are straight lines if the vertices are positioned as depicted above. However, since all planar instances of this problem are satisfiable [Pil19], this variant cannot be used as a starting point in reductions for obtaining NP-hardness results (assuming P $\neq$ NP).

above, the constant size also opens the door for a search for such a gadget that is aided or completely done by a computer program). Moreover, when designing the gadget for a clause, it may be convenient to assume that either all or none of the contained variables are negated, and that the literals are formed over distinct variables. Noting that gadgets for variables and clauses are usually interconnected (either directly or by another type of gadget), we do not have to worry about potential side effects of having several connections between a variable gadget and a clause gadget (this can happen in general, i.e., if a variable appears more than once in some clause).

We end this section by presenting an example showing how a restricted variant of $\forall\exists$ 3-Sat can be used to obtain stronger hardness results. To this end, we consider the decision problem $\forall\exists$ Directed-Disjoint-Connecting-Paths that we introduced in Section 3.2.3 (recall that we used a variant of this problem in a chain of reductions finally leading to the Π_2^P-completeness result for Display-Set-Equivalence). For convenience, we restate the definition here (π_i denotes a directed path from s_i to t_i):

∀∃ Directed-Disjoint-Connecting-Paths [DLS19, p. 136]
Instance. A directed graph G and two collections

$$P^{\forall} = \{(s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p)\},$$
$$P^{\exists} = \{(s_{p+1}, t_{p+1}), (s_{p+2}, t_{p+2}), \ldots, (s_k, t_k)\}$$

of disjoint pairs of vertices in G such that $1 \leq p < k$ and, for each $(s_i, t_i) \in P^{\forall}$, there exists a directed path from s_i to t_i in G.
Question. For each set $\Pi^{\forall} = \{\pi_1, \pi_2, \ldots, \pi_p\}$ of directed paths, does there exist a set $\Pi^{\forall} \cup \{\pi_{p+1}, \pi_{p+2}, \ldots, \pi_k\}$ of mutually vertex-disjoint directed paths in G?

By reducing from Balanced ∀∃ 3-SAT-(2,2,2,2)—which is Π_2^P-complete by Theorem 17— and using the same variable gadget and clause gadget as in the proof of Theorem 6 (see Döcker et al. [DLS19, Fig. 5] for these gadgets), it can be shown that ∀∃ Directed-Disjoint-Connecting-Paths remains Π_2^P-complete if each vertex in G has degree bounded by 3, where both the in-degree and the out-degree are bounded by 2, and the longest directed path that connects a pair of vertices $(s_i, t_i) \in P^{\forall} \cup P^{\exists}$ contains six arcs (the longest directed path in the construction contains seven arcs[12]). The latter property is a consequence of the bounds on variable appearances satisfied by the quantified Boolean formula. We note that transforming the graph G into a phylogenetic network using the approach taken in Döcker et al. [DLS19, Thm. 4.1] does not preserve a constant bound on path lengths. Ending on a positive note, if we do not require that the constructed phylogenetic network is binary, it is straightforward to adapt the proof such that the longest directed path in the construction is bounded by a constant (simply introduce a root vertex and connect it to the vertex with in-degree 0 in each gadget).

[12] Consider the gadgets in Döcker et al. [DLS19, Fig. 5]. A path starting at a vertex s_i corresponding to a variable gadget and ending at a vertex t_j corresponding to a clause gadget can contain seven arcs (4 arcs of one of the two paths in the variable gadget and then 3 arcs in the clause gadget).

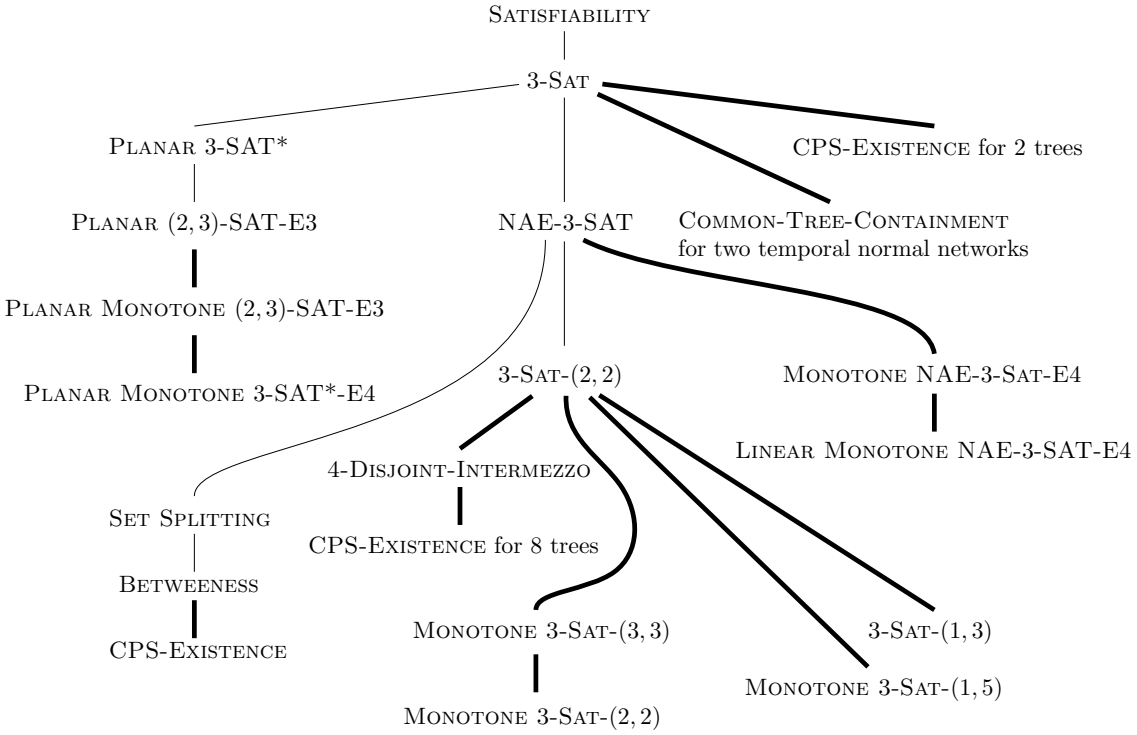


Figure 3.10: A selection of NP-completeness results derived from Satisfiability by a chain of polynomial-time reductions. Each depicted edge is directed downwards and represents a known polynomial-time reduction from the starting point to the end point of this edge (e.g., there is a polynomial-time reduction from 4-Disjoint-Intermezzo to CPS-Existence for 8 trees). Our contributions are indicated by bold edges (for the other edges see the references given in Section 3.3 and the book by Garey and Johnson [GJ79]).

3.4 Concluding remarks

We placed several problems from phylogenetics and (quantified) propositional logic in the polynomial hierarchy. In Figures 3.10 and 3.11, we provide an overview of a selection of hardness results collected in this thesis.

First, we showed that CPS-Existence is NP-complete in restricted settings and that Display-Set-Containment and Display-Set-Equivalence are even harder as they turned out to be Π_2^P-complete. For CPS-Existence, we presented a polynomial-time algorithm using automata theory for the case that (1) the number of trees in the input is bounded by a constant and, additionally, (2) each tree has a number of cherries that is bounded by a constant (these constants may be any fixed positive integers). Recall that, unless P = NP, restriction (1) is not sufficient to obtain a polynomial-time algorithm

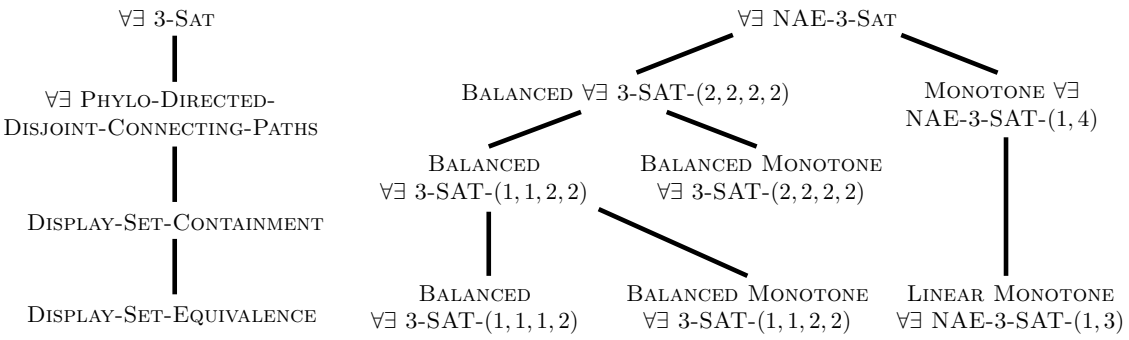


Figure 3.11: Our main Π_2^P-completeness results derived from ∀∃ 3-Sat and ∀∃ NAE-3-Sat by chains of polynomial-time reductions (references regarding the Π_2^P-completeness of the two base problems are given in Section 3.3). As in Figure 3.10, edges are directed downwards and represent polynomial-time reductions from the starting point to the end point of a directed edge.

as we established NP-hardness of CPS-Existence if the input consists of $m \geq 2$ trees. As alluded to in Section 3.2.2, it would be interesting to know whether restriction (2) is sufficient to alleviate the computational complexity. It may also be worth to look at restriction (2) through the lens of parameterized complexity[13]. Specifically, we leave the following questions for future research:

Let $\mathcal{P} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_m\}$ be a collection of phylogenetic X-trees. Recalling that $c_{\mathcal{T}_i}$ denotes the number of cherries in $\mathcal{T}_i$, is CPS-Existence fixed-parameter tractable with respect to the maximum element $c \in \{c_{\mathcal{T}_1}, c_{\mathcal{T}_2}, \ldots, c_{\mathcal{T}_m}\}$? Perhaps it makes sense to first consider the case that m is a constant. In this case, we would like to know whether CPS-Existence can be solved in time $|X|^{\mathcal{O}(1)} \cdot f(c)$, where f is a computable function.

It is worth noting that fixed-parameter algorithms exist for the closely related problem of deciding whether there is a temporal network with at most k reticulations that embeds a given collection of phylogenetic trees when the problem is parameterized by k (this was established by Humphries et al. [HLS13a] for two trees and very recently by Borst et al. [BvIJK20] for an arbitrary number of trees).

It would also be interesting to analyze Display-Set-Containment and Display-Set-Equivalence in the spirit of de Haan [dH19] who developed new parameterized complexity classes that lie between the first and the second level of the polynomial hierarchy.

Regarding Display-Set-Containment and Display-Set-Equivalence, we es-

[13] We refer to the books by Downey and Fellows [DF99] and Niedermeier [Nie06] for an introduction to parameterized complexity.

tablished Π_2^P-completeness of both of these decision problems. As mentioned in Section 3.2, our results have negative implications for a popular approach to tackle NP-hard problems that need to be solved regardless of their inherent computational complexity. Specifically, it means that powerful solvers for problems like SAT are limited in their usefulness with respect to solving Display-Set-Containment and Display-Set-Equivalence as these problems cannot be transformed into any problem in NP within polynomial time unless the polynomial hierarchy collapses (cf. de Haan and Szeider [dHS19]). On the positive side, we showed that Display-Set-Equivalence can be solved in polynomial time if the input consists of a normal network and a tree-child network. However, both problems remain open for some popular classes of phylogenetic networks. In particular, it would be interesting to know whether Display-Set-Equivalence can still be solved in polynomial time if both networks in the input are tree-child networks. Recall that Gunawan et al. [GDZ17] previously asked this question for two reticulation visible networks—which is also still open as of writing this thesis. Since the class of tree-child networks is contained in the class of reticulation visible networks (which follows from the equivalence established by Cardona et al. [CRV09, Lem. 2]), the former class may be better suited when attempting to find a polynomial-time algorithm and the latter one when attempting to show co-NP-hardness (recall that Display-Set-Equivalence is in co-NP for these network classes). Of course, Display-Set-Equivalence could turn out to be in the same complexity class for both network classes.

Second, we obtained hardness results for several restrictions of the satisfiability problem for (quantified) Boolean formulas (see Figure 3.10 for the main NP-completeness results and Figure 3.11 for the main Π_2^P-completeness results). A notable open problem concerns the challenge we presented in Darmann and Döcker [DD21]:

> Is Monotone 3-Sat-$(k, 1)$ NP-hard for $k \in \{3, 4\}$?

In Section 3.3, we hinted at possible ways to approach this problem and deduced that each of these problems is NP-hard if there is an unsatisfiable instance of the respective problem. Hence, we modify the question for future research accordingly and ask the following: Are there unsatisfiable instances of Monotone 3-Sat-$(k, 1)$ for $k \in \{3, 4\}$?

Further, we indicated in Section 3.3.6 how restricted SAT variants are useful in deriving results for other decision problems by the means of polynomial-time reductions—for instance, since the restrictions of a SAT variant reflect in certain properties of the instance generated by the transformation. We also discussed another benefit of having

NP-complete variants of SAT that satisfy strong properties: It makes it easier to design gadgets for a reduction, sometimes even to the point that a computer search for gadgets is a viable option. As a matter of fact, we found the unsatisfiable instance of Monotone 3-SAT-$(2,2)$ using a similar approach: An initial computer search for such an instance was not successful (both trying to enumerate instances of increasing size and generating random instances of different sizes). Hence, we suspected that in case unsatisfiable instances exist, they must have more variables than a desktop computer can find within reasonable time and the search algorithms we used (i.e., brute force and an approach based on the concept of an evolutionary algorithm[14]). We then decomposed the problem of finding an unsatisfiable instance into several smaller problems consisting of finding formulas (or gadgets) that seemed to be better suited for a computer search. Indeed, by the approach based on an evolutionary algorithm, we found the required gadgets and, consequently, we were able to complete our construction of an unsatisfiable instance of Monotone 3-SAT-$(2,2)$ which was a key ingredient in establishing NP-completeness [Döc19, DD21] of the latter problem.

Noting that most NP-completeness results are derived by a chain of polynomial-time reductions from SAT, it would be exciting to see which results can be obtained by considering existing reductions from SAT (or 3-SAT) and to see what happens if, e.g., Monotone 3-SAT-$(2,2)$ is used instead (i.e., in what way the generated instance reflects the variable bounds and the monotonicity requirement). A similar path for future research could analogously be taken for polynomial-time reductions from $\forall\exists$ 3-Sat or $\forall\exists$ NAE-3-Sat with respect to obtaining Π_2^P-completeness results for restricted variants of decision problems. It is hard to say whether this approach leads to meaningful NP- or Π_2^P-completeness results, but if so, the derived results would be low hanging fruits.

[14] See, e.g., the book by Kramer [Kra09, Chap. 2] for an introduction to evolutionary algorithms.

Bibliography

[AH89] K. Appel and W. Haken. *Every planar map is four colorable*, volume 98 of *Contemporary Mathematics*. American Mathematical Soc., 1989.

[AK00] P. Alimonti and V. Kann. Some APX-completeness results for cubic graphs. *Theoretical Computer Science*, 237:123–134, 2000.

[AKM17] M. J. Alam, S. G. Kobourov, and D. Mondal. Orthogonal layout with optimal face complexity. *Computational Geometry*, 63:40–52, 2017.

[BDLN20] M. Bordewich, B. Dorn, S. Linz, and R. Niedermeier. Algorithms and Complexity in Phylogenetics (Dagstuhl Seminar 19443). *Dagstuhl Reports*, 9(10):134–151, 2020.

[BK98] T. Biedl and G. Kant. A better heuristic for orthogonal graph drawings. *Computational Geometry*, 9(3):159–180, 1998.

[BKS03] P. Berman, M. Karpinski, and A. D. Scott. Approximation hardness of short symmetric instances of MAX-3SAT. Electronic Colloquium on Computational Complexity, 2003. Report No. 49.

[BM08] J. A. Bondy and U. S. R. Murty. *Graph Theory*, volume 244 of *Graduate Texts in Mathematics*. Springer, 2008.

[BS07] M. Bordewich and C. Semple. Computing the minimum number of hybridization events for a consistent evolutionary history. *Discrete Applied Mathematics*, 155(8):914–928, 2007.

[BS16] M. Bordewich and C. Semple. Reticulation-visible networks. *Advances in Applied Mathematics*, 76:114–141, 2016.

[BvIJK20] S. Borst, L. van Iersel, M. Jones, and S. Kelk. New FPT algorithms for finding the temporal hybridization number for sets of phylogenetic trees. arXiv:2007.13615 [cs.DS], 2020.

[CHR+20] M. Chudnovsky, S. Huang, P. Rzążewski, S. Spirkl, and M. Zhong. Complexity of C_k-coloring in hereditary classes of graphs. arXiv:2005.01824 [cs.DS], 2020.

[CLS14] P. Cordue, S. Linz, and C. Semple. Phylogenetic networks that display a tree twice. *Bulletin of Mathematical Biology*, 76:2664–2679, 2014.

[Coo71] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158. ACM, 1971.

[CRV09] G. Cardona, F. Rosselló, and G. Valiente. Comparison of tree-child phylogenetic networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 6:552–569, 2009.

[Dar18] A. Darmann. A social choice approach to ordinal group activity selection. *Mathematical Social Sciences*, 93:57–66, 2018.

[dBK12] M. de Berg and A. Khosravi. Optimal binary space partitions for segments in the plane. *International Journal of Computational Geometry & Applications*, 22(03):187–205, 2012.

[DD16] A. Darmann and J. Döcker. Monotone 3-Sat-4 is NP-complete. arXiv:1603.07881 [cs.CC], 2016.

[DD20] A. Darmann and J. Döcker. On a simple hard variant of Not-All-Equal 3-SAT. *Theoretical Computer Science*, 815:147–152, 2020.

[DD21] A. Darmann and J. Döcker. On simplified NP-complete variants of Monotone 3-Sat. *Discrete Applied Mathematics*, 292:45–58, 2021.

[DDD16] A. Darmann, J. Döcker, and B. Dorn. On planar variants of the monotone satisfiability problem with bounded variable appearances. arXiv:1604.05588 [cs.CC], 2016.

[DDD+17] A. Darmann, J. Döcker, B. Dorn, J. Lang, and S. Schneckenburger. On simplified group activity selection. In *Proceedings of the 5th International Conference on Algorithmic Decision Theory (ADT 2017)*, 2017.

[DDD18a] A. Darmann, J. Döcker, and B. Dorn. The monotone satisfiability problem with bounded variable appearances. *International Journal of Foundations of Computer Science*, 29(06):979–993, 2018.

[DDD+18b] A. Darmann, J. Döcker, B. Dorn, J. Lang, and S. Schneckenburger. Simplified group activity selection. In *Proceedings of the Seventh International Workshop on Computational Social Choice (COMSOC-2018)*, 2018.

[DDDS21] A. Darmann, J. Döcker, B. Dorn, and S. Schneckenburger. Simplified group activity selection with group size constraints. Accepted for publication in *International Journal of Game Theory*, 2021.

[DDE+18] J. Döcker, B. Dorn, U. Endriss, R. de Haan, and S. Schneckenburger. Tool auctions. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI-2018)*, 2018.

[DDEK16] J. Döcker, B. Dorn, U. Endriss, and D. Krüger. Complexity and tractability islands for combinatorial auctions on discrete intervals with gaps. In *Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI-2016)*, 2016.

[DDLS20] J. Döcker, B. Dorn, S. Linz, and C. Semple. Placing quantified variants of 3-SAT and Not-All-Equal 3-SAT in the polynomial hierarchy. *Theoretical Computer Science*, 822:72–91, 2020.

[DF99] R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer, 1999.

[dH19] R. de Haan. *Parameterized Complexity in the Polynomial Hierarchy - Extending Parameterized Complexity Theory to Higher Levels of the Hierarchy.* Lecture Notes in Computer Science 11880. Springer-Verlag Berlin Heidelberg, 2019.

[dHS19] R. de Haan and S. Szeider. A compendium of parameterized problems at higher levels of the polynomial hierarchy. *Algorithms*, 12(9), 2019.

[DKY14] E. Demaine, J. Ku, and Y. W. Yu. Class 4 scribe notes. http://courses.csail.mit.edu/6.890/fall14/scribe/lec4.pdf; Instructor: E. Demaine; Notetakers: J. Ku, Y. W. Yu, 2014.

[DL18] J. Döcker and S. Linz. On the existence of a cherry-picking sequence. *Theoretical Computer Science*, 714:36–50, 2018.

[DLS19] J. Döcker, S. Linz, and C. Semple. Displaying trees across two phylogenetic networks. *Theoretical Computer Science*, 796:129–146, 2019.

[DLS21] J. Döcker, S. Linz, and C. Semple. Display sets of normal and tree-child networks. *The Electronic Journal of Combinatorics*, 28(1):P1.8 (21 pages), 2021.

[Döc19] J. Döcker. Monotone 3-SAT-(2,2) is NP-complete. arXiv:1912.08032 [cs.CC], 2019.

[DvIKL19] J. Döcker, L. van Iersel, S. Kelk, and S. Linz. Deciding the existence of a cherry-picking sequence is hard on two trees. *Discrete Applied Mathematics*, 260:131–143, 2019.

[Dö14] J. Döcker. Komplexitätsanalyse von kombinatorischen Auktionen. Master's thesis, University of Tübingen, 2014.

[EG95] T. Eiter and G. Gottlob. Note on the complexity of some eigenvector problems. Technical Report CD-TR 95/89, Christian Doppler Laboratory for Expert Systems, TU Vienna, 1995.

[EIS76] S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM Journal on Computing*, 5(4):691–703, 1976.

[Fil19] I. Filho. Characterizing boolean satisfiability variants. Master's thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2019.

[FS15] R. Francis and M. Steel. Which phylogenetic networks are merely trees with additional arcs? *Systematic Biology*, 64:768–777, 2015.

[GDZ17] A. D. M. Gunawan, B. DasGupta, and L. Zhang. A decomposition theorem and two algorithms for reticulation-visible networks. *Information and Computation*, 252:161–175, 2017.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.

[GM06] W. Guttmann and M. Maucher. Variations on an ordering theme with constraints. In *Fourth IFIP International Conference on Theoretical Computer Science*, pages 77–90. Springer, 2006.

[Gol78] M. E. Gold. Complexity of automaton identification from given data. *Information and Control*, 37(3):302–320, 1978.

[Gut96] S. Gutner. The complexity of planar graph choosability. *Discrete Mathematics*, 159(1–3):119–130, 1996.

[HK10] D. Hall and D. Klein. Finding cognate groups using phylogenies. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1030–1039, 2010.

[HLS13a] P. J. Humphries, S. Linz, and C. Semple. Cherry picking: a characterization of the temporal hybridization number for a set of phylogenies. *Bulletin of Mathematical Biology*, 75:1879–1890, 2013.

[HLS13b] P. J. Humphries, S. Linz, and C. Semple:. On the complexity of computing the temporal hybridization number for two phylogenies. *Discrete Applied Mathematics*, 161:871–880, 2013.

[HRS10] D. H. Huson, R. Rupp, and C. Scornavacca. *Phylogenetic Networks: Concepts, Algorithms and Applications*. Cambridge University Press, 2010.

[HRTS07] I. Haviv, O. Regev, and A. Ta-Shma. On the hardness of satisfiability with bounded occurrences in the polynomial-time hierarchy. *Theory of Computing*, 3:45–60, 2007.

[HU79] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller, J. W. Thatcher, and J. D. Bohlinger, editors, *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Springer, Boston, MA, 1972.

[KMZ20] P. Krysta, M. Mari, and N. Zhi. Ultimate greedy approximation of independent sets in subcubic graphs. arXiv:2001.11997 [cs.DS], 2020.

[KNTX08] I. A. Kanj, L. Nakhleh, C. Than, and G. Xia. Seeing the trees and their branches in the network is hard. *Theoretical Computer Science*, 401:153–164, 2008.

[Koz97] D. C. Kozen. *Automata and Computability*. Undergraduate Texts in Computer Science. Springer, 1997.

[KP17] M. Karpiński and K. Piecuch. On vertex coloring without monochromatic triangles. arXiv:1710.07132 [cs.DS], 2017.

[KP18] M. Karpiński and K. Piecuch. On vertex coloring without monochromatic triangles. In F. Fomin and V. Podolskii, editors, *Computer Science – Theory and Applications, CSR 2018*, volume 10846 of *Lecture Notes in Computer Science*, pages 220–231. Springer, 2018.

[Kra94] J. Kratochvíl. A special planar satisfiability problem and a consequence of its NP-completeness. *Discrete Applied Mathematics*, 52(3):233–252, 1994.

[Kra09] O. Kramer. *Computational Intelligence*. Informatik im Fokus. Springer, 2009.

[LHT10] T. T.-Y. Lam, C.-C. Hon, and J. W. Tang. Use of phylogenetics in the molecular epidemiology and evolutionary studies of viral infections. *Critical Reviews in Clinical Laboratory Sciences*, 47(1):5–49, 2010.

[Li97] W. N. Li. Two-segmented channel routing is strong NP-complete. *Discrete Applied Mathematics*, 78(1–3):291–298, 1997.

[Lic82] D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.

[LSS10] S. Linz, C. Semple, and T. Stadler. Analyzing and reconstructing reticulation networks under timing constraints. *Journal of Mathematical Biology*, 61:715–735, 2010.

[Man83] A. Mansfield. Determining the thickness of graphs is NP-hard. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 93, pages 9–23. Cambridge University Press, 1983.

[MNW+04] B. M. E. Moret, L. Nakhleh, T. Warnow, C. R. Linder, A. Tholse, A. Padolina, J. Sun, and R. Timme. Phylogenetic networks: modeling, reconstructibility, and accuracy. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):13–23, 2004.

[Mor88] B. Moret. Planar NAE3SAT is in P. *ACM SIGACT News*, 19(2):51–54, 1988.

[Mor12] D. Morrison. Time inconsistency in evolutionary networks. http://phylonetworks.blogspot.com/2012/07/time-inconsistency-in-evolutionary.html, 2012. Blog post. Accessed: 03.01.2021.

[MS72] A. R. Meyer and L. J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proceedings of the 13th Annual Symposium on Switching & Automata Theory*, pages 125–129, 1972.

[MSH+14] T. Marcussen, S. R. Sandve, L. Heier, M. Spannagl, M. Pfeifer, I. W. G. S. Consortium, K. S. Jakobsen, B. B. H. Wulff, B. Steuernagel, K. F. X. Mayer, and O.-A. Olsen. Ancient hybridizations among the ancestral genomes of bread wheat. *Science*, 345(6194), 2014.

[Nie06] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms.* Oxford University Press, 2006.

[Opa79] J. Opatrny. Total ordering problem. *SIAM Journal on Computing*, 8(1):111–114, 1979.

[Pil19] A. Pilz. Planar 3-SAT with a clause/variable cycle. *Discrete Mathematics & Theoretical Computer Science*, 21(3), 2019.

[PRS04] S. Porschen, B. Randerath, and E. Speckenmeyer. Linear time algorithms for some not-all-equal satisfiability problems. In E. Giunchiglia and A. Tacchella, editors, *Theory and Applications of Satisfiability Testing (SAT 2003)*, pages 172–187. Springer, 2004.

[PSSW14] S. Porschen, T. Schmidt, E. Speckenmeyer, and A. Wotzlaw. XSAT and NAE-SAT of linear CNF classes. *Discrete Applied Mathematics*, 167:1–14, 2014.

[PSYB06] G. Petersen, O. Seberg, M. Yde, and K. Berthelsen. Phylogenetic relationships of triticum and aegilops and evidence for the origin of the A, B, and D genomes of common wheat (triticum aestivum). *Molecular Phylogenetics and Evolution*, 39(1), 2006.

[Sch78] T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, pages 216–226, 1978.

[SS03] C. Semple and M. Steel. *Phylogenetics*. Oxford University Press, 2003.

[SS17] R. Schwartz and A. A. Schäffer. The evolution of tumour phylogenetics: principles and practice. *Nature Reviews Genetics*, 18(4):213–229, 2017.

[ST13] U. Schöning and J. Torán. *The Satisfiability Problem: Algorithms and Analyses*, volume 3 of *Mathematik für Anwendungen*. Lehmanns Media, 2013.

[Ste16] M. Steel. *Phylogeny: Discrete and random processes in evolution*. CBMS-NSF Regional conference series in Applied Mathematics. SIAM, 2016.

[Sto76] L. J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1976.

[SU02] M. Schaefer and C. Umans. Completeness in the polynomial-time hierarchy: A compendium. *SIGACT News*, 33(3):32–49, 2002.

[Tip16] S. Tippenhauer. On planar 3-SAT and its variants. Master's thesis, Freie Universität Berlin, 2016.

[Tov84] C. A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85–89, 1984.

[vISS10] L. van Iersel, C. Semple, and M. Steel. Locating a tree in a phylogenetic network. *Information Processing Letters*, 110:1037–1043, 2010.

[Wil10] S. J. Willson. Properties of normal phylogenetic networks. *Bulletin of Mathematical Biology*, 72:340–358, 2010.

[WLL+16] M. Willems, E. Lord, L. Laforest, G. Labelle, F.-J. Lapointe, A. M. Di Sciullo, and V. Makarenkov. Using hybridization networks to retrace the evolution of indo-european languages. *BMC Evolutionary Biology*, 16, 2016.

[Wra76] C. Wrathall. Complete sets and the polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):23–33, 1976.

Appendix

1 Accepted Manuscripts

1.1 On the existence of a cherry-picking sequence

The following paper [DL18] is also available online at the following URL: `https://doi.org/10.1016/j.tcs.2017.12.005`.

Theoretical Computer Science 714 (2018) 36–50

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

On the existence of a cherry-picking sequence

Janosch Döcker [a], Simone Linz [b,*]

[a] Department of Computer Science, University of Tübingen, Germany
[b] Department of Computer Science, University of Auckland, New Zealand

ARTICLE INFO

Article history:
Received 21 August 2017
Received in revised form 30 November 2017
Accepted 5 December 2017
Available online 8 December 2017
Communicated by T. Calamoneri

Keywords:
2P2N-3-SAT
Cherry
Cherry-picking sequence
Intermezzo
Phylogenetic tree
Temporal phylogenetic network

ABSTRACT

Recently, the minimum number of reticulation events that is required to simultaneously embed a collection $\mathcal{P}$ of rooted binary phylogenetic trees into a so-called temporal network has been characterized in terms of cherry-picking sequences. Such a sequence is a particular ordering on the leaves of the trees in $\mathcal{P}$. However, it is well-known that not all collections of phylogenetic trees have a cherry-picking sequence. In this paper, we show that the problem of deciding whether or not $\mathcal{P}$ has a cherry-picking sequence is NP-complete for when $\mathcal{P}$ contains at least eight rooted binary phylogenetic trees. Moreover, we use automata theory to show that the problem can be solved in polynomial time if the number of trees in $\mathcal{P}$ and the number of cherries in each such tree are bounded by a constant.

1. Introduction

To represent evolutionary relationships among species, phylogenetic trees have long been a powerful tool. However, as we now not only acknowledge speciation but also non-tree-like processes such as hybridization and lateral gene transfer to be driving forces in the evolution of certain groups of organisms (e.g. bacteria, plants, and fish) [16,20], phylogenetic networks become more widely used to represent ancestral histories. A phylogenetic network is a generalization of a rooted phylogenetic tree. More precisely, such a network is a rooted directed acyclic graph whose leaves are labeled [14].

The following optimization problem, which is biologically relevant and mathematically challenging, motivates much of the theoretical work that has been done in reconstructing phylogenetic networks from phylogenetic trees. Given a collection $\mathcal{P}$ of rooted binary phylogenetic trees on a set of species such that $\mathcal{P}$ correctly represents the tree-like evolution of different parts of the species' genomes, what is the smallest number of reticulation events that is required to simultaneously embed the trees in $\mathcal{P}$ into a phylogenetic network? Here, reticulation events are collectively referring to all non-tree-like events and they are represented by vertices in a phylogenetic network whose in-degree is at least two. Without any structural constraints on a phylogenetic network, it is well-known that $\mathcal{P}$ can always be embedded into such a network [2,19] and, hence, the optimization problem is well-defined. Moreover, despite the problem being NP-hard [4], even for when $|\mathcal{P}| = 2$, several exact algorithms have been developed that, given two rooted phylogenetic trees, construct a phylogenetic network whose number of reticulation events is minimized over the space of all networks that embed both trees [1,7,18,22].

Motivated by the introduction of temporal networks [3,17], which are phylogenetic networks that satisfy several time constraints, Humphries et al. [12,13] recently investigated the special case of the aforementioned optimization problem for

* Corresponding author.
E-mail addresses: janosch.doecker@uni-tuebingen.de (J. Döcker), s.linz@auckland.ac.nz (S. Linz).

https://doi.org/10.1016/j.tcs.2017.12.005
0304-3975/

when one is interested in minimizing the number of reticulation events over the smaller space of all temporal networks that embed a given collection of rooted binary phylogenetic trees. More precisely, in the context of their two papers, the authors considered *temporal networks* to be phylogenetic networks that satisfy the following three constraints:

(1) speciation events occur successively,
(2) reticulation events occur instantaneously, and
(3) each non-leaf vertex has a child whose in-degree is one.

The second constraint implies that the three species that are involved in a reticulation event, i.e. the new species resulting from this event and its two distinct parents, must coexist in time. Moreover, a phylogenetic network that satisfies the third constraint (but not necessarily the first two constraints) is referred to as a *tree-child* network in the literature [6]. Intuitively, if a phylogenetic network $\mathcal{N}$ is temporal, then one can assign a time stamp to each of its vertices such that the following holds for each edge (u, v) in $\mathcal{N}$. If v is a reticulation, then the time stamp assigned to u is the same as the time stamp assigned to v. Otherwise, the time stamp assigned to v is strictly greater than that assigned to u. Baroni et al. [3] showed that it can be checked in polynomial time whether or not a given phylogenetic network satisfies the first two constraints.

Humphries et al. [12] have established a new characterization to compute the minimum number of reticulation events that is needed to simultaneously embed an arbitrarily large collection $\mathcal{P}$ of rooted binary phylogenetic trees into a temporal network. This characterization, which is formally defined in Section 2, is in terms of *cherries*, and the existence of a particular type of sequence on the leaves of the trees, called a *cherry-picking sequence*. It was shown that such a sequence for $\mathcal{P}$ exists if and only if the trees in $\mathcal{P}$ can simultaneously be embedded into a temporal network [12, Theorem 1]. Moreover, a cherry-picking sequence for $\mathcal{P}$ can be exploited further to compute the minimum number of reticulation events that is needed over all temporal networks. Importantly, not every collection $\mathcal{P}$ is guaranteed to have a solution, i.e. there may be no cherry-picking sequence for $\mathcal{P}$ and, hence no temporal network that embeds all trees in $\mathcal{P}$. It was left as an open problem by Humphries et al. [12] to analyze the computational complexity of deciding whether or not $\mathcal{P}$ has a cherry-picking sequence for when $|\mathcal{P}| = 2$.

In this paper, we make progress towards this question and show that it is NP-complete to decide if $\mathcal{P}$ has a cherry-picking sequence for when $|\mathcal{P}| \geq 8$. Translated into the language of phylogenetic networks, this result directly implies that it is computationally hard to decide if a collection of at least eight rooted binary phylogenetic trees can simultaneously be embedded into a temporal network. To establish our result, we use a reduction from a variant of the Intermezzo problem [9]. On a more positive note, we show that deciding if $\mathcal{P}$ has a cherry-picking sequence can be done in polynomial time if the number of trees and the number of cherries in each such tree are bounded by a constant. To this end, we explore connections between phylogenetic trees and automata theory and show how the problem at hand can be solved by using a deterministic finite automaton.

The remainder of the paper is organized as follows. The next section contains notation and terminology that is used throughout the paper. Section 3 establishes NP-completeness of a variant of the Intermezzo problem which is then, in turn, used in Section 4 to show that it is NP-complete to decide if $\mathcal{P}$ has a cherry-picking sequence for when $|\mathcal{P}| \geq 8$. In Section 5, we show that deciding if $\mathcal{P}$ has a cherry-picking sequence is polynomial-time solvable if the number of cherries in each tree and the size of $\mathcal{P}$ are bounded by a constant. We finish the paper with some concluding remarks in Section 6.

2. Preliminaries

This section provides notation and terminology that is used in the subsequent sections. Throughout this paper, X denotes a finite set.

Phylogenetic trees. A *rooted binary phylogenetic X-tree* $\mathcal{T}$ is a rooted tree with leaf set X and, apart from the root which has degree two, all interior vertices have degree three. Furthermore, a pair of leaves $\{a, b\}$ of $\mathcal{T}$ is called a *cherry* if a and b are leaves that are adjacent to a common vertex. Note that every rooted binary phylogenetic tree has at least one cherry. We denote by $c_{\mathcal{T}}$ the number of cherries in $\mathcal{T}$. We now turn to a rooted binary phylogenetic tree with exactly one cherry. More precisely, we call $\mathcal{T}$ a *caterpillar* if $|X| = n \geq 2$ and the elements in X can be ordered, say $x_1, x_2, \ldots, x_n$, so that $\{x_1, x_2\}$ is a cherry and, if p_i denotes the parent of x_i, then, for all $i \in \{3, 4, \ldots, n\}$, we have (p_i, p_{i-1}) as an edge in $\mathcal{T}$, in which case we denote the caterpillar by $(x_1, x_2, \ldots, x_n)$. To illustrate, Fig. 1 shows the caterpillar $(D_1, D_2, \ldots, D_{|A'|})$ with cherry $\{D_1, D_2\}$. Two rooted binary phylogenetic X-trees $\mathcal{T}$ and $\mathcal{T}'$ are said to be *isomorphic* if the identity map on X induces a graph isomorphism on the underlying trees.

Subtrees. Now, let $\mathcal{T}$ be a rooted binary phylogenetic X-tree, and let $X' = \{x_1, x_2, \ldots, x_k\}$ be a subset of X. The minimal rooted subtree of $\mathcal{T}$ that connects all vertices in X' is denoted by $\mathcal{T}(X')$. Furthermore, the rooted binary phylogenetic tree obtained from $\mathcal{T}(X')$ by contracting all non-root degree-2 vertices is the *restriction of $\mathcal{T}$ to X'* and is denoted by $\mathcal{T}|X'$. We also write $\mathcal{T}[-x_1, x_2, \ldots, x_k]$ or $\mathcal{T}[-X']$ for short to denote $\mathcal{T}|(X - X')$. For a set $\mathcal{P} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_m\}$ of rooted binary phylogenetic X-trees, we write $\mathcal{P}|X'$ (resp. $\mathcal{P}[-X']$) when referring to the set $\{\mathcal{T}_1|X', \mathcal{T}_2|X', \ldots, \mathcal{T}_m|X'\}$ (resp. $\{\mathcal{T}_1[-X'], \mathcal{T}_2[-X'], \ldots, \mathcal{T}_m[-X']\}$). Lastly, a rooted binary phylogenetic tree is *pendant* in $\mathcal{T}$ if it can be detached from $\mathcal{T}$ by deleting a single edge.

Cherry-picking sequences. Let $\mathcal{P}$ be a set of rooted binary phylogenetic X-trees with $|X| = n$. We say that an ordering of the elements in X, say $(x_1, x_2, \ldots, x_n)$, is a *cherry-picking sequence* for $\mathcal{P}$ precisely if each x_i with $i \in \{1, 2, \ldots, n-1\}$ labels a leaf of a cherry in each tree that is contained in $\mathcal{P}[-x_1, x_2, \ldots, x_{i-1}]$. Clearly, if $|\mathcal{P}| = 1$, then $\mathcal{P}$ has a cherry-picking sequence. However, if $|\mathcal{P}| > 1$, then $\mathcal{P}$ may or may not have a cherry-picking sequence.

We now formally state the decision problem that this paper is centered around.

CPS-Existence
Instance. A collection $\mathcal{P}$ of rooted binary phylogenetic X-trees.
Question. Does there exist a cherry-picking sequence for $\mathcal{P}$?

The significance of CPS-Existence is the problem's equivalence to the question whether or not all trees in $\mathcal{P}$ can simultaneously be embedded into a rooted phylogenetic network that satisfies the three temporal constraints as alluded to in the introduction.

Automata and languages. Let Σ be an alphabet. A *language* $\mathcal{L}$ is a subset of all possible strings (also called *words*) whose symbols are in Σ. More precisely, $\mathcal{L}$ is a subset of Σ^*, where the operator $*$ is the Kleene star. A *deterministic finite automaton* (or short *automaton*) is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_{\text{ini}}, F)$, where

(i) Q is a finite set of states,
(ii) Σ is a finite alphabet,
(iii) $\delta\colon Q \times \Sigma \to Q$ is a transition relation,
(iv) q_{ini} is the initial state, and
(v) $F \subseteq Q$ are final states.

A given automaton $\mathcal{A}$ *accepts* a word $w = a_1 a_2 \ldots a_n$ if and only if $\mathcal{A}$ is in a final state after having read all symbols from left to right, i.e.

$$\delta(\ldots\delta(\delta(q_{\text{ini}}, a_1), a_2), \ldots a_n) \in F.$$

The language $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^*$ that is *recognized* by $\mathcal{A}$ is defined as the set of words that $\mathcal{A}$ accepts. For the automata constructed in this paper, we have $|F| = 1$ and δ being a total function that maps each pair of a state in Q and a symbol in Σ to a state in Q. For a detailed introduction to automata theory and languages, see the book by Hopcroft and Ullman [11].

3. A variant of the Intermezzo problem

In this section, we establish NP-completeness of a variant of the ordering problem Intermezzo. Let A be a finite set, and let $\mathcal{O}$ be an ordering on the elements in A. For two elements a and b in A, we write $a < b$ precisely if a precedes b in $\mathcal{O}$. With this notation in hand, we now formally state Intermezzo which was shown to be NP-complete via reduction from 3-SAT [9, Lemma 1].

Intermezzo
Instance. A finite set A, a collection B of pairs from A, and a collection C of pairwise-disjoint triples of distinct elements in A.
Question. Does there exist a total linear ordering on the elements in A such that $a_i < a_j$ for each (a_i, a_j) in B, and $a_i < a_j < a_k$ or $a_j < a_k < a_i$ for each (a_i, a_j, a_k) in C?

Example. Consider the following instance of Intermezzo with three pairs and two disjoint triples (when viewed as sets):

$$A = \{a_1, a_2, a_3, a_4, a_5, a_6\},$$
$$B = \{(a_1, a_6), (a_4, a_1), (a_4, a_3)\},$$
$$C = \{(a_1, a_2, a_3), (a_4, a_5, a_6)\}.$$

A total linear ordering on the elements in A that satisfies all constraints defined by B and C is

$$\mathcal{O} = (a_2, a_4, a_3, a_1, a_5, a_6).$$

While each element $a_i \in A$ can appear an unbounded number of times in the input of a given Intermezzo instance, this number is bounded from above by N in the following Intermezzo variant.

N-Disjoint-Intermezzo
Instance. A finite set A, collections $B_1, B_2, \ldots, B_N$ of pairs from A, and collections $C_1, C_2, \ldots, C_N$ of triples of distinct elements in A such that, for each $\ell \in \{1, 2, \ldots, N\}$, the elements in $B_\ell \cup C_\ell$ are pairwise disjoint.
Question. Does there exist a total linear ordering on the elements in A such that

$$a_i < a_j \text{ for each } (a_i, a_j) \in \bigcup_{1 \leq \ell \leq N} B_\ell,$$

and

$$a_i < a_j < a_k \text{ or } a_j < a_k < a_i \text{ for each } (a_i, a_j, a_k) \in \bigcup_{1 \leq \ell \leq N} C_\ell?$$

Let I be an instance of N-Disjoint-Intermezzo, and let $\mathcal{O}$ be an ordering on the elements of A that satisfies the two ordering constraints for each pair and triple in the statement of N-Disjoint-Intermezzo. We say that $\mathcal{O}$ is an N-Disjoint-Intermezzo *ordering* for I.

We next show that 4-Disjoint-Intermezzo is NP-complete via reduction from the following restricted version of 3-SAT.

2P2N-3-SAT
Instance. A set U of variables, and a set $\mathcal{C}$ of clauses, where each clause is a disjunction of exactly three literals, such that each variable appears negated exactly twice and unnegated exactly twice in $\mathcal{C}$.
Question. Does there exist a truth assignment for U that satisfies each clause in $\mathcal{C}$?

Berman et al. [5, Theorem 1] established NP-completeness for 2P2N-3-SAT.

Theorem 3.1. 4-Disjoint-Intermezzo *is* NP*-complete.*

Proof. We show that the construction by Guttmann and Maucher [9, Lemma 1], that was used to show that Intermezzo is NP-complete via reduction from 3-SAT, yields an instance of 4-Disjoint-Intermezzo if we reduce from 2P2N-3-SAT.

Using the same notation as Guttmann and Maucher [9, Lemma 1], their construction is as follows. Let I be an instance of 2P2N-3-SAT that is given by a set of variables $U = \{u_1, \ldots, u_n\}$ and a set of clauses

$$\mathcal{C} = \{(c_{1,1} \vee c_{1,2} \vee c_{1,3}), \ldots, (c_{m,1} \vee c_{m,2} \vee c_{m,3})\},$$

where each $c_{i,j} \in \{u_1, \bar{u}_1, u_2, \bar{u}_2, \ldots, u_n, \bar{u}_n\}$. Furthermore, for $a, b \in \mathbb{N}$, let $a \oplus b$ denote the number $c \in \{1, 2, 3\}$ such that $a + b \equiv c \pmod 3$. We define the following three sets:

$$\begin{aligned} A = {} & \{u_{k,l}, \bar{u}_{k,l} \mid 1 \leq k \leq n \wedge 1 \leq l \leq 3\} \cup \\ & \{c^l_{i,j} \mid 1 \leq i \leq m \wedge 1 \leq j \leq 3 \wedge 1 \leq l \leq 3\}, \end{aligned}$$

$$\begin{aligned} B = {} & \{(u_{k,1}, \bar{u}_{k,3}), (\bar{u}_{k,1}, u_{k,3}) \mid 1 \leq k \leq n\} \cup \\ & \{(c_{i,j,2}, c^1_{i,j}), (c^2_{i,j}, c_{i,j,1}) \mid 1 \leq i \leq m \wedge 1 \leq j \leq 3\} \cup \\ & \{(c^1_{i,j\oplus 1}, c^3_{i,j}) \mid 1 \leq i \leq m \wedge 1 \leq j \leq 3\}, \end{aligned}$$

$$\begin{aligned} C = {} & \{(u_{k,1}, u_{k,2}, u_{k,3}), (\bar{u}_{k,1}, \bar{u}_{k,2}, \bar{u}_{k,3}) \mid 1 \leq k \leq n\} \cup \\ & \{(c^1_{i,j}, c^2_{i,j}, c^3_{i,j}) \mid 1 \leq i \leq m \wedge 1 \leq j \leq 3\}, \end{aligned}$$

where $c_{i,j,l}$ is an abbreviation of $u_{k,l}$ with $u_k = c_{i,j}$. By construction, the elements in C are pairwise-disjoint triples of distinct elements in A and, so, the three sets A, B, and C form an instance of Intermezzo.

Now, we show how the pairs and triples in $B \cup C$ can be partitioned into sets $B_\ell \cup C_\ell$ with $B_\ell \subseteq B$, $C_\ell \subseteq C$, and $1 \leq \ell \leq 4$ such that the elements in $B_\ell \cup C_\ell$ are pairwise disjoint. Recalling that C is a set of pairwise-disjoint triples, we start by setting $B_1 = \emptyset$ and $C_1 = C$. Furthermore, we set

$$\begin{aligned} B_2 = {} & \{(u_{k,1}, \bar{u}_{k,3}), (\bar{u}_{k,1}, u_{k,3}) \mid 1 \leq k \leq n\} \cup \\ & \{(c^1_{i,j\oplus 1}, c^3_{i,j}) \mid 1 \leq i \leq m \wedge 1 \leq j \leq 3\} \end{aligned}$$

and $C_2 = \emptyset$. By construction, it is easy to check that the pairs in B_2 are pairwise disjoint. Lastly, consider the remaining pairs

$$B \setminus B_2 = \{(c_{i,j,2}, c^1_{i,j}), (c^2_{i,j}, c_{i,j,1}) \mid 1 \leq i \leq m \wedge 1 \leq j \leq 3\}$$

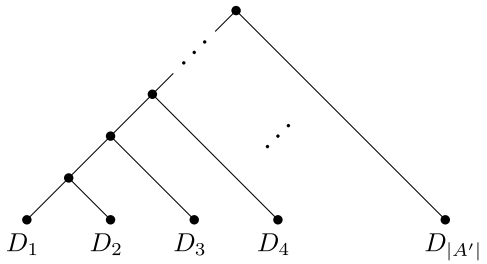


Fig. 1. A caterpillar on $|A'|$ leaves and with cherry $\{D_1, D_2\}$.

and observe that the only possibility for two pairs in $B \setminus B_2$ to have a non-empty intersection is to have an element $c_{i,j,l}$ with $l \in \{1, 2\}$ in common. Now, since each $c_{i,j,l}$ is equal to an element in

$$U' = \{u_{k,l}, \bar{u}_{k,l} \mid 1 \leq k \leq n \wedge 1 \leq l \leq 3\},$$

and each element u_k appears exactly twice negated and twice unnegated in $\mathcal{C}$, it follows that there is a partition of $B \setminus B_2$ into B_3 and B_4 so that all pairs in the resulting two sets are pairwise disjoint. Setting $C_3 = C_4 = \emptyset$ completes the construction of an instance of 4-Disjoint-Intermezzo. Noting that it is straightforward to compute the partition

$$B \cup C = \bigcup_{1 \leq \ell \leq 4} (B_\ell \cup C_\ell)$$

in polynomial time and that we did not modify the construction described by Guttmann and Maucher [9, Lemma 1] itself, it follows from the same proof that I has a satisfying truth assignment if and only if $\bigcup_{1 \leq \ell \leq 4} (B_\ell \cup C_\ell)$ has a 4-Disjoint-Intermezzo ordering. □

Remark. By the construction of an instance of 4-Disjoint-Intermezzo in the proof of Theorem 3.1, we note that no pair or triple occurs twice and that, for each $\ell \in \{1, 2, 3, 4\}$, we have $B_\ell \cup C_\ell \neq \emptyset$. We will freely use these facts throughout the remainder of the paper.

4. Hardness of CPS-Existence

In this section, we show that the decision problem CPS-Existence is NP-complete for any collection of rooted binary phylogenetic trees on the same leaf set that consists of a constant number m of trees with $m \geq 8$. To establish the result, we use a reduction from 4-Disjoint-Intermezzo.

Let I be an instance of 4-Disjoint-Intermezzo. Using the same notation as in the definition of N-Disjoint-Intermezzo, let

$$A' = A \cup \left\{ c_r^1, c_r^2, c_r^3, c_r^4 \mid c_r \in \bigcup_{1 \leq \ell \leq 4} C_\ell \right\},$$

and let $D = \{d_1, d_2, \ldots, d_{|A'|}\}$. For each $\ell \in \{1, 2, 3, 4\}$, we next construct two rooted binary phylogenetic trees. Let A_ℓ be the subset of A' that precisely contains each element of A' that is neither contained in an element of B_ℓ nor contained in an element of

$$C_\ell \cup \{c_r^1, c_r^2, \ldots, c_r^4 \mid c_r \in C_\ell\}.$$

Furthermore, let $\mathcal{S}_\ell$ and $\mathcal{S}'_\ell$ both be the caterpillar shown in Fig. 1. Setting $q = 1$, let $\mathcal{T}_\ell$ and $\mathcal{T}'_\ell$ be the two rooted binary phylogenetic trees obtained from $\mathcal{S}_\ell$ and $\mathcal{S}'_\ell$ that result from the following four-step process.

(i) For each $(a_i, a_j) \in B_\ell$ in turn, replace the leaf D_q in $\mathcal{S}_\ell$ (resp. $\mathcal{S}'_\ell$) with the 3-taxon tree on the top left (resp. bottom left) in Fig. 2 and increment q by one.
(ii) For each $c_r \in C_\ell$ with $c_r = (a_i, a_j, a_k)$ in turn, replace the leaf D_q in $\mathcal{S}_\ell$ (resp. $\mathcal{S}'_\ell$) with the 8-taxon tree on the top right (resp. bottom right) in Fig. 2 and increment q by one.
(iii) For each $a_i \in A_\ell$ in turn, replace the leaf D_q in $\mathcal{S}_\ell$ and $\mathcal{S}'_\ell$ with the cherry $\{a_i, d_q\}$ and increment q by one.
(iv) For each element in $\{q, q+1, \ldots, |A'|\}$, replace the leaf label D_q in $\mathcal{S}_\ell$ and $\mathcal{S}'_\ell$ with d_q.

We call $\mathcal{P}_I = \{\mathcal{T}_\ell, \mathcal{T}'_\ell \mid 1 \leq \ell \leq 4\}$ the set of *intermezzo trees* associated with I. The next observation is an immediate consequence from the above construction and the fact that, for each $1 \leq \ell \leq 4$, the elements in B_ℓ and C_ℓ are pairwise disjoint.

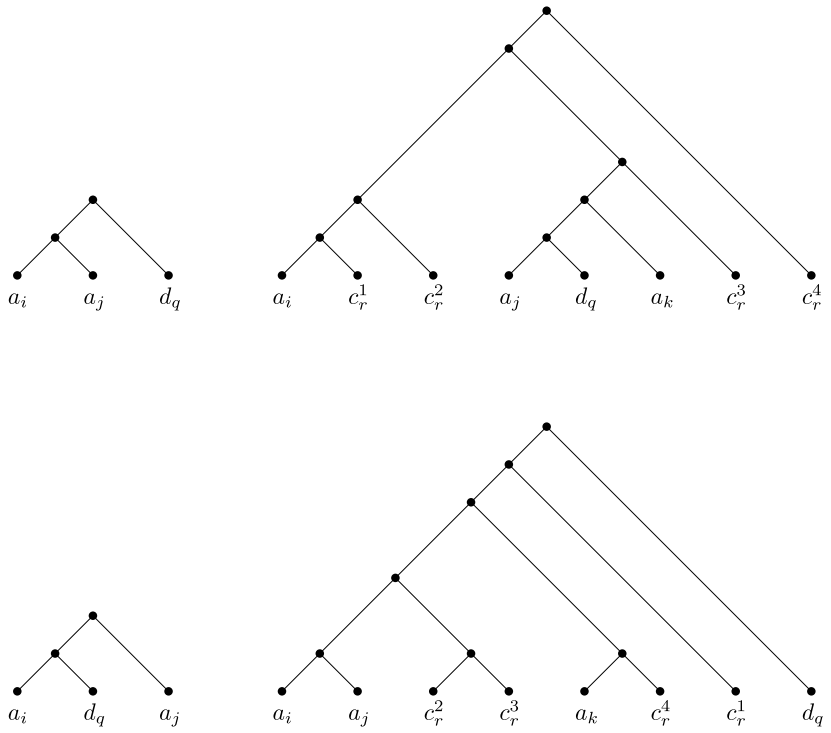


Fig. 2. Gadgets for a pair (a_i, a_j) (left) and gadgets for a triple (a_i, a_j, a_k) (right) that are used in the reduction from 4-Disjoint-Intermezzo to CPS-Existence.

Observation 4.1. *For an instance I of* 4-Disjoint-Intermezzo*, the set of intermezzo trees associated with I consists of eight pairwise non-isomorphic rooted binary phylogenetic trees whose set of leaves is $A' \cup D$.*

We now establish the main result of this section.

Theorem 4.2. *Let $\mathcal{P} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_m\}$ be a collection of rooted binary phylogenetic X-trees.* CPS-Existence *is* NP*-complete for $m = 8$.*

Proof. Clearly, CPS-Existence for $m = 8$ is in NP because, given an ordering $\mathcal{O}$ on the elements in X, we can decide in polynomial time if $\mathcal{O}$ is a cherry-picking sequence for $\mathcal{P}$. Let I be an instance of 4-Disjoint-Intermezzo, and let $\mathcal{P}_I = \{\mathcal{T}_\ell, \mathcal{T}'_\ell \mid 1 \leq \ell \leq 4\}$ be the set of eight intermezzo trees that are associated with I. Note that each tree in $\mathcal{P}_I$ can be constructed in polynomial time and has a size that is polynomial in $|A|$. The remainder of the proof essentially consists of establishing the following claim.

Claim. *I is a 'yes'-instance of* 4-Disjoint-Intermezzo *if and only if $\mathcal{P}_I$ has a cherry-picking sequence.*

First, suppose that $\mathcal{P}_I$ has a cherry-picking sequence. Let $\mathcal{O}$ be a cherry-picking sequence for $\mathcal{P}_I$, and let $\mathcal{O}'$ be the subsequence of $\mathcal{O}$ of length $|A|$ that contains each element in A. We next show that $\mathcal{O}'$ is a 4-Disjoint-Intermezzo ordering for I. Let (a_i, a_j) be an element of some B_ℓ with $1 \leq \ell \leq 4$, and let d_q, with $q \in \{1, 2, \ldots, |A'|\}$, be the unique leaf label of $\mathcal{T}_\ell$ and $\mathcal{T}'_\ell$ such that $\{a_i, a_j, d_q\}$ is the leaf set of a pendant subtree of $\mathcal{T}_\ell$ and $\mathcal{T}'_\ell$. By construction of $\mathcal{T}_\ell$ and $\mathcal{T}'_\ell$, it is easily seen that d_q exists and $a_i < a_j$ in $\mathcal{O}$. Hence, $a_i < a_j$ in $\mathcal{O}'$. Turning to the triples, let $c_r = (a_i, a_j, a_k)$ be an element of some C_ℓ with $1 \leq \ell \leq 4$, and let d_q, with $q \in \{1, 2, \ldots, |A'|\}$, be the unique leaf label of $\mathcal{T}_\ell$ and $\mathcal{T}'_\ell$ such that $\{a_i, a_j, a_k, c_r^1, c_r^2, c_r^3, c_r^4, d_q\}$ is the leaf set of a pendant subtree of $\mathcal{T}_\ell$ and $\mathcal{T}'_\ell$. Again, by construction, d_q exists. Let $\mathcal{S}_\ell = \mathcal{T}_\ell|\{a_i, a_j, a_k, c_r^1, c_r^2, c_r^3, c_r^4, d_q\}$ and, similarly, let $\mathcal{S}'_\ell = \mathcal{T}'_\ell|\{a_i, a_j, a_k, c_r^1, c_r^2, c_r^3, c_r^4, d_q\}$. It is straightforward to check that each cherry-picking sequence for $\mathcal{S}_\ell$ and $\mathcal{S}'_\ell$ satisfies either

$$a_i < a_j < a_k, \text{ or } a_j < a_k < a_i.$$

Hence, as $\mathcal{S}_\ell$ and $\mathcal{S}'_\ell$ are pendant in $\mathcal{T}_\ell$ and $\mathcal{T}'_\ell$, respectively, we have $a_i < a_j < a_k$, or $a_j < a_k < a_i$ in $\mathcal{O}$ and, consequently, in $\mathcal{O}'$. Since the above argument holds for each pair and each triple, it follows that $\mathcal{O}'$ is a 4-Disjoint-Intermezzo ordering for I and, so, I is a 'yes'-instance.

Conversely, suppose that I is a 'yes'-instance of 4-Disjoint-Intermezzo. Let $\mathcal{O}'$ be a 4-Disjoint-Intermezzo ordering on the elements of A. To ease reading, let

$$C = \bigcup_{1 \leq \ell \leq 4} C_\ell.$$

Modify $\mathcal{O}'$ as follows to obtain an ordering $\mathcal{O}$.

(1) Concatenate $\mathcal{O}'$ with the sequence $(d_1, d_2, \ldots, d_{|A'|})$.
(2) For each $c_r = (a_i, a_j, a_k)$ in C, do one of the following two depending on the order of a_i, a_j, and a_k in $\mathcal{O}'$. If $a_i < a_j < a_k$ in $\mathcal{O}'$, then replace a_i with a_i, c_r^2 and replace a_k with a_k, c_r^3, c_r^1, c_r^4. Otherwise, if $a_j < a_k < a_i$, then replace a_k with a_k, c_r^3 and replace a_i with a_i, c_r^2, c_r^1, c_r^4.

Since $\mathcal{O}'$ is a 4-Disjoint-Intermezzo ordering with $a_i < a_j < a_k$ or $a_j < a_k < a_i$ for each $(a_i, a_j, a_k) \in C$, it follows from the construction of $\mathcal{O}$ from $\mathcal{O}'$ that $\mathcal{O}$ is an ordering on the elements in $A' \cup D$. It remains to show that $\mathcal{O}$ is a cherry-picking sequence for $\mathcal{P}_I$. First, consider a pendant subtree with leaf set $\{a_i, a_j, d_q\}$ in $\mathcal{T}_\ell$ and $\mathcal{T}'_\ell$ for some $1 \leq \ell \leq 4$. By construction, (a_i, a_j) is a pair in B_ℓ and, so, we have $a_i < a_j$ in $\mathcal{O}'$ and $a_i < a_j < d_q$ in $\mathcal{O}$. Second, consider a pendant subtree with leaf set $\{a_i, a_j, a_k, c_r^1, c_r^2, c_r^3, c_r^4, d_q\}$ in $\mathcal{T}_\ell$ and $\mathcal{T}'_\ell$ for some $1 \leq \ell \leq 4$. By construction, (a_i, a_j, a_k) is a triple in C_ℓ and, so, we have either $a_i < a_j < a_k$ in $\mathcal{O}'$ and

$$a_i < c_r^2 < a_j < a_k < c_r^3 < c_r^1 < c_r^4 < d_q$$

in $\mathcal{O}$, or $a_j < a_k < a_i$ in $\mathcal{O}'$ and

$$a_j < a_k < c_r^3 < a_i < c_r^2 < c_r^1 < c_r^4 < d_q$$

in $\mathcal{O}$. Third, consider a pendant subtree with leaf set $\{a_i, d_q\}$ in $\mathcal{T}_\ell$ and $\mathcal{T}'_\ell$ for some $1 \leq \ell \leq 4$. By construction, we have $a_i < d_q$ in $\mathcal{O}$. Fourth, if $(a_i, a_j, a_k) \in C$, then, as I has a 4-Disjoint-Intermezzo ordering, there does not exist a pair (a_k, a_j) in B_ℓ for some $1 \leq \ell \leq 4$. Lastly, observe that $(d_1, d_2, \ldots, d_{|A'|})$ is a suffix of $\mathcal{O}$ and that, for any two trees, say $\mathcal{S}$ and $\mathcal{S}'$ in $\mathcal{P}_I$, we have that $\mathcal{S}|D$ and $\mathcal{S}'|D$ are isomorphic. Since $\mathcal{O}'$ is a 4-Disjoint-Intermezzo ordering, it is now straightforward to check that $\mathcal{O}$ is a cherry-picking sequence of $\mathcal{P}_I$. This establishes the proof of the claim and, thereby, the theorem. □

The next corollary shows that CPS-Existence is not only NP-complete for a collection of eight rooted binary phylogenetic trees on the same leaf set, but for any such collection with a fixed number m of trees with $m \geq 8$.

Corollary 4.3. *Let $\mathcal{P} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_m\}$ be a collection of rooted binary phylogenetic X-trees.* CPS-Existence *is* NP*-complete for any fixed m with $m \geq 8$.*

Proof. Clearly, CPS-Existence for $m = t + 8$ with $t \geq 0$ is in NP. To establish the corollary, we show how one can modify the reduction that is described prior to Theorem 4.2 to obtain a set of $t + 8$ rooted binary phylogenetic trees from an instance of 4-Disjoint-Intermezzo.

Let I be an instance of 4-Disjoint-Intermezzo. Throughout the remainder of the proof, we assume that there exists an $1 \leq \ell \leq 4$ such that $|B_\ell \cup C_\ell| > t$. Otherwise, since $t = m - 8$ and m is fixed, it follows that I has a constant number c of pairs and triples with $c \leq 4(m-8)$ and is solvable in polynomial time.

Now, let B_i and C_i with $i \in \{1, 2, 3, 4\}$ be a collection of pairs and triples, respectively, such that $|B_i \cup C_i| > t$. Theorem 4.2 establishes the result for when $t = 0$. We may therefore assume that $t > 0$ and consider two cases. First, suppose that t is even. Replace B_i and C_i in I with a partition of $B_i \cup C_i$ into $\frac{t}{2} + 1$ sets. Each of the resulting new sets can be split naturally into a collection of pairs and a collection of triples of which at most one is empty. This results in

$$\left(\frac{t}{2} + 1\right) + (4 - 1) = \frac{t}{2} + 4$$

collections of pairs and triples, respectively. Now, for each B_ℓ and C_ℓ with $\ell \in \{1, 2, \ldots, \frac{t}{2} + 4\}$, construct two rooted binary phylogenetic trees as described in the definition of the set of intermezzo trees associated with I. This yields

$$2\left(\frac{t}{2} + 4\right) = t + 8 = m$$

pairwise non-isomorphic trees. Second, suppose that t is odd. Replace B_i and C_i in I with a partition of $B_i \cup C_i$ into $\frac{t-1}{2} + 1$ sets. Additionally, add $B_* = \emptyset$ and $C_* = \emptyset$. Analogous to the first case, this results in

$$\left(\frac{t-1}{2} + 1 + 1\right) + (4 - 1) = \frac{t-1}{2} + 5$$

collections of pairs and triples, respectively. Again, for each B_ℓ and C_ℓ with $\ell \in \{1, 2, \ldots, \frac{t-1}{2} + 5\}$ construct two rooted binary phylogenetic trees as described in the definition of the set of intermezzo trees associated with I. Noting that the two trees for B_* and C_* are isomorphic, it follows that the construction yields

$$2\left(\frac{t-1}{2}+5\right)-1=\frac{2t-2}{2}+10-1=t+8=m$$

pairwise non-isomorphic trees. Since the proof of Theorem 4.2 generalizes to a set of m intermezzo trees, the corollary now follows for both cases. □

5. Bounding the number of cherries

The main result of this section is the following theorem.

Theorem 5.1. *Let $\mathcal{P}=\{\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_m\}$ be a collection of rooted binary phylogenetic X-trees. Let c be the maximum element in $\{c_{\mathcal{T}_1},c_{\mathcal{T}_2},\ldots,c_{\mathcal{T}_m}\}$. Then solving* CPS-Existence *for $\mathcal{P}$ takes time*

$$O\left(|X|^{m(4c-2)+1}+\sum_{i=1}^{m} f_i(|X|,c_{\mathcal{T}_i})\right),$$

where $f_i(|X|,c_{\mathcal{T}_i})\in|X|^{O(c_{\mathcal{T}_i})}$. In particular, the running time is polynomial in $|X|$ if c and m are constant.

Let $\mathcal{T}$ be a rooted binary phylogenetic X-tree. We denote by $\mathcal{C}(\mathcal{T})$ the recursively defined set of trees that contains $\mathcal{T}$ and $\emptyset$, and that satisfies the following property.

(P) If a tree $\mathcal{T}'$ is in $\mathcal{C}(\mathcal{T})$ and $\{a,b\}$ is a cherry in $\mathcal{T}'$, then $\mathcal{T}'[-a]$ and $\mathcal{T}'[-b]$ are also contained in $\mathcal{C}(\mathcal{T})$.

We refer to $\mathcal{C}(\mathcal{T})$ as the *set of cherry-picked trees* of $\mathcal{T}$. Intuitively, $\mathcal{C}(\mathcal{T})$ contains each tree that can be obtained from $\mathcal{T}$ by repeatedly deleting a leaf of a cherry.

To establish Theorem 5.1, we consider the set $\mathcal{C}(\mathcal{T})$ of cherry-picked trees of $\mathcal{T}$. First, we develop a new vector representation for each tree in $\mathcal{C}(\mathcal{T})$ and show that the size of $\mathcal{C}(\mathcal{T})$ is at most $(|X|+1)^{O(c_{\mathcal{T}})}$. We then construct an automaton whose number of states is $|\mathcal{C}(\mathcal{T})|+1$ and that recognizes whether or not a word that contains each element in X precisely once is a cherry-picking sequence for $\mathcal{T}$. Lastly, we show how to use a product automaton construction to solve CPS-Existence for a set of rooted binary phylogenetic X-trees in time that is polynomial if the number of cherries and the number of trees in $\mathcal{P}$ is bounded by a constant.

We start with a simple lemma, which shows that deleting a leaf of a cherry never increases the number of cherries.

Lemma 5.2. *Let $\mathcal{T}$ be a rooted binary phylogenetic X-tree, and let a be an element of a cherry in $\mathcal{T}$. Then,*

$$c_{\mathcal{T}}-c_{\mathcal{T}[-a]}\in\{0,1\}.$$

Proof. Let b be the unique element in X such that $\{a,b\}$ is a cherry in $\mathcal{T}$. Observe that each cherry of $\mathcal{T}$ other than $\{a,b\}$ is also a cherry of $\mathcal{T}[-a]$. Now, let p be the parent of the parent of a in $\mathcal{T}$, and let c be the child of p that is not the parent of a. If c is a leaf, then it is easily checked that $\{b,c\}$ is a cherry in $\mathcal{T}[-a]$ and, so $c_{\mathcal{T}}-c_{\mathcal{T}[-a]}=0$. On the other hand, if c is not a leaf, then b is not part of a cherry in $\mathcal{T}[-a]$ and, so, $c_{\mathcal{T}}-c_{\mathcal{T}[-a]}=1$. □

We now define a labeled tree that will play an important role throughout the remainder of this section. Let $\mathcal{T}$ be a rooted binary phylogenetic X-tree with cherries $\{\{a_1,b_1\},\{a_2,b_2\},\ldots,\{a_{c_{\mathcal{T}}},b_{c_{\mathcal{T}}}\}\}$. Obtain a tree $\mathcal{T}_{\mathcal{I}}$ from $\mathcal{T}$ as follows.

Step (1). Set $\mathcal{T}_{\mathcal{I}}$ to be $\mathcal{T}$.
Step (2). Delete all leaves of $\mathcal{T}_{\mathcal{I}}$ that are not part of a cherry.
Step (3). Suppress any resulting degree-2 vertex.
Step (4). If the root, say ρ, has degree one, delete ρ.
Step (5). For each cherry $\{a_i,b_i\}$ with $i\in\{1,2,\ldots,c_{\mathcal{T}}\}$, label the parent of a_i and b_i with i, and delete the two leaves a_i and b_i.
Step (6). Bijectively label the non-leaf vertices of $\mathcal{T}_{\mathcal{I}}$ with $c_{\mathcal{T}}+1,c_{\mathcal{T}}+2,\ldots,2c_{\mathcal{T}}-1$.

We call $\mathcal{T}_{\mathcal{I}}$ the *index tree* of $\mathcal{T}$. By construction, $\mathcal{T}_{\mathcal{I}}$ is a labeled rooted binary tree that is unique up to relabeling the internal vertices. To illustrate, an example of the construction of an index tree is shown in Fig. 3. The next observation follows immediately from the construction of an index tree.

Observation 5.3. *Let $\mathcal{T}$ be a rooted binary phylogenetic tree, and let $\mathcal{T}_{\mathcal{I}}$ be the index tree associated with $\mathcal{T}$. The size of $\mathcal{T}_{\mathcal{I}}$ is $O(c_{\mathcal{T}})$. In particular, if the number of cherries in $\mathcal{T}$ is constant, the size of $\mathcal{T}_{\mathcal{I}}$ is $O(1)$.*

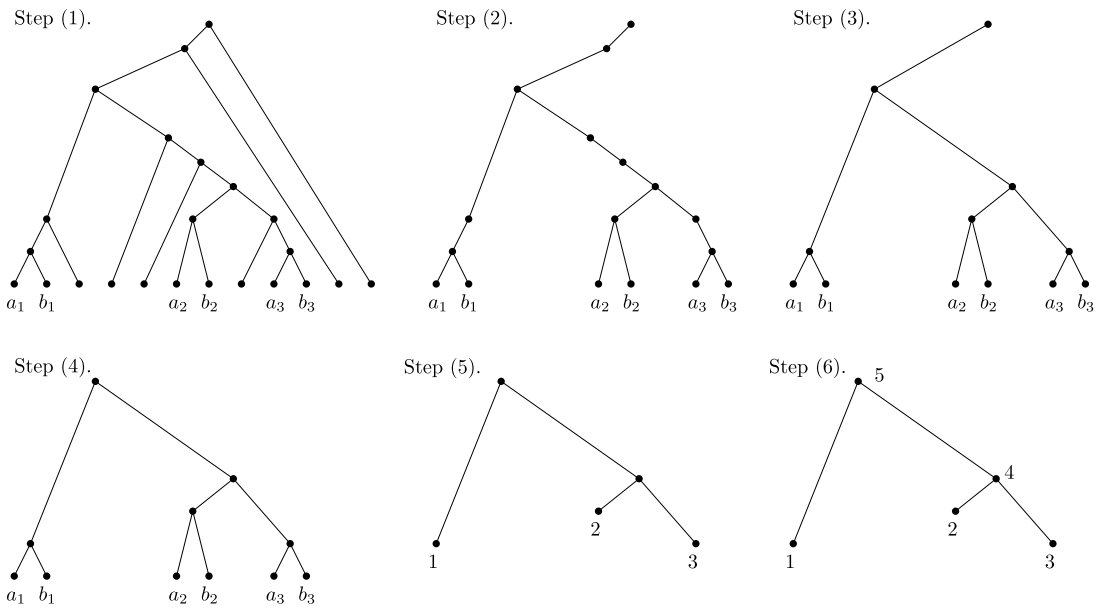


Fig. 3. An example of the construction of an index tree. Steps (1) to (6) refer to the corresponding steps in the definition of an index tree. For simplicity, in Step (1), we have only indicated the leaf labels of leaves that are part of a cherry.

We next define a particular vector relative to a given set. Let S be a finite set, let ϵ be an element that is not in S, and let n be a non-negative integer. We call

$$v = \quad (\quad [\xi_1](x_1^1, x_1^2, \ldots, x_1^{q_1}, \epsilon),$$
$$[\xi_2](x_2^1, x_2^2, \ldots, x_2^{q_2}, \epsilon),$$
$$\vdots$$
$$[\xi_n](x_n^1, x_n^2, \ldots, x_n^{q_n}, \epsilon) \quad)$$

an *S-vector* if each element in S appears at most once in v, each ξ_i is an element in $S \cup \{\epsilon\}$, and each x_i^j is an element in S. Now consider the following two S-vectors:

$$v = \quad (\quad [\xi_1](x_1^1, x_1^2, \ldots, x_1^{q_1}, \epsilon),$$
$$[\xi_2](x_2^1, x_2^2, \ldots, x_2^{q_2}, \epsilon),$$
$$\vdots$$
$$[\xi_n](x_n^1, x_n^2, \ldots, x_n^{q_n}, \epsilon) \quad)$$

and

$$v' = \quad (\quad [\psi_1](y_1^1, y_1^2, \ldots, y_1^{r_1}, \epsilon),$$
$$[\psi_2](y_2^1, y_2^2, \ldots, y_2^{r_2}, \epsilon),$$
$$\vdots$$
$$[\psi_n](y_n^1, y_n^2, \ldots, y_n^{r_n}, \epsilon) \quad)$$

We say that v' has the *suffix-property* relative to v if, for each $s \in \{1, 2, \ldots, n\}$, the vector component $[\psi_s](y_s^1, y_s^2, \ldots, y_s^{r_s}, \epsilon)$ is equal to $[\psi_s](\epsilon)$ or satisfies each of the following equations

$$y_s^{r_s} = x_s^{q_s},\ y_s^{r_s-1} = x_s^{q_s-1},\ \ldots,\ y_s^1 = x_s^{q_s-r_s+1}.$$

Lastly, if v' has the property that $[\psi_i](y_i^1, y_i^2, \ldots, y_i^{r_i}, \epsilon) = [\epsilon](\epsilon)$ for each $i \in \{1, 2, \ldots, n\}$, we call v' the *empty vector*. Note that the empty vector satisfies the suffix-property relative to every S-vector.

Building on the definition of an S-vector, we now describe a vector representation of a rooted binary phylogenetic tree that can be constructed by using its index tree as a guide. Roughly, the representation associates a caterpillar-type structure

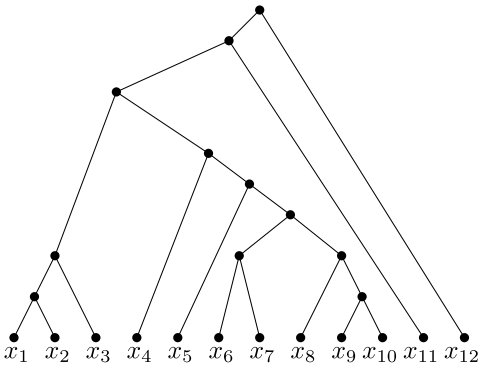


Fig. 4. A rooted binary phylogenetic tree $\mathcal{T}$ whose index tree $\mathcal{T}_\mathcal{I}$ is shown in Step (6) of Fig. 3. The vector representation of $\mathcal{T}$ relative to $\mathcal{T}_\mathcal{I}$ is $([x_1](x_2, x_3, \epsilon), [x_6], (x_7, \epsilon), [x_9](x_{10}, x_8, \epsilon), [\epsilon](x_5, x_4, \epsilon), [\epsilon](x_{11}, x_{12}, \epsilon))$.

to each vertex in the index tree. Let $\mathcal{T}$ be a rooted binary phylogenetic X-tree, let $X' \subseteq X$, and let $\epsilon \notin X$. For two vertices u and v in $\mathcal{T}$, we say that u (resp. v) is an *ancestor* (resp. *descendant*) of v (resp. u) if there is a directed path from u to v in $\mathcal{T}$. Throughout this section, we regard a vertex v of $\mathcal{T}$ to be an ancestor and a descendant of itself. The *most recent common ancestor* of X' is the vertex v in $\mathcal{T}$ whose set of descendants contains X' and no descendant of v, except v itself, has this property. We denote v by $\mathrm{mrca}_\mathcal{T}(X')$. Now, let $\{\{a_1, b_1\}, \{a_2, b_2\}, \ldots, \{a_{c_\mathcal{T}}, b_{c_\mathcal{T}}\}\}$ be the set of all cherries in $\mathcal{T}$. First, for each leaf $i \in \{1, 2, \ldots, c_\mathcal{T}\}$ in $\mathcal{T}_\mathcal{I}$, let $(a_i, x_i^1, x_i^2, \ldots, x_i^q)$ be the maximal pendant caterpillar in $\mathcal{T}$ with cherry $\{a_i, b_i\}$. We denote this by

$$[\xi_i](x_i^1, x_i^2, \ldots, x_i^q, \epsilon),$$

where $\xi_i = a_i$ and $x_i^1 = b_i$. Second, for each non-leaf vertex labeled i in $\mathcal{T}_\mathcal{I}$ with $i \in \{c_\mathcal{T} + 1, c_\mathcal{T} + 2, \ldots, 2c_\mathcal{T} - 1\}$, let v_i be the vertex in $\mathcal{T}$ such that

$$v_i = \mathrm{mrca}_\mathcal{T}(\{a_j, b_j \mid j \text{ is a descendant of } i \text{ in } \mathcal{T}_\mathcal{I}\}),$$

and let $\mathcal{T}_i$ be the rooted binary phylogenetic tree obtained from $\mathcal{T}$ by replacing the pendant subtree rooted at v_i with a leaf labeled v_i. Now, if v_i is a leaf of a cherry in $\mathcal{T}_i$, let $(v_i, x_i^1, x_i^2, \ldots, x_i^q)$ be the maximal pendant caterpillar in $\mathcal{T}_i$ with cherry $\{v_i, x_i^1\}$. We denote this by

$$[\epsilon](x_i^1, x_i^2, \ldots, x_i^q, \epsilon).$$

Otherwise, if v_i is not a leaf of a cherry in $\mathcal{T}_i$, we denote this by

$$[\epsilon](\epsilon).$$

Now, recall that $2c_\mathcal{T} - 1$ is the number of vertices in $\mathcal{T}_\mathcal{I}$. Setting $n = 2c_\mathcal{T} - 1$, we call

$$v_\mathcal{T} = \begin{pmatrix} [\xi_1](x_1^1, x_1^2, \ldots, x_1^{q_1}, \epsilon), \\ [\xi_2](x_2^1, x_2^2, \ldots, x_2^{q_2}, \epsilon), \\ \vdots \\ [\xi_n](x_n^1, x_n^2, \ldots, x_n^{q_n}, \epsilon) \end{pmatrix}$$

the *vector representation of $\mathcal{T}$ relative to $\mathcal{T}_\mathcal{I}$*, and note that

$$\xi_{c_\mathcal{T}+1} = \xi_{c_\mathcal{T}+2} = \cdots = \xi_{2c_\mathcal{T}-2} = \xi_n = \epsilon.$$

An example of a tree and its vector representation is shown in Fig. 4.

Let $\mathcal{T}$ be a rooted binary phylogenetic X-tree with $\epsilon \notin X$, and let $\mathcal{T}_\mathcal{I}$ be the index tree of $\mathcal{T}$. Let $v_\mathcal{T}$ be the vector representation relative to $\mathcal{T}_\mathcal{I}$. Furthermore, let $\mathcal{T}'$ be an element in $\mathcal{C}(\mathcal{T})$, and let

$$v_{\mathcal{T}'} = \begin{pmatrix} [\psi_1](y_1^1, y_1^2, \ldots, y_1^{r_1}, \epsilon), \\ [\psi_2](y_2^1, y_2^2, \ldots, y_2^{r_2}, \epsilon), \\ \vdots \\ [\psi_n](y_n^1, y_n^2, \ldots, y_n^{r_n}, \epsilon) \end{pmatrix}$$

be an X-vector for $\mathcal{T}'$. We say that $v_{\mathcal{T}'}$ has the *cherry-property relative to* $v_{\mathcal{T}}$ if, for each cherry $\{a, b\}$ in $\mathcal{T}'$, exactly one of the following conditions holds:

(i) There is an index $s \in \{1, 2, \ldots, n\}$ such that $\{\psi_s, y_s^1\} = \{a, b\}$.
(ii) There are two distinct indices $s, t \in \{1, 2, \ldots, n\}$ such that $\{\psi_s, \psi_t\} = \{a, b\}$, the two corresponding vector components are $[\psi_s](\epsilon)$ and $[\psi_t](\epsilon)$, respectively, there is a vertex labeled u in $\mathcal{T}_{\mathcal{I}}$ whose two children are labeled s and t, and $\psi_u = \epsilon$.

To establish Theorem 5.1, we next prove three lemmas.

Lemma 5.4. *Let $\mathcal{T}$ be a rooted binary phylogenetic X-tree, and let $v_{\mathcal{T}}$ be the vector representation of $\mathcal{T}$ relative to an index tree of $\mathcal{T}$. Then each tree $\mathcal{T}''$ in $\mathcal{C}(\mathcal{T})$ can be mapped to an X-vector that satisfies the suffix-property and the cherry-property relative to $v_{\mathcal{T}}$. Moreover, the mapping is one-to-one.*

Proof. Set $n = 2c_{\mathcal{T}} - 1$. We define a mapping f from the elements in $\mathcal{C}(\mathcal{T})$ into the set of all X-vectors that satisfy the suffix-property and the cherry-property relative to $v_{\mathcal{T}}$. First, we map $\mathcal{T}$ to $v_{\mathcal{T}}$ and note that $v_{\mathcal{T}}$ satisfies the suffix-property and the cherry-property relative to $v_{\mathcal{T}}$. Second, we map the element $\emptyset$ in $\mathcal{C}(\mathcal{T})$ to the empty vector, say $v_\emptyset$, with n vector components. Again, $v_\emptyset$ satisfies the suffix-property and the cherry-property relative to $v_{\mathcal{T}}$. Now, let $\mathcal{T}''$ be an element in $\mathcal{C}(\mathcal{T}) \setminus \{\mathcal{T}, \emptyset\}$. Recalling the recursive definition of $\mathcal{C}(\mathcal{T})$, there exists a tree $\mathcal{T}'$ in $\mathcal{C}(\mathcal{T})$ with cherry $\{a, b\}$ such that $\mathcal{T}'[-a]$ is isomorphic to $\mathcal{T}''$. Suppose that f (defined below) has already mapped $\mathcal{T}'$ to the X-vector

$$v_{\mathcal{T}'} = \begin{pmatrix} [\psi_1](y_1^1, y_1^2, \ldots, y_1^{r_1}, \epsilon), \\ [\psi_2](y_2^1, y_2^2, \ldots, y_2^{r_2}, \epsilon), \\ \vdots \\ [\psi_n](y_n^1, y_n^2, \ldots, y_n^{r_n}, \epsilon) \end{pmatrix},$$

that satisfies the suffix-property as well as the cherry-property relative to $v_{\mathcal{T}}$. Then f maps $\mathcal{T}''$ to a vector that can be obtained from $v_{\mathcal{T}'}$ in one of the following two cases.

(M1) If there is an index $s \in \{1, 2, \ldots, n\}$ such that $\{\psi_s, y_s^1\} = \{a, b\}$, then f maps $\mathcal{T}''$ to a vector $v_{\mathcal{T}''}$ that is obtained from $v_{\mathcal{T}'}$ by replacing the vector component

$$[\psi_s](y_s^1, y_s^2, \ldots, y_s^{r_s}, \epsilon) \text{ with } [b](y_s^2, y_s^3 \ldots, y_s^{r_s}, \epsilon).$$

(M2) Otherwise, there are two indices $s, t \in \{1, 2, \ldots, n\}$ with $s \neq t$ such that $\{\psi_s, \psi_t\} = \{a, b\}$, where the two corresponding components have the form $[\psi_s](\epsilon)$ and $[\psi_t](\epsilon)$, respectively. Furthermore, by construction, there is a vertex labeled u in $\mathcal{T}_{\mathcal{I}}$ whose two children are the vertices labeled s and t and $\psi_u = \epsilon$. Then, f maps $\mathcal{T}''$ to a vector $v_{\mathcal{T}''}$ that is obtained from $v_{\mathcal{T}'}$ by replacing each of the two vector components

$$[\psi_s](\epsilon) \text{ and } [\psi_t](\epsilon) \text{ with } [\epsilon](\epsilon),$$

and replacing the vector component

$$[\epsilon](y_u^1, y_u^2, \ldots, y_u^{r_u}, \epsilon) \text{ with } [b](y_u^1, y_u^2 \ldots, y_u^{r_u}, \epsilon).$$

For both cases, it is easily checked that $v_{\mathcal{T}''}$ is an X-vector that satisfies the suffix-property relative to $v_{\mathcal{T}}$.

We next show that $v_{\mathcal{T}''}$ satisfies the cherry-property relative to $v_{\mathcal{T}}$. By Lemma 5.2, we have $c_{\mathcal{T}'} - c_{\mathcal{T}''} \in \{0, 1\}$. If $c_{\mathcal{T}'} - 1 = c_{\mathcal{T}''}$ then, by construction, each cherry in $\mathcal{T}''$ is a cherry in $\mathcal{T}'$. Hence, as $v_{\mathcal{T}'}$ satisfies the cherry-property relative to $v_{\mathcal{T}}$, we have that $v_{\mathcal{T}''}$ satisfies the cherry-property relative to $v_{\mathcal{T}}$. Otherwise, if $c_{\mathcal{T}'} = c_{\mathcal{T}''}$, then the cherry $\{a, b\}$ in $\mathcal{T}'$ is replaced with a new cherry that contains b, while all other cherries in $\mathcal{T}'$ are also cherries in $\mathcal{T}''$. First, suppose that $\mathcal{T}''$ is obtained from $\mathcal{T}'$ according to mapping (M1). Observe that $r_s \geq 1$. If $r_s \geq 2$, then $\{b, y_s^2\}$ is the new cherry and, thus, $v_{\mathcal{T}''}$ satisfies the cherry-property relative to $v_{\mathcal{T}}$. On the other hand, if $r_s = 1$, let $[\psi_t](y_t^1, y_t^2, \ldots, y_t^{r_t}, \epsilon)$ be the vector component in $v_{\mathcal{T}'}$ such that the vertices labeled s and t in $\mathcal{T}_{\mathcal{I}}$ have the same parent. Note that t exists because, otherwise, s is the root of $\mathcal{T}_{\mathcal{I}}$ and so the existence of a cherry in $\mathcal{T}''$ that is not a cherry in $\mathcal{T}'$ implies that $r_s \geq 2$; a contradiction. If $r_t \geq 1$, then $\{\psi_t, y_t^1\}$ is a cherry in $\mathcal{T}'$ and $\mathcal{T}''$. Thus, the sibling of b in $\mathcal{T}''$ is not a leaf, thereby contradicting that b is a leaf of a cherry in $\mathcal{T}''$. Hence, $[\psi_t](y_t^1, y_t^2, \ldots, y_t^{r_t}, \epsilon) = [\psi_t](\epsilon)$. Now, as $[b](\epsilon)$ and $[\psi_t](\epsilon)$ are two vector components of $v_{\mathcal{T}''}$, it again follows that $v_{\mathcal{T}''}$ satisfies the cherry-property relative to $v_{\mathcal{T}}$. Second, suppose that $\mathcal{T}''$ is obtained from $\mathcal{T}'$ according to mapping (M2). Noting that b is an element of the vector component $[\psi_u](y_u^1, y_u^2, \ldots, y_t^{r_u}, \epsilon)$ with $\psi_u = b$ in $v_{\mathcal{T}''}$, the result can be established by using an argument that is similar to the previous case.

It remains to show that the mapping is one-to-one. Let $\mathcal{T}'$ and $\mathcal{T}''$ be two distinct elements in $\mathcal{C}(\mathcal{T}) \setminus \{\emptyset\}$. Since each element in $\mathcal{C}(\mathcal{T}) \setminus \{\mathcal{T}, \emptyset\}$ can be obtained from $\mathcal{T}$ by repeatedly deleting a leaf of a cherry and suppressing the resulting degree-2 vertex, there exists an element ℓ in X that is a leaf in $\mathcal{T}'$ and not a leaf in $\mathcal{T}''$. Let X' and X'' be the leaf set of $\mathcal{T}'$ and $\mathcal{T}''$, respectively. Noting that $v_{\mathcal{T}}$ is an X-vector that contains each element in X exactly once, it follows from construction of the mapping that $v_{\mathcal{T}'}$ is an X'-vector that contains each element in X' exactly once and that $v_{\mathcal{T}''}$ is an X''-vector that contains each element in X'' exactly once. Hence $v_{\mathcal{T}'} \neq v_{\mathcal{T}''}$. Moreover, since no element in $\mathcal{C}(\mathcal{T}) \setminus \{\emptyset\}$ is mapped to $v_\emptyset$, the mapping is one-to-one. This completes the proof of the lemma. □

Lemma 5.5. *Let $\mathcal{T}$ be a rooted binary phylogenetic X-tree. Then*

$$|\mathcal{C}(\mathcal{T})| \leq (|X|+1)^{4c_{\mathcal{T}}-2}.$$

Proof. Let

$$\begin{aligned} v_{\mathcal{T}} = \quad (\quad & [\xi_1](x_1^1, x_1^2, \ldots, x_1^{q_1}, \epsilon), \\ & [\xi_2](x_2^1, x_2^2, \ldots, x_2^{q_2}, \epsilon), \\ & \vdots \\ & [\xi_n](x_n^1, x_n^2, \ldots, x_n^{q_n}, \epsilon) \quad) \end{aligned}$$

be the vector representation of $\mathcal{T}$ relative to an index tree of $\mathcal{T}$, where $n = 2c_{\mathcal{T}} - 1$. We first derive an upper bound on the number of X-vectors that satisfy the suffix-property relative to $v_{\mathcal{T}}$. For each $i \in \{1, 2, \ldots, n\}$, consider the vector component $[\xi_i](x_i^1, x_i^2, \ldots, x_i^{q_i}, \epsilon)$. Then each X-vector that satisfies the suffix-property relative to $\mathcal{T}$ has an ith vector component, say $[\psi_i](y_i^1, y_i^2, \ldots, y_i^{r_i}, \epsilon)$, such that $\psi_i \in X \cup \{\epsilon\}$ and $(y_i^1, y_i^2, \ldots, y_i^{r_i}, \epsilon)$ is a suffix of $(x_i^1, x_i^2, \ldots, x_i^{q_i}, \epsilon)$. Since there are at most $|X|+1$ such suffixes, it follows that there are at most $(|X|+1)^2$ variations of $[\psi_i](y_i^1, y_i^2, \ldots, y_i^{r_i}, \epsilon)$. Hence, there are at most

$$((|X|+1)^2)^n = (|X|+1)^{4c_{\mathcal{T}}-2}.$$

X-vectors that satisfy the suffix-property relative to $v_{\mathcal{T}}$. By Lemma 5.4, each tree in $\mathcal{C}(\mathcal{T})$ can be mapped to one such vector and, as the map is one-to-one, it follows that $\mathcal{C}(\mathcal{T})$ contains at most $(|X|+1)^{4c_{\mathcal{T}}-2}$ trees. □

For a rooted binary phylogenetic X-tree $\mathcal{T}$, the next lemma constructs an automaton that recognizes whether or not a word that contains each element in X precisely once is a cherry-picking sequence for $\mathcal{T}$.

Lemma 5.6. *Let $\mathcal{T}$ be a rooted binary phylogenetic X-tree. There is a deterministic finite automaton $\mathcal{A}_{\mathcal{T}}$ with $O(|X|^{4c_{\mathcal{T}}-2})$ states that recognizes the language*

$$\mathcal{L}_X(\mathcal{T}) = \{x_1 x_2 \ldots x_{|X|} \mid (x_1, x_2, \ldots, x_{|X|}) \text{ is a cherry-picking sequence for } \mathcal{T}\}.$$

Moreover, the automaton $\mathcal{A}_{\mathcal{T}}$ can be constructed in time $f(|X|, c_{\mathcal{T}}) \in |X|^{O(c_{\mathcal{T}})}$.

Proof. Throughout this proof, we denote the tree without a vertex by $\emptyset$. Let M and M' be two sets. Setting $M = \{\mathcal{T}\}$ and $M' = \emptyset$, we construct $\mathcal{A}_{\mathcal{T}}$ as follows.

(1) Create the states $q_{\mathcal{T}}$, $q_\emptyset$, and q_e. For each $a \in X$, set $\delta(q_e, a) = \delta(q_\emptyset, a) = q_e$.
(2) For each $\mathcal{T} \in M$ and each $a \in X$ do the following.
  (a) If a is a leaf of a cherry in $\mathcal{T}$ or a is the only vertex of $\mathcal{T}$, then
    (i) create the state $q_{\mathcal{T}[-a]}$ if $\mathcal{T}[-a]$ is not isomorphic to a tree in M',
    (ii) set $M' = M' \cup \mathcal{T}[-a]$, and
    (iii) set $\delta(q_{\mathcal{T}}, a) = q_{\mathcal{T}[-a]}$.
  (b) Otherwise, set $\delta(q_{\mathcal{T}}, a) = q_e$.
(3) Set $M = M'$ and, subsequently, set $M' = \emptyset$. If $M \neq \{\emptyset\}$, continue with (2).

We set the initial state of $\mathcal{A}_{\mathcal{T}}$ to be $q_{\mathcal{T}}$ and the final state to be $q_\emptyset$. To illustrate, the construction of $\mathcal{A}_{\mathcal{T}}$ is shown in Fig. 5 for a phylogenetic tree on four leaves.

By construction, we have $\mathcal{A}_{\mathcal{T}} = (\mathcal{C}(\mathcal{T}), X, \delta, q_{\mathcal{T}}, \{q_\emptyset\})$. As each cherry-picked tree in $\mathcal{C}(\mathcal{T})$ is mapped to a unique state, it follows from Lemma 5.5 that the number of states of $\mathcal{A}_{\mathcal{T}}$ is $O(|X|^{4c_{\mathcal{T}}-2})$. Moreover, for each $a \in X$ and each pair of two distinct states $\mathcal{T}', \mathcal{T}'' \in \mathcal{C}(\mathcal{T})$, there is a transition $\delta(q_{\mathcal{T}'}, a) = q_{\mathcal{T}''}$ if and only if $\mathcal{T}'[-a] = \mathcal{T}''$ and a is either a leaf of a cherry in $\mathcal{T}'$ or $\mathcal{T}'$ consists of the single vertex a. The state q_e collects all inputs that do not correspond to the continuation

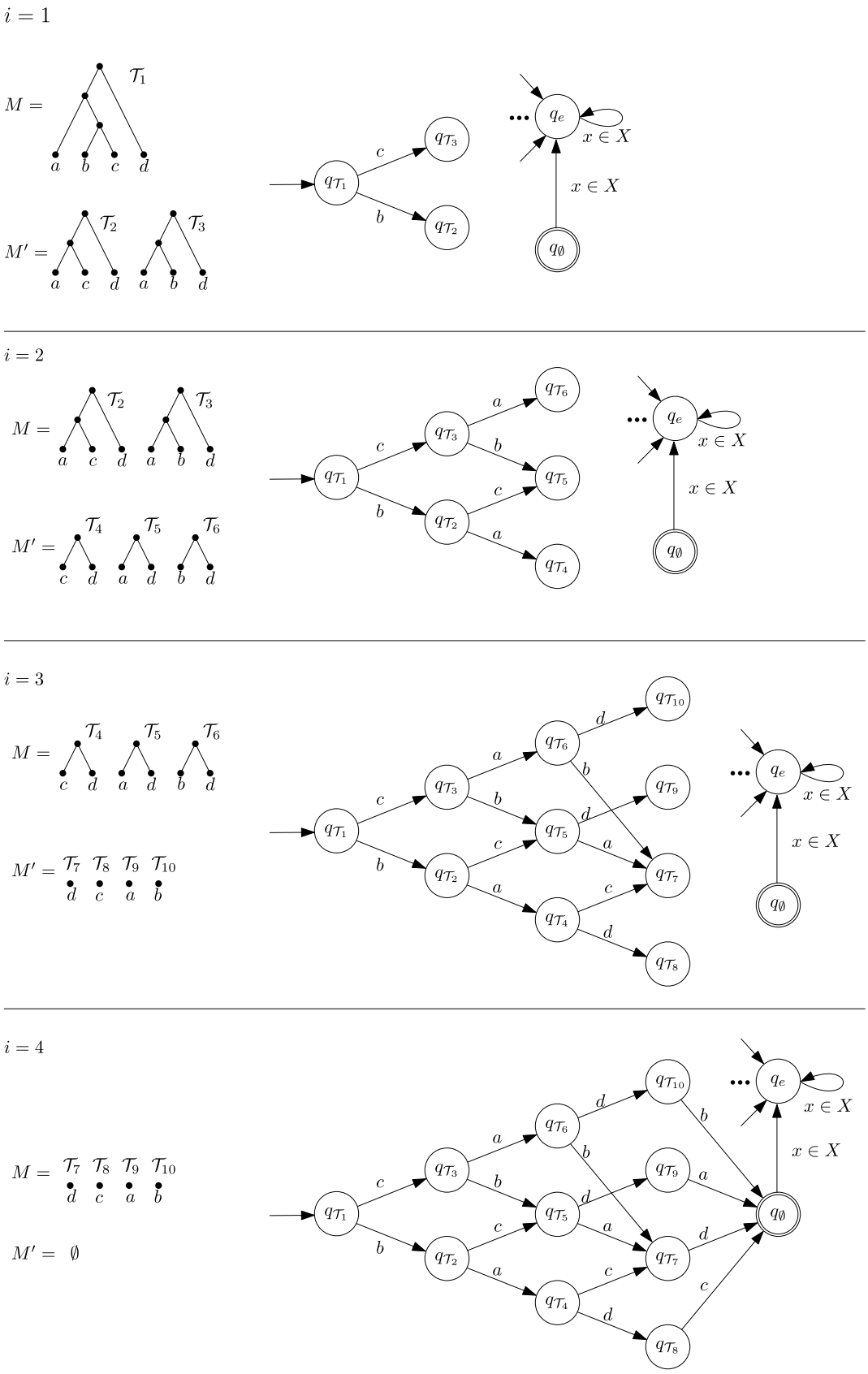


Fig. 5. Construction of an automaton that recognizes the language $\mathcal{L}_X(\mathcal{T}_1)$ as described in the statement of Lemma 5.6 and with $\mathcal{T}_1$ shown in the top left of this figure. Each vertex (resp. edge) represents a state (resp. transition). The vertex $q_{\mathcal{T}_1}$ indicates the initial state whereas the final state is $q_\emptyset$ as indicated by a double circle. To increase readability, most transitions to q_e are omitted. In row i, the figure shows M, M', and the automaton after the ith execution of the for-loop as described in Step (2) in the proof of Lemma 5.6.

of a cherry-picking sequence. More precisely, there is a transition $\delta(q_{\mathcal{T}'}, a) = q_e$ if and only if a is not a leaf of a cherry in $\mathcal{T}'$ and $\mathcal{T}'$ does not consist of the single vertex a. It now follows that there is a one-to-one correspondence between the directed paths from $q_{\mathcal{T}}$ to $q_\emptyset$ in $\mathcal{A}_{\mathcal{T}}$ and the cherry-picking sequences of $\mathcal{T}$ and, hence, $\mathcal{A}_{\mathcal{T}}$ recognizes $\mathcal{L}_X(\mathcal{T})$.

The time taken to construct $\mathcal{A}_{\mathcal{T}}$ is dominated by the number of iterations of the for-loop in Step (2). Since $|M| < |\mathcal{C}(\mathcal{T})|$ and $|\mathcal{C}(\mathcal{T})| \in O(|X|^{4c_{\mathcal{T}}-2})$, the number of iterations in Step (2) is $O(|\mathcal{C}(\mathcal{T})| \cdot |X|) \subseteq O(|X|^{4c_{\mathcal{T}}-1})$. Moreover, since Step (2) is executed $|X|$ times, each operation of the for-loop is executed $O(|X|^{4c_{\mathcal{T}}})$ times in total. While the complexity of these operations depend on the implementation and data structure, they can clearly be implemented such that $\mathcal{A}_{\mathcal{T}}$ can be constructed in time $|X|^{O(c_{\mathcal{T}})}$. This establishes the lemma. □

Generalizing the language that is described in the statement of Lemma 5.6, the next straightforward observation describes a language for the decision problem CPS-Existence.

Observation 5.7. *Let $\mathcal{P} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_m\}$ be a collection of rooted binary phylogenetic X-trees. Then, solving* CPS-Existence *for $\mathcal{P}$ is equivalent to deciding if*

$$\bigcap_{1 \le i \le m} \mathcal{L}_X(\mathcal{T}_i) \neq \emptyset.$$

We are now in a position to establish Theorem 5.1.

Proof of Theorem 5.1. By Observation 5.7, it follows that there is a cherry-picking sequence for $\mathcal{P}$ if and only if $\bigcap_{\mathcal{T}_i \in \mathcal{P}} \mathcal{L}_X(\mathcal{T}_i) \neq \emptyset$, where

$$\mathcal{L}_X(\mathcal{T}) = \{x_1 x_2 \ldots x_{|X|} \mid (x_1, x_2, \ldots, x_{|X|}) \text{ is a cherry-picking sequence for } \mathcal{T}\}.$$

For each $\mathcal{T}_i \in \mathcal{P}$ with $1 \le i \le m$, we follow the notation and construction that is described in the proof of Lemma 5.6 to obtain an automaton $\mathcal{A}_{\mathcal{T}_i}$ with $O(|X|^{4c_{\mathcal{T}_i}-2})$ states that recognizes the language $\mathcal{L}_X(\mathcal{T}_i)$. To solve the question whether or not the intersection of these m languages is empty, we use the well-known construction of a product automaton [15] as follows.

For each $\mathcal{T}_i \in \mathcal{P}$, let $Q_{\mathcal{T}_i}$ be set of states, and let $\delta_{\mathcal{T}_i}$ be the transition relation of $\mathcal{A}_{\mathcal{T}_i}$. We construct a new automaton $\mathcal{A}_{\mathcal{P}}$, where the set of states $Q_{\mathcal{P}}$ is the Cartesian product $Q_{\mathcal{T}_1} \times Q_{\mathcal{T}_2} \times \ldots \times Q_{\mathcal{T}_m}$. Furthermore, the alphabet of $\mathcal{A}_{\mathcal{P}}$ is X and the transition relation $\delta_{\mathcal{P}} \colon Q_{\mathcal{P}} \times X \to Q_{\mathcal{P}}$ is defined as

$$\delta_{\mathcal{P}}((q_1, \ldots, q_m), a) = (\delta_{\mathcal{T}_1}(q_1, a), \ldots, \delta_{\mathcal{T}_m}(q_m, a)).$$

Lastly, the initial (resp. final) state of $\mathcal{A}_{\mathcal{P}}$ is $(q_1, \ldots, q_m)$ where, for all $i \in \{1, 2, \ldots, m\}$, q_i is the initial (resp. final) state of $\mathcal{A}_{\mathcal{T}_i}$. Intuitively, $\mathcal{A}_{\mathcal{P}}$ simulates the parallel execution of the automata $\mathcal{A}_{\mathcal{T}_1}, \mathcal{A}_{\mathcal{T}_2}, \ldots, \mathcal{A}_{\mathcal{T}_m}$. By construction, an input sequence is accepted by $\mathcal{A}_{\mathcal{P}}$ if and only if it is accepted by each automaton $\mathcal{A}_{\mathcal{T}_i}$. It now follows that there is a cherry-picking sequence for $\mathcal{P}$ if and only if the final state of $\mathcal{A}_{\mathcal{P}}$ can be reached from the initial state of $\mathcal{A}_{\mathcal{P}}$ and, hence, $\mathcal{L}(\mathcal{A}_{\mathcal{P}}) \neq \emptyset$.

It remains to show that the computational complexity is as claimed in the statement of the theorem. Viewing $\mathcal{A}_{\mathcal{P}}$ as a directed graph, each directed path from the initial to the final state of $\mathcal{A}_{\mathcal{P}}$ has length $|X|$. We can therefore decide whether the final state from $\mathcal{A}_{\mathcal{P}}$ is reachable by using breadth-first search [8] in time $O(|Q_{\mathcal{P}}| + |X| \cdot |Q_{\mathcal{P}}|)$, where $|X| \cdot |Q_{\mathcal{P}}|$ is the number of transitions in $\mathcal{A}_{\mathcal{P}}$. By construction and Lemma 5.6, it follows that $\mathcal{A}_{\mathcal{P}}$ has

$$O\left(\prod_{i=1}^{m} |X|^{4c_{\mathcal{T}_i}-2}\right) \subseteq O(|X|^{m(4c-2)})$$

states, i.e. $|Q_{\mathcal{P}}| \in O(|X|^{m(4c-2)})$. Hence, we can decide in time $O(|X|^{m(4c-2)+1})$ whether $\mathcal{L}(\mathcal{A}_{\mathcal{P}}) \neq \emptyset$. By Lemma 5.6, it takes time $f_i(|X|, c_{\mathcal{T}_i}) \in |X|^{O(c_{\mathcal{T}_i})}$ to construct each automaton $\mathcal{A}_{\mathcal{T}_i}$ and, thus, it follows that deciding if there is a cherry-picking sequence for $\mathcal{P}$ can be done in time

$$O\left(|X|^{m(4c-2)+1} + \sum_{i=1}^{m} f_i(|X|, c_{\mathcal{T}_i})\right),$$

which is polynomial in $|X|$ if c and m are constant. □

6. Concluding remarks

In this paper, we have shown that CPS-Existence, a problem of relevance to the construction of phylogenetic networks from a set of phylogenetic trees, is NP-complete for all sets $\mathcal{P}$ of rooted binary phylogenetic trees with $|\mathcal{P}| \ge 8$. This result

partially answers a question posed by Humphries et al. [12]. They asked if CPS-Existence is computationally hard for $|\mathcal{P}| = 2$. To establish our result, we first showed that 4-Disjoint-Intermezzo, which is a variant of the Intermezzo problem that is new to this paper, is NP-complete. Subsequently, we established a reduction from an instance I of 4-Disjoint-Intermezzo to an instance I' of CPS-Existence with $|\mathcal{P}| = 8$. Since each of the four collections of pairs and triples in I reduces to two trees in I', a possible approach to obtain a stronger hardness result for CPS-Existence with $|\mathcal{P}| < 8$ is to show that N-Disjoint-Intermezzo is NP-complete for $N < 4$. However, it seems likely that such a result can only be achieved by following a strategy that is different from the one that we used in this paper. In particular, there is no obvious reduction from 2P2N-3-SAT to 3-Disjoint-Intermezzo. Moreover, 1-Disjoint-Intermezzo is solvable in polynomial time since all pairs and triples are pairwise disjoint and, so, it cannot be used for a reduction even if CPS-Existence turns out to be NP-complete for $|\mathcal{P}| = 2$.

In the second part of the paper, we have translated CPS-Existence into an equivalent problem on languages and used automata theory to show that CPS-Existence can be solved in polynomial time if the number of trees in $\mathcal{P}$ and the number of cherries in each such tree are bounded by a constant. There are currently only a small number of other problems in phylogenetics that have been solved with the help of automata theory (e.g. [10,21]) and it is to be hoped that the results presented in this paper will stimulate further research to explore connections between combinatorial problems in phylogenetics and automata theory.

Acknowledgements

We thank Britta Dorn for insightful comments on a draft version of this paper. The second author was supported by the New Zealand Marsden Fund.

References

[1] B. Albrecht, C. Scornavacca, A. Cenci, D.H. Huson, Fast computation of minimum hybridization networks, Bioinformatics 28 (2012) 191–197.
[2] M. Baroni, S. Grünewald, V. Moulton, C. Semple, Bounding the number of hybridization events for a consistent evolutionary history, J. Math. Biol. 51 (2005) 171–182.
[3] M. Baroni, C. Semple, M. Steel, Hybrids in real time, Syst. Biol. 55 (2006) 46–56.
[4] M. Bordewich, C. Semple, Computing the minimum number of hybridization events for a consistent evolutionary history, Discrete Appl. Math. 155 (2007) 914–928.
[5] P. Berman, M. Karpinski, A.D. Scott, Approximation Hardness of Short Symmetric Instances of MAX-3SAT, Electronic Colloquium on Computational Complexity, 2003, Report No. 49.
[6] G. Cardona, F. Rosselló, G. Valiente, Comparison of tree-child phylogenetic networks, IEEE/ACM Trans. Comput. Biol. Bioinform. 6 (2009) 552–569.
[7] Z.Z. Chen, L. Wang, An ultrafast tool for minimum reticulate networks, J. Comput. Biol. 20 (2013) 38–41.
[8] T.H. Cormen, C.E. Leierson, R.L. Rivest, C. Stein, Introduction to Algorithms, The MIT Press, 2009.
[9] W. Guttmann, M. Maucher, Variations on an ordering theme with constraints, in: Fourth IFIP International Conference on Theoretical Computer Science, Springer, 2006, pp. 77–90.
[10] D. Hall, D. Klein, Finding cognate groups using phylogenies, in: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, 2010, pp. 1030–1039.
[11] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison–Wesley, 1979.
[12] P.J. Humphries, S. Linz, C. Semple, Cherry picking: a characterization of the temporal hybridization number for a set of phylogenies, Bull. Math. Biol. 75 (2013) 1879–1890.
[13] P.J. Humphries, S. Linz, C. Semple, On the complexity of computing the temporal hybridization number for two phylogenies, Discrete Appl. Math. 161 (2013) 871–880.
[14] D.H. Huson, R. Rupp, C. Scornavacca, Phylogenetic Networks: Concepts, Algorithms and Applications, Cambridge University Press, 2010.
[15] D.C. Kozen, Automata and Computability, Undergraduate Texts in Computer Science, Springer, 1997.
[16] J. Mallet, N. Besansky, M.W. Hahn, How reticulated are species?, BioEssays 38 (2016) 140–149.
[17] B.M.E. Moret, L. Nakhleh, T. Warnow, C.R. Linder, A. Tholse, A. Padolina, J. Sun, R. Timme, Phylogenetic networks: modeling, reconstructibility, and accuracy, IEEE/ACM Trans. Comput. Biol. Bioinform. 1 (2004) 13–23.
[18] T. Piovesan, S. Kelk, A simple fixed parameter tractable algorithm for computing the hybridization number of two (not necessarily binary) trees, IEEE/ACM Trans. Comput. Biol. Bioinform. 10 (2013) 18–25.
[19] C. Semple, Hybridization networks, in: O. Gascuel, M. Steel (Eds.), Reconstructing Evolution: New Mathematical and Computational Advances, Oxford University Press, 2007, pp. 277–314.
[20] S.M. Soucy, J. Huang, J.P. Gogarten, Horizontal gene transfer: building the web of life, Nat. Rev. Genet. 16 (2015) 472–482.
[21] O. Westesson, G. Lunter, B. Paten, Ian Holmes, Accurate reconstruction of insertion–deletion histories by statistical phylogenetics, PLoS ONE 7 (4) (2012) e34572.
[22] Y. Wu, J. Wang, Fast computation of the exact hybridization number of two phylogenetic trees, in: International Symposium on Bioinformatics Research and Applications, Springer, 2010, pp. 203–214.

1.2 Deciding the existence of a cherry-picking sequence is hard on two trees

The following paper [DvIKL19] is also available online at the following URL: `https://doi.org/10.1016/j.dam.2019.01.031`.

Discrete Applied Mathematics 260 (2019) 131–143

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Deciding the existence of a cherry-picking sequence is hard on two trees

Janosch Döcker [a], Leo van Iersel [b], Steven Kelk [c,*], Simone Linz [d]

[a] Department of Computer Science, University of Tübingen, Germany
[b] Delft Institute of Applied Mathematics, Delft University of Technology, Van Mourik Broekmanweg 6, 2628 XE Delft, The Netherlands
[c] Department of Data Science and Knowledge Engineering (DKE), Maastricht University, The Netherlands
[d] School of Computer Science, University of Auckland, New Zealand

ARTICLE INFO

Article history:
Received 30 November 2017
Received in revised form 20 October 2018
Accepted 22 January 2019
Available online 13 February 2019

Keywords:
Phylogenetics
NP-hardness
Satisfiability
Phylogenetic networks
Elimination orders
Temporal networks

ABSTRACT

Here we show that deciding whether two rooted binary phylogenetic trees on the same set of taxa permit a *cherry-picking sequence*, a special type of elimination order on the taxa, is NP-complete. This improves on an earlier result which proved hardness for eight or more trees. Via a known equivalence between cherry-picking sequences and temporal phylogenetic networks, our result proves that it is NP-complete to determine the existence of a temporal phylogenetic network that contains topological embeddings of both trees. The hardness result also greatly strengthens previous inapproximability results for the minimum temporal-hybridization number problem. This is the optimization version of the problem where we wish to construct a temporal phylogenetic network that topologically embeds two given rooted binary phylogenetic trees and that has a minimum number of indegree-2 nodes, which represent events such as hybridization and horizontal gene transfer. We end on a positive note, pointing out that fixed parameter tractability results in this area are likely to ensure the continued relevance of the temporal phylogenetic network model.

1. Introduction

In the field of phylogenetics it is common to represent the evolution of a set of species X by a rooted phylogenetic tree; essentially a rooted, bifurcating tree whose leaves are bijectively labeled by X [21]. Driven by the realization that evolution is not always treelike there has been growing attention for the construction of phylogenetic *networks*, which generalize phylogenetic trees to directed acyclic graphs [1,11,14,22]. One well known optimization problem for phylogenetic networks is as follows: given a set of rooted phylogenetic trees $\mathcal{T}$ on the same set of taxa X, compute a phylogenetic network $N = (V, E)$ which *displays* (i.e. contains topological embeddings of) all the trees in $\mathcal{T}$, such that the *reticulation number* $|E| - (|V| - 1)$ is minimized. When N is restricted to being *binary* this is equivalent to minimizing the number of nodes of N with indegree-2. This optimization model is known as *minimum hybridization* and it has been extensively studied in the last decade (see e.g. [2,6,15,18,24]). More recently variations of minimum hybridization have been proposed which constrain the topology of N to be more biologically relevant. One such constraint is to demand that N is *temporal* [19]. Informally, a phylogenetic network N is temporal if (i) the nodes of N can be labeled with times, such that nodes of indegree-2 have contemporaneous parents, and time moves strictly forwards along treelike parts of the network; and (ii) each non-leaf vertex has a child whose

* Corresponding author.
E-mail addresses: janosch.doecker@uni-tuebingen.de (J. Döcker), L.J.J.vanIersel@tudelft.nl (L. van Iersel), steven.kelk@maastrichtuniversity.nl (S. Kelk), s.linz@auckland.ac.nz (S. Linz).

https://doi.org/10.1016/j.dam.2019.01.031
0166-218X/

indegree is 1. Property (ii) by itself is referred to as *tree-child* in the literature [5]. It has been shown that when $|\mathcal{T}| = 2$ it is NP-hard to solve the minimum temporal-hybridization number problem to optimality [13]. To establish the result, the authors proved that the problem is in fact APX-hard, which implies that for some constant $c > 1$ it is not possible in polynomial time to approximate the optimum within a factor of c, unless P = NP [20].

A more fundamental question remained, however, open: is it possible in polynomial time to determine if *any* temporal phylogenetic network exists that displays the input trees, regardless of how large $|E| - (|V| - 1)$ is [12,23]? Here we settle this question by showing that, even for $|\mathcal{T}| = 2$, it is NP-complete to determine whether such a network exists. We prove this by using the *cherry-picking* characterization of temporal phylogenetic networks introduced in [12]. There it was shown that, given an arbitrarily large set $\mathcal{T}$ of rooted binary phylogenetic trees on X, there exists a temporal phylogenetic network that displays each tree in $\mathcal{T}$ precisely if $\mathcal{T}$ has a so-called cherry-picking sequence. Informally, a *cherry-picking sequence* on $\mathcal{T}$ is an elimination order on X that deletes one element of X at a time, where at each step only elements can be deleted which are in a cherry of every tree in $\mathcal{T}$ [12]. We show here that the seminal NP-complete problem 3-SAT [17] can be reduced to the question of whether two trees permit a cherry-picking sequence. This improves upon a recent result by two of the present authors which shows that, for $|\mathcal{T}| \geq 8$, it is NP-complete to determine whether $\mathcal{T}$ has a cherry-picking sequence [8]. Our hardness result is highly non-trivial and requires extensive gadgetry; to clarify we include an explicit example of the construction after the main proof.

As we discuss in the final section of the paper, this result has quite significant negative consequences: given that the decision problem is already hard, the minimum temporal-hybridization number problem is in some sense "effectively inapproximable", even for two trees. This greatly strengthens the earlier APX-hard inapproximability result. Nevertheless, as we subsequently point out, positive fixed parameter tractability (FPT) [7] results for the minimum temporal-hybridization number problem do already exist [12] and our results emphasize the importance of further developing such algorithms, since fixed parameter tractability forms the most promising remaining avenue towards practical exact methods.

2. Preliminaries

A *rooted binary phylogenetic tree* on a set of taxa X, where $|X| \geq 2$, is a rooted, connected, directed tree with a unique *root* (a vertex of indegree-0 and outdegree-2), where the leaves (vertices with indegree-1 and outdegree-0) are bijectively labeled by X, and where all interior vertices of the tree are indegree-1 and outdegree-2. If $|X| = 1$, we consider the single isolated node labeled by the unique element of X, to also be a rooted binary phylogenetic tree. Since all phylogenetic trees considered in this paper are rooted and binary, we henceforth write *tree* for brevity, and draw no distinction between the elements of X and the leaves they label. Let T be a tree, and let $\mathcal{T} = \{T_1, T_2, \ldots, T_m\}$ be a set of trees. We use $X(T)$ to denote the taxa set of T and, similarly, we use $X(\mathcal{T})$ to denote the union of taxa sets over all elements in $\mathcal{T}$, i.e. $X(\mathcal{T}) = X(T_1) \cup X(T_2) \cup \cdots \cup X(T_m)$. Lastly, for two distinct elements x and y in X, we call $\{x, y\}$ a *cherry* of T if they have the same parent. A tree with a single cherry is referred to as a *caterpillar*.

Now, let T be a tree on X, and let $X' = \{x_1, x_2, \ldots, x_k\}$ be an arbitrary set. We write $T|X'$ to denote the tree obtained from T by taking the minimum subtree spanning the elements of X' and repeatedly suppressing all vertices with indegree-1 and outdegree-1. (If v is a vertex with indegree-1 and outdegree-1, with incident edges (u, v) and (v, w), then *suppressing* v is achieved as follows: v and its two incident edges are deleted, and an edge (u, w) is added.)

Furthermore, we also write $T[-x_1, x_2, \ldots, x_k]$ or $T[-X']$ for short to denote $T|(X - X')$. If $X \cap X' = \emptyset$, then $T|X'$ is the null tree and $T[-X']$ is T itself. For a set $\mathcal{T} = \{T_1, T_2, \ldots, T_m\}$ of trees on subsets of X, we write $\mathcal{T}|X'$ (resp. $\mathcal{T}[-X']$) when referring to the set $\{T_1|X', T_2|X', \ldots, T_m|X'\}$ (resp. $\{T_1[-X'], T_2[-X'], \ldots, T_m[-X']\}$). Lastly, a rooted binary phylogenetic tree is *pendant* in T if it can be detached from T by deleting a single edge.

2.1. Cherry-picking sequence problem on trees with the same set of taxa

We say that a taxon $x \in X$ is *in* a cherry of T if there exists some $y \neq x$ such that $\{x, y\}$ is a cherry of T or T consists of a single leaf x. If x is in a cherry of T, we say that x is *picked* (or *pruned*) from T to denote the operation of replacing T with $T[-x]$. Given a set of trees $\mathcal{T}$, all on the same set of taxa X, we say that a taxon $x \in X$ is *available (for picking)* in $\mathcal{T}$ if x is in a cherry in each tree in $\mathcal{T}$. When this is the case, we say that x is *picked* (or *pruned*) from $\mathcal{T}$ to denote the operation of replacing $\mathcal{T}$ with $\mathcal{T}[-x]$.

Let $\mathcal{T}$ be a set of trees on the same set of taxa X. A *cherry-picking sequence* is an order on X, say $(x_1, x_2, \ldots, x_{|X|})$, such that each x_i with $i \in \{1, 2, \ldots, |X|\}$ is available in $\mathcal{T}[-x_1, x_2, \ldots, x_{i-1}]$. Such a sequence is not guaranteed to exist; if it does, we say that $\mathcal{T}$ *permits* a cherry-picking sequence. It was shown in [8] that deciding whether such a sequence exists is NP-complete if $|\mathcal{T}| \geq 8$. Note that, if $|\mathcal{T}| = 1$, then $\mathcal{T}$ always has a cherry-picking sequence. To illustrate, a cherry-picking sequence for the two trees that are shown at the top of Fig. 1 is (f, g, d, b, a, c, e).

2.2. A more general cherry-picking sequence problem

Let $\mathcal{T}$ be a set of trees, and let $X = X(\mathcal{T})$. Suppose we consider the variant of the problem described in Section 2.1 in which the trees in $\mathcal{T}$ do not necessarily have the same set of taxa. In this case, some taxa may be missing from some trees. This requires us to generalize the concept of being *in* a cherry of a tree. We say that a taxon x is *in* a cherry of a tree T, if exactly one of the following conditions holds:

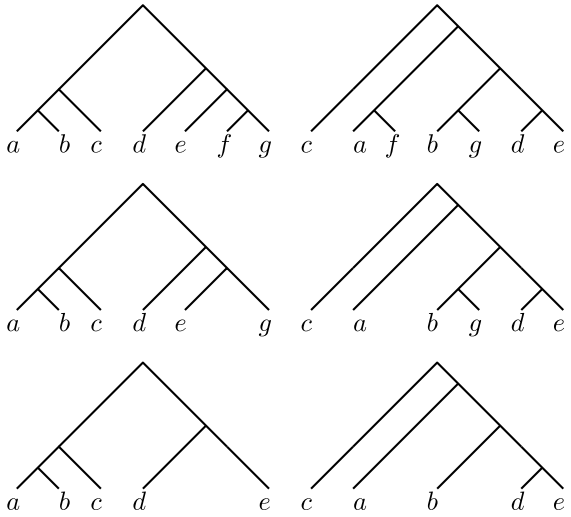


Fig. 1. A cherry-picking sequence for the two trees T and T' at the top is (f, g, d, b, a, c, e). The two trees in the middle have been obtained from T and T', respectively, by pruning f, and the two trees at the bottom have been obtained from T and T' by first pruning f and, subsequently, pruning g. While we can alternatively prune a and, subsequently, b, from T and T', note that no cherry-picking sequence exists for T and T' whose first two elements are a and b.

(1) $x \notin X(T)$ or
(2) $x \in X(T)$ and T has a cherry $\{x, y\}$, where x and y are distinct elements in $X(T)$.

(Note that, once again, this means that if x is the only taxon in T, then x is vacuously considered to be in a cherry of T.) It initially seems counter-intuitive to say, when condition 1 applies, that x is "in" a cherry of T. However, the idea behind this is that such trees do not constrain whether x can be picked; they "do not care". More formally, we say that a taxon x is *available* in $\mathcal{T}$ if it is in a cherry in each tree in $\mathcal{T}$. Similar to Section 2.1, we say that an order on X, say $(x_1, x_2 \ldots, x_{|X|})$ is a *cherry-picking sequence* of $\mathcal{T}$ if each x_i with $i \in \{1, 2, \ldots, |X|\}$ is available in $\mathcal{T}[-x_1, x_2, \ldots, x_{i-1}]$. If a tree becomes the null tree due to all its taxa being pruned away then this tree plays no further role. Moreover we note that, if all trees in $\mathcal{T}$ have the same set of taxa, then the more general definition of a cherry-picking sequence given in this subsection and that will be used throughout the rest of this paper coincides with that given in Section 2.1.

3. Main results

In this section, we establish the main result of this paper. We start with two lemmas.

Lemma 1. *Let $\mathcal{T}$ be a set of m trees on not necessarily the same set of taxa. Then we can construct in polynomial time a set $\mathcal{T}'$ of m trees all on the same set of taxa, such that $\mathcal{T}$ has a cherry-picking sequence if and only if $\mathcal{T}'$ does.*

Proof. Let $X = X(\mathcal{T})$, and let $Y = \{y_1, y_2, \ldots\}$ be the set of taxa that are missing from at least one input tree. Let $Y' = \{y'_1, y'_2, \ldots\}$ be a disjoint copy of this set. Every modified tree will have taxon set $X \cup Y' \cup \{\rho\}$. The idea is as follows. Let $T_{Y'}$ be an arbitrary rooted binary tree on Y'. For each input tree T_i, we start by joining T_i and ρ beneath a root, and then join this new tree and $T_{Y'}$ together beneath a root. Next, for each $y_j \in Y$ that is missing from T_i, we add y_j by subdividing the edge that feeds into y'_j and attaching y_j there (so y_j and y'_j become siblings). For an example, see Fig. 2. We call the set of trees constructed in this way $\mathcal{T}'$. The high-level idea is that if a tree T_i does not contain some taxon x, we attach x just above x' and thus ensure that, trivially, x is in a cherry in that tree (i.e. together with x'). So T_i does "not care" about x and will not obstruct it from being pruned.

First, assume that $\mathcal{T}$ has a cherry-picking sequence σ. (We show that $\mathcal{T}'$ has a cherry-picking sequence.) We start by applying exactly the same sequence of pruning operations to $\mathcal{T}'$. These picking operations will always be possible because, if a taxon $y \in Y$ is missing from a tree $T_i \in \mathcal{T}$, it will be in a cherry together with y' in the corresponding tree of $\mathcal{T}'$. After doing this, all the trees will be isomorphic and have the same set of taxa: $Y' \cup \{\rho\}$. At this point these remaining taxa can be pruned in bottom-up fashion (since two isomorphic trees always have a cherry-picking sequence). Hence $\mathcal{T}'$ has a cherry-picking sequence. Note that the taxon ρ is included to ensure that if, during σ, a tree T_i has been pruned down to a single taxon, this taxon can still be pruned in the corresponding tree of $\mathcal{T}'$ (because it is sibling to ρ).

In the other direction, let σ' be a cherry-picking sequence for $\mathcal{T}'$. Let σ be the sequence obtained by deleting all taxa from σ' that are not in X. Let x be an arbitrary element of X and let i be the position of x in σ'. Let $\ell'_1, \ell'_2, \ldots, \ell'_{i-1}$ be the prefix of σ' that has been pruned from $\mathcal{T}'$ prior to x, and let $\ell_1, \ell_2, \ldots, \ell_j$ (where $j \leq i - 1$) be the prefix of σ that has been pruned prior to x. We claim that, if x is available in $\mathcal{T}'[-\ell'_1, \ell'_2, \ldots, \ell'_{i-1}]$, then it is also available in $\mathcal{T}[-\ell_1, \ell_2, \ldots, \ell_j]$. To see this, let T be an arbitrary tree in $\mathcal{T}[-\ell_1, \ell_2, \ldots, \ell_j]$. If $x \notin X(T)$, then (by definition) x is in a cherry of T. If x is the only

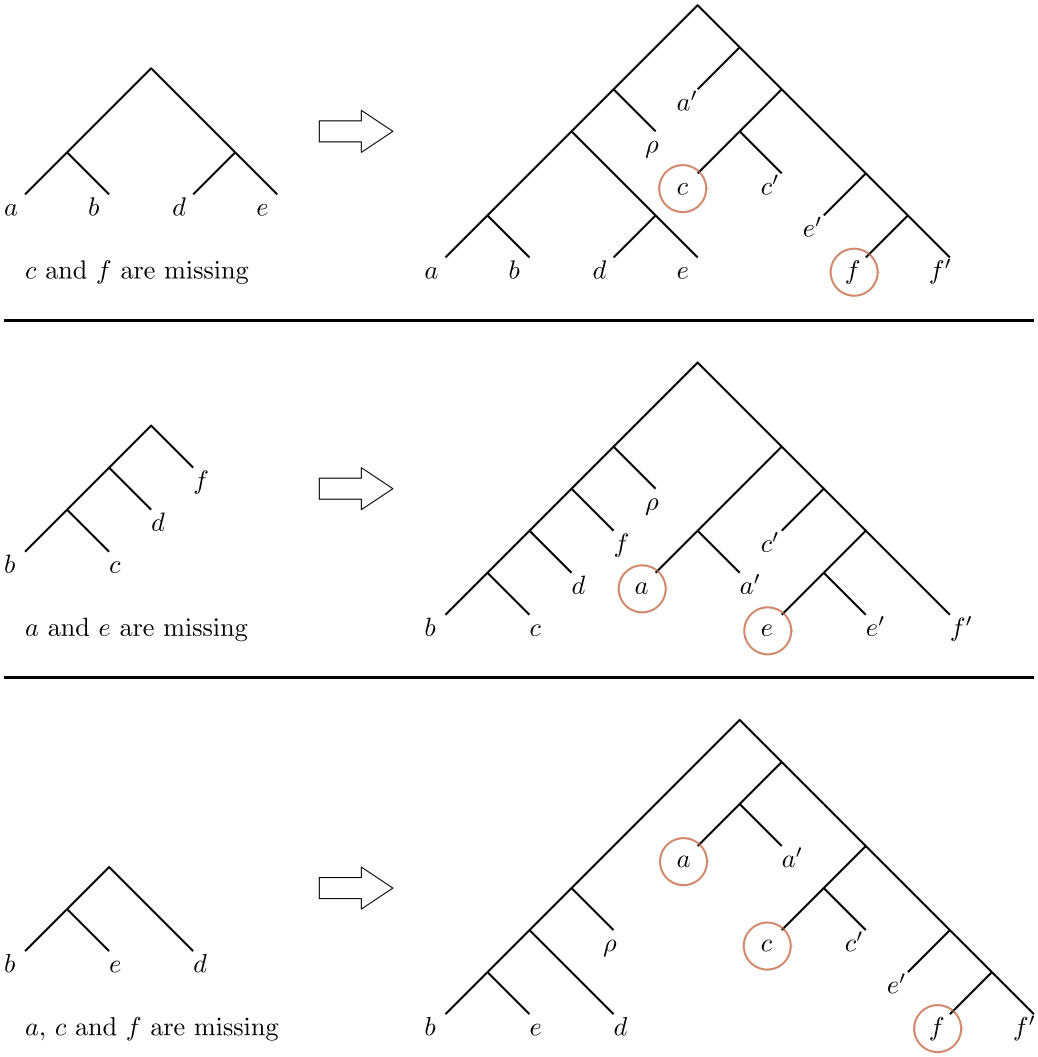


Fig. 2. The construction described in Lemma 1. Here Y, the set of taxa missing from at least one tree, is $\{a, c, e, f\}$. In each modified tree the artificially added members of Y are circled; note that they are always in cherries. A cherry-picking sequence for the original trees is e, b, c, d, a, f. A corresponding sequence for the modified trees is $e, b, c, d, a, f, f', e', c', a', \rho$.

taxon in T, then it is (also by definition) in a cherry. So the only case remaining is that $x \in X(T)$ and $|X(T)| \geq 2$. Let T' be the tree from $\mathcal{T}'[-\ell'_1, \ell'_2, \ldots, \ell'_{i-1}]$ that corresponds to T. The critical observation here is that, by construction, T occurs as a pendant subtree of T'. So if x was not in a cherry of T, then x would not be in a cherry of T' which gives a contradiction to the assumption that $\mathcal{T}'$ has a cherry-picking sequence. Hence, x is in a cherry of T. Due to the arbitrary choice of x and T, it follows that σ is a cherry-picking sequence for $\mathcal{T}$.

It remains to show that the reduction is polynomial time. Observe that, depending on the instance, the size of $\mathcal{T}$ can be dominated by $|X|$ or m. Each of the m trees in $\mathcal{T}'$ contains $|X| + |Y| + 1$ taxa, where $|Y| \leq |X|$, and the transformation itself involves straightforward operations, so overall the reduction takes poly$(|X|, m)$ time. □

Let $\mathcal{T}$ be a set of rooted binary trees, and let T_i and T_j be two trees in $\mathcal{T}$ such that $X(T_i) \cap X(T_j) = \emptyset$. Furthermore, let ρ_i and ρ_j be the root vertex of T_i and T_j, respectively. Obtain a new tree from T_i and T_j in the following way.

(1) Create a new vertex ρ and add new edges $e = (\rho, \rho_i)$ and $e' = (\rho, \rho_j)$.
(2) Subdivide e (resp. e') with a new vertex v (resp. v') and add a new edge (v, x) (resp. (v', y)), where x and y are two new taxa such that $\{x, y\} \cap X(\mathcal{T}) = \emptyset$.

We call the resulting rooted binary tree the *compound tree* of T_i and T_j. To illustrate, Fig. 3 depicts the compound tree of T_i and T_j.

The next lemma shows that, for a set $\mathcal{T}$ of rooted binary trees, the replacement of two trees in $\mathcal{T}$ with their compound tree preserves the existence and non-existence of a cherry-picking sequences for $\mathcal{T}$.

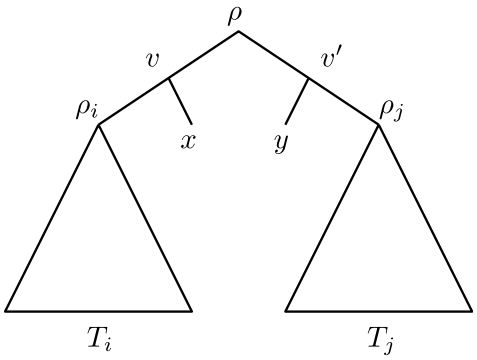


Fig. 3. The compound tree of two rooted binary trees T_i and T_j. The taxon x (resp. y) simply ensures that the last taxon pruned away in the T_i (resp. T_j) part is in a cherry with x (resp. y).

Lemma 2. *Let $\mathcal{T}$ be a set of rooted binary trees, and let T_i and T_j be two trees in $\mathcal{T}$ such that $X(T_i) \cap X(T_j) = \emptyset$. Let $T_{i,j}$ be the compound tree of T_i and T_j. Then $\mathcal{T}$ has a cherry-picking sequence if and only if $(\mathcal{T} - \{T_i, T_j\}) \cup \{T_{i,j}\}$ has a cherry-picking sequence.*

Proof. To ease reading, let $\mathcal{T}' = (\mathcal{T} - \{T_i, T_j\}) \cup \{T_{i,j}\}$. Furthermore, let $|X(\mathcal{T})| = n$, and let x and y be the unique two taxa in $X(T_{i,j})$ that do not label a leaf in T_i or T_j.

Suppose that $\sigma = (\ell_1, \ell_2, \ldots, \ell_n)$ is a cherry-picking sequence for $\mathcal{T}$. Let i' be the maximum index of an element in σ such that $\ell_{i'} \in X(T_i)$ and, similarly, let j' be the maximum index of an element in σ such that $\ell_{j'} \in X(T_j)$. Then $\mathcal{T}[-\ell_1, \ell_2, \ldots \ell_{i'-1}]$ contains a tree that is a single vertex labeled $\ell_{i'}$ and $\mathcal{T}[-\ell_1, \ell_2, \ldots \ell_{j'-1}]$ contains a tree that is a single vertex labeled $\ell_{j'}$. Moreover, by the construction of $T_{i,j}$, the set $\mathcal{T}'[-\ell_1, \ell_2, \ldots \ell_{i'-1}]$ contains a tree with cherry $\{\ell_{i'}, x\}$ and the set $\mathcal{T}'[-\ell_1, \ell_2, \ldots \ell_{j'-1}]$ contains a tree with cherry $\{\ell_{j'}, y\}$. Since T_i and T_j are pendant subtrees in $T_{i,j}$ and σ is a cherry-picking sequence for $\mathcal{T}$, it now follows that

$$(\ell_1, \ell_2, \ldots, \ell_n, x, y)$$

is a cherry-picking sequence for $\mathcal{T}'$.

Conversely, suppose that $\sigma' = (\ell_1, \ell_2, \ldots, \ell_{n+2})$ is a cherry-picking sequence for $\mathcal{T}'$. Let $\{\ell_{i'}, \ell_{j'}\} = \{x, y\}$. Without loss of generality, we may assume that $i' < j'$. Then, as x and y are only contained in the leaf set of $T_{i,j}$, it is straightforward to check that

$$(\ell_1, \ell_2, \ldots, \ell_{i'-1}, \ell_{i'+1}, \ell_{i'+2}, \ldots, \ell_{j'-1}, \ell_{j'+1}, \ell_{j'+2}, \ldots, \ell_{n+2})$$

is a cherry-picking sequence for $\mathcal{T}$. □

Now we establish the main result of this paper.

Theorem 1. *It is* NP*-complete to decide if two rooted binary phylogenetic trees T and T' on X have a cherry-picking sequence.*

Proof. Given an order $\sigma = (x_1, x_2, \ldots, x_{|X|})$ on X, we can decide in polynomial time if, for each $i \in \{1, 2, \ldots, |X|\}$, x_i is in a cherry in $T[-x_1, x_2, \ldots, x_{i-1}]$ and $T'[-x_1, x_2, \ldots, x_{i-1}]$. Hence, the problem of deciding if T and T' have a cherry-picking sequence is in NP. To establish the theorem, we use a reduction from 3-Sat. This is the variant of Satisfiability where each clause contains *exactly* three literals, and the logical expression is in conjunctive normal form, i.e.,

$$\bigwedge_{i=1}^{m} C_i = \bigwedge_{i=1}^{m} (l_{i,1} \vee l_{i,2} \vee l_{i,3}),$$

where $l_{i,j} \in \{v^{(k)}, \neg v^{(k)} \mid 1 \leq k \leq n\}$. The corresponding set of variables is denoted with

$$V := \{v^{(1)}, v^{(2)}, \ldots, v^{(n)}\}.$$

We reduce from the NP-complete version of 3-Sat in which no variable occurs more than once in a given clause. Such restricted instances can easily be obtained by a standard transformation as described in [10]. In the remainder of this proof, n and m refer to the number of variables and clauses in a restricted 3-Sat instance, respectively.

Now, given an instance I of 3-Sat, we first construct a set $\mathcal{T}$ of $3n + 5m + 2$ trees with overlapping taxa sets and show that I has a satisfying truth assignment if and only if $\mathcal{T}$ has a cherry-picking sequence. We then repeatedly apply Lemma 2 in order to replace $\mathcal{T}$ with two trees and, finally, apply Lemma 1 to complete the proof of this theorem.

We start by describing the construction of $\mathcal{T}$ that makes use of the introduction of a set $\{b_1, b_2, \ldots, b_{4n+3m}, b_X, b_Y, b_Z\}$ of blocking taxa. As we will see later, each such taxon can only be pruned from $\mathcal{T}$ after certain other taxa have been

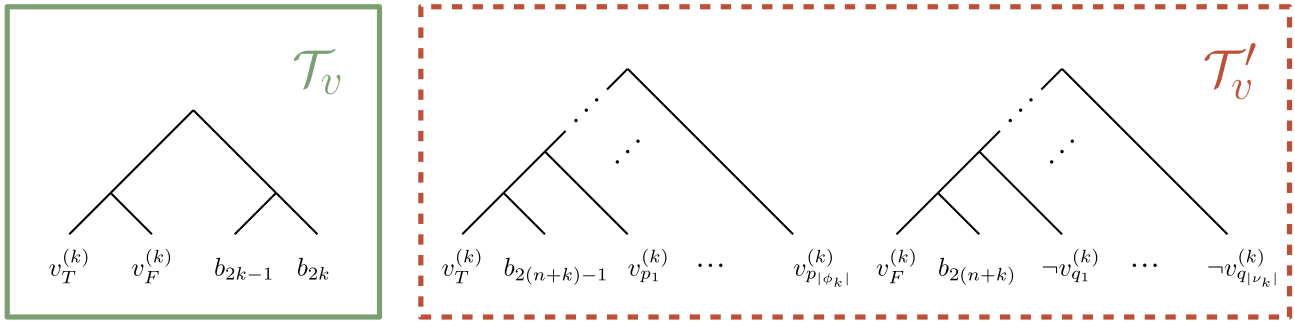


Fig. 4. Each variable $v^{(k)}$, is represented by a single tree in $\mathcal{T}_v$ and two trees in $\mathcal{T}'_v$.

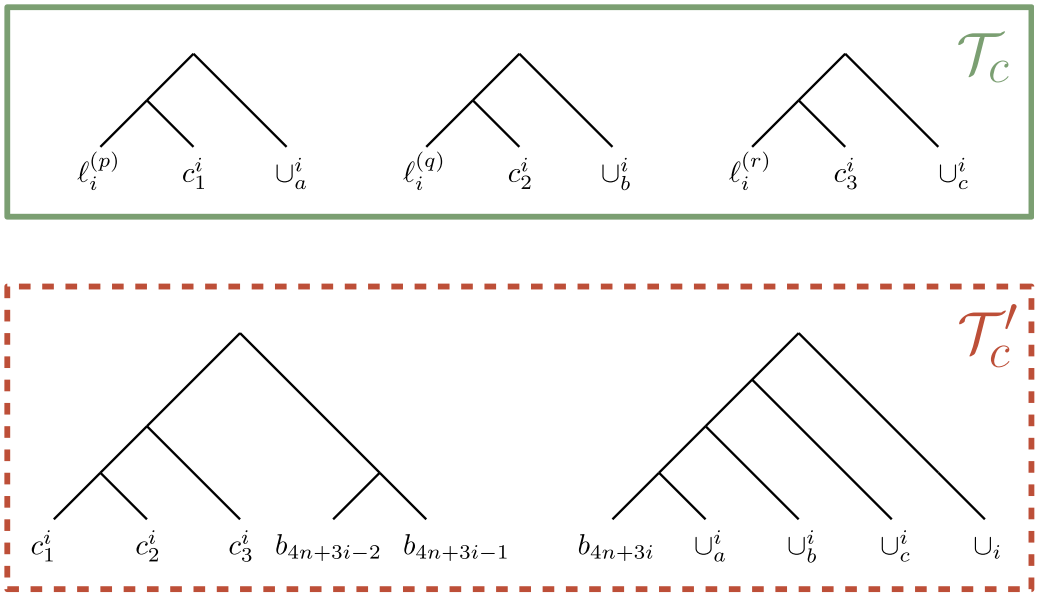


Fig. 5. Each clause C_i is represented by three trees in $\mathcal{T}_c$ and two trees in $\mathcal{T}'_c$.

pruned first and so the main function of the blocking taxa is to be unavailable for pruning which in turn constraints the number of possibilities to construct a cherry-picking sequence from $\mathcal{T}$. An explicit example of the construction of $\mathcal{T}$ is given subsequently to this proof.

Variable gadget. We construct two sets $\mathcal{T}_v$ and $\mathcal{T}'_v$ of trees. Each variable $v^{(k)}$ with $k \in \{1, 2, \ldots, n\}$ adds one tree on four taxa to $\mathcal{T}_v$ which is the tree shown in the solid box of Fig. 4. Each such tree has two blocking taxa and, intuitively, encodes whether $v^{(k)}$ is set to be true or false, depending on whether $v_T^{(k)}$ or $v_F^{(k)}$ is pruned first. Moreover, each variable $v^{(k)}$ adds two caterpillars to $\mathcal{T}'_v$. Relative to a fixed $v^{(k)}$, the precise construction of these caterpillars is based on the definition of two particular tuples. Let $\phi_k := (p_1, p_2, \ldots, p_{|\phi_k|})$ (resp. $\nu_k := (q_1, q_2, \ldots, q_{|\nu_k|})$) be the indices, in ascending order, of all the clauses in which $v^{(k)}$ appears unnegated (resp. negated). Since no clause contains any variable more than once, the elements in ϕ_k (resp. ν_k) are pairwise distinct.

Now the taxon set of one caterpillar contains $v_T^{(k)}$, a new blocking taxon and, for each element p_j in ϕ_k, a new taxon $v_{p_j}^{(k)}$, while the taxon set of the other caterpillar contains $v_F^{(k)}$, a new blocking taxon and, for each element q_j in ν_k, a new taxon $\neg v_{q_j}^{(k)}$. The precise ordering of the leaves in both caterpillars is shown in the dashed box of Fig. 4. It is easily checked that $|X(\mathcal{T}_v)| = 4n$ and, since each clause contains precisely three distinct variables, $|X(\mathcal{T}'_v)| = 4n + 3m$. Noting that the taxa sets of the trees in $X(\mathcal{T}_v)$ and $X(\mathcal{T}'_v)$ only overlap in $v_T^{(k)}$ and $v_F^{(k)}$, we have

$$|X(\mathcal{T}_v \cup \mathcal{T}'_v)| = 4n + 4n + 3m - 2n = 6n + 3m \tag{1}$$

distinct taxa over all trees in $\mathcal{T}_v$ and $\mathcal{T}'_v$.

Clause gadget. We construct two sets $\mathcal{T}_c$ and $\mathcal{T}'_c$ of trees. For each $i \in \{1, 2, \ldots, m\}$, consider the clause $C_i = \ell^{(p)} \vee \ell^{(q)} \vee \ell^{(r)}$, where each $k \in \{p, q, r\}$ is an element in $\{1, 2, \ldots, n\}$ with $\ell^{(k)} \in \{v^{(k)}, \neg v^{(k)}\}$. Relative to C_i, we add three three-taxon trees to $\mathcal{T}_c$ which are shown in the solid box of Fig. 5. The first such tree has taxon set $\{\ell_i^{(p)}, c_1^i, \cup_a^i\}$ where $\ell_i^{(p)}$ is an element in $\{v_i^{(p)}, \neg v_i^{(p)}\}$. Note that $\ell_i^{(p)}$ labels a leaf of a tree in $\mathcal{T}'_v$ while the other two taxa do not label a leaf of a tree in $\mathcal{T}_v$ or $\mathcal{T}'_v$. The other two trees in $\mathcal{T}_c$ are constructed in an analogous way. Furthermore, for each C_i, we add two five-taxon trees to $\mathcal{T}'_c$ which

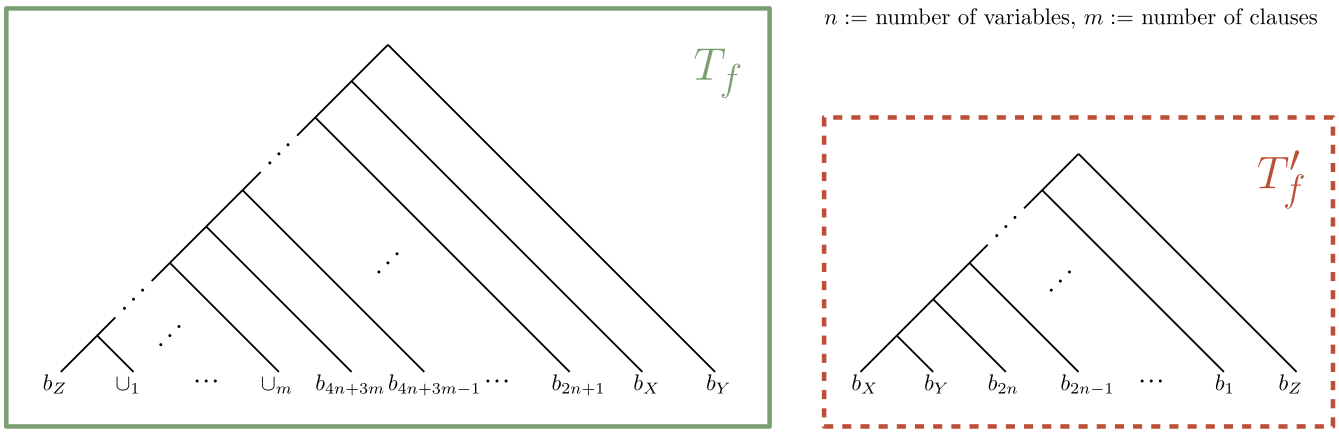


Fig. 6. The two trees T_f and T'_f in the construction of $\mathcal{T}$ from I.

are shown in the dashed box of Fig. 5. The taxa set of the first tree contains two new blocking taxa and the three previously encountered elements $\{c_1^i, c_2^i, c_3^i\}$, while the second tree contains one new blocking taxon, the new taxon $\cup_i$, and the three previously encountered elements $\{\cup_a^i, \cup_b^i, \cup_c^i\}$. Similar to the variable gadgets, we now count the number of taxa in trees in $\mathcal{T}_c$ and $\mathcal{T}'_c$. As no two trees in $\mathcal{T}_c$ or $\mathcal{T}'_c$ share a taxon, we have $|X(\mathcal{T}_c)| = 9m$ and $|X(\mathcal{T}'_c)| = 10m$. Moreover, since all taxa of trees in $\mathcal{T}'_c$, except for the blocking taxa and elements in $\{\cup_1, \cup_2, \ldots, \cup_m\}$, are also taxa of trees in $\mathcal{T}_c$, we have

$$|X(\mathcal{T}_c \cup \mathcal{T}'_c)| = 9m + 10m - 6m = 13m. \tag{2}$$

Formula gadget. We complete the construction of $\mathcal{T}$ by constructing two caterpillars T_f and T'_f which are shown in the solid and dashed box of Fig. 6, and define

$$\mathcal{T} = \mathcal{T}_v \cup \mathcal{T}'_v \cup \mathcal{T}_c \cup \mathcal{T}'_c \cup \{T_f, T'_f\}.$$

Summarizing the construction, we have $|\mathcal{T}| = 3n + 5m + 2$. Moreover, by construction and Eqs. (1)–(2), it follows that $|X((\mathcal{T}_v \cup \mathcal{T}'_v) \cap (\mathcal{T}_c \cup \mathcal{T}'_c))| = 3m$. Now, since the three taxa b_X, b_Y, and b_Z, which are common to T_f and T'_f, are the only taxa of these two trees that are not contained in the taxa set of any other constructed tree, we have

$$|X(\mathcal{T})| = 6n + 3m + 13m - 3m + 3 = 6n + 13m + 3. \tag{3}$$

We next prove the following claim:

Claim 1. *I is satisfiable if and only if $\mathcal{T}$ has a cherry-picking sequence.*

First, suppose that I is satisfiable. Let $\beta : V \to \{T, F\}$ be a truth assignment for V such that each clause is satisfied. We next describe a sequence of pruning operation. Noting that each taxon in $X(\mathcal{T})$ is contained in the taxa sets of exactly two trees in $\mathcal{T}$ (a fact that we freely use throughout the rest of this proof), it is straightforward to verify that this sequence implies a cherry-picking sequence for $\mathcal{T}$.

Part 1: variable gadgets. For each variable $v^{(k)}$ with $k \in \{1, 2, \ldots, n\}$ do the following. If $\beta(v^{(k)}) = T$ prune taxon $v_T^{(k)}$ from the two trees in $\mathcal{T}_v \cup \mathcal{T}'_v$ whose taxa sets contain $v_T^{(k)}$. On the other hand, if $\beta(v^{(k)}) = F$ prune taxon $v_F^{(k)}$ from the two trees in $\mathcal{T}_v \cup \mathcal{T}'_v$ whose taxa sets contain $v_F^{(k)}$. Taken together, these pruning steps delete a single leaf of each tree in $\mathcal{T}_v$ and a single leaf of half of the trees in $\mathcal{T}'_v$.

Part 2: clause gadgets. Consider the set of trees resulting from the pruning described in Part 1. For each $C_i = \ell^{(p)} \vee \ell^{(q)} \vee \ell^{(r)}$ with $i \in \{1, 2, \ldots, m\}$, let L_i be a subset of $\{p, q, r\}$ such that $|L_i| = 2$ and, if $\ell_i^{(k)}$ is not satisfied by β, then $k \in L_i$. Setting $i = 1$, process the three literals in C_i from left to right in the following way.

(1) If $\ell_i^{(k)}$ is satisfied by β, prune $\ell_i^{(k)}$ from the tree in $\mathcal{T}_c$ whose taxa set contains $\ell_i^{(k)}$ and, noting that $\ell_i^{(k)} \in \{v_i^{(k)}, \neg v_i^{(k)}\}$, prune $\ell_i^{(k)}$ from the tree in $\mathcal{T}'_v$ whose taxa set contains $\ell_i^{(k)}$.
(2) If $k \in L_i$, prune c_s^i, where $s = 1$ if $k = p$, $s = 2$ if $k = q$, and $s = 3$ if $k = r$, from the two trees in $\mathcal{T}_c \cup \mathcal{T}'_c$ whose taxa sets contain c_s^i.
(3) Prune $\cup_t^i$, where $t = a$ if $k = p$, $t = b$ if $k = q$, and $t = c$ if $k = r$, from the two trees in $\mathcal{T}_c \cup \mathcal{T}'_c$ whose taxa sets contain $\cup_t^i$.

Now prune $\cup_i$ from the tree in $\mathcal{T}'_c$ whose taxa set contains $\cup_i$, and prune $\cup_i$ from T_f. If $i < m$, increment i by one and repeat this process with the next clause. Intuitively, by definition of L_i, the above process prunes exactly two elements in $\{c_1^i, c_2^i, c_3^i\}$. Since each clause is satisfied by β, this guarantees that we can prune each element in $\{\cup_a^i, \cup_b^i, \cup_c^i\}$ and, subsequently $\cup_i$.

Part 3: formula gadget and remaining taxa. Consider the set of trees resulting from the pruning described in Part 2. We prune the remaining taxa as follows.

(1) In order, prune each of

$$b_{4n+3m},\ b_{4n+3m-1},\ b_{4n+3m-2},\ \ldots,\ b_{4n+3i},\ b_{4n+3i-1},\ b_{4n+3i-2},\ \ldots,\ b_{4n+3},\ b_{4n+2},\ b_{4n+1}$$

from a tree $\mathcal{T}_c'$ whose taxa set contains the respective blocking taxa and from T_f. After all taxa have been pruned, each tree in $\mathcal{T}_c'$ is either the null tree or consists of a single vertex labeled c_s^i for some $s \in \{1, 2, 3\}$.

(2) For each $i \in \{1, 2, \ldots, m\}$, prune the unique taxon c_s^i with $s \in \{1, 2, 3\}$ that has not been pruned in Part 2 from two trees in $\mathcal{T}_c \cup \mathcal{T}_c'$. Now, each tree in $\mathcal{T}_c \cup \mathcal{T}_c'$ that is not the null tree consists of a single vertex labeled $\ell_i^{(k)}$ for some $i \in \{1, 2, \ldots, m\}$ and $k \in \{1, 2, \ldots, n\}$.

(3) For each $k \in \{1, 2, \ldots, n\}$, note that one of $\{b_{2(n+k)-1}, b_{2(n+k)}\}$ labels a leaf of a cherry in a tree in $\mathcal{T}_v'$ while the other labels the leaf of a tree in $\mathcal{T}_v'$ that consists of a single vertex. In order, prune each of

$$b_{4n},\ b_{4n-1},\ \ldots,\ b_{2(n+k)},\ b_{2(n+k)-1},\ \ldots,\ b_{2n+2},\ b_{2n+1}$$

from the tree in $\mathcal{T}_v'$ whose taxa set contains the respective blocking taxa and from T_f.

(4) In order, prune b_X and b_Y from T_f and T_f'.

(5) Consider the remaining trees in $\mathcal{T}_v$ and observe that each such tree consists of exactly three leaves, two of which are blocking taxa that form a cherry. In order, prune each of

$$b_{2n},\ b_{2n-1},\ \ldots,\ b_{2k},\ b_{2k-1},\ \ldots,\ b_2,\ b_1$$

from T_f' and the tree in $\mathcal{T}_v$ whose taxa set contains the respective blocking taxon.

(6) For each $k \in \{1, 2, \ldots, n\}$, let $v_X^{(k)}$ be the unique element in $\{v_T^{(k)}, v_F^{(k)}\}$ that has not been pruned in Part 1. Prune $v_X^{(k)}$ from the two trees in $\mathcal{T}_v \cup \mathcal{T}_v'$ whose taxa sets contain $v_X^{(k)}$.

(7) For each $i \in \{1, 2, \ldots, m\}$ in increasing order, consider each literal $\ell_i^{(k)}$ in $C_i = \ell^{(p)} \vee \ell^{(q)} \vee \ell^{(r)}$ with $k \in \{p, q, r\}$ that is not satisfied by β. By processing such literals from left to right in C_i, prune $\ell_i^{(k)}$ from the two trees in $\mathcal{T}_v' \cup \mathcal{T}_c$ whose taxa sets contain $\ell_i^{(k)}$. It is easily seen that the corresponding tree in $\mathcal{T}_v'$ either consists of a single vertex or contains a cherry with a leaf labeled $\ell_i^{(k)}$.

(8) Prune b_Z from T_f and T_f'.

Now, relative to the elements in $X(\mathcal{T})$, we prune $2n$ elements in Parts 1 and 3.6, all $4m$ elements in

$$\{\cup_1, \cup_a^i, \cup_b^i, \cup_c^i, \ldots, \cup_m, \cup_a^m, \cup_b^m, \cup_c^m\}$$

in Part 2, and all $4n + 3m + 3$ blocking taxa in Parts 3.1, 3.3, 3.4, 3.5, and 3.8. Additionally, in Parts 2.1 and 3.7 we prune $3m$ taxa, and in Parts 2.2 and 3.2, we prune again $3m$ taxa. Summing up, we prune

$$6n + 13m + 3$$

taxa, which is equal to the number of elements in $X(\mathcal{T})$.

Second, suppose that $\mathcal{T}$ has a cherry-picking sequence $\sigma = (x_1, x_2, \ldots, x_{|\sigma|})$. We write $x_i \prec x_j$ if and only if $i < j$ and $x_i \succ x_j$ if and only if $i > j$. Further, let

$$M := \{1, 2, \ldots, m\},\ N := \{1, 2, \ldots, n\},\ B := \{b_1, b_2, \ldots, b_{4n+3m}, b_X, b_Y, b_Z\}.$$

We define a truth assignment $\beta\colon V \to \{T, F\}$ as follows

$$\beta\left(v^{(k)}\right) = \begin{cases} T & \text{if } \exists i \in M\colon v_T^{(k)} \prec \cup_i, \\ F & \text{else.} \end{cases}$$

In order to show that β satisfies each clause of I, we establish four necessary conditions that σ fulfills by construction.

(1) All taxa in $\{\cup_1, \cup_2, \ldots, \cup_m\}$ are pruned earlier than any blocking taxon:

$$\forall i \in M\ \forall b \in B\colon \cup_i \prec b. \tag{4}$$

Argument: Observe that the arrangement of b_X, b_Y, b_Z in T_f and T_f' implies that all taxa in $\{\cup_1, \cup_2, \ldots, \cup_m\}$ are pruned prior to any blocking taxon. Furthermore, we cannot prune any taxon in T_f' until we have pruned all taxa from T_f except for b_X, b_Y, and b_Z. We will freely use Condition 1 throughout the remainder of this proof.

(2) Let $C_i = \ell^{(p)} \vee \ell^{(q)} \vee \ell^{(r)}$ be a clause of I. At least one taxon in $\{\ell_i^{(p)}, \ell_i^{(q)}, \ell_i^{(r)}\}$ is pruned earlier than $\cup_i$. Stated more formally:

$$\forall i \in M\ \exists \ell_i^{(s_i)} \in \left\{\ell_i^{(p_i)}, \ell_i^{(q_i)}, \ell_i^{(r_i)}\right\}\colon \ell_i^{(s_i)} \prec \cup_i. \tag{5}$$

Argument: Consider the five trees in $\mathcal{T}_c \cup \mathcal{T}_c'$ representing C_i (see Fig. 5). In order to prune $\cup_i$, we have to prune all taxa in $\{\cup_a^i, \cup_b^i, \cup_c^i\}$ first. Since we can prune at most two taxa in $\{c_1^i, c_2^i, c_3^i\}$ prior to an element in $\{b_{4n+3i-2}, b_{4n+3i-1}\}$, pruning all taxa in $\{\cup_a^i, \cup_b^i, \cup_c^i\}$ is only possible if at least one taxon in $\{\ell_i^{(p)}, \ell_i^{(q)}, \ell_i^{(r)}\}$ has been pruned previously.

(3) Let $v^{(k)} \in V$ be any variable of I. Recall the definition of the tuples ϕ_k and ν_k that is used in the construction of the variable gadget. If there exists a $v_i^{(k)}$ with $v_i^{(k)} \prec \cup_i$ for some $i \in \phi_k$, then $v_T^{(k)}$ is also pruned earlier than $\cup_i$. Stated formally:

$$\forall k \in N\ \forall i \in \phi_k \colon \left(v_i^{(k)} \prec \cup_i \implies v_T^{(k)} \prec \cup_i\right). \tag{6}$$

Argument: Consider a variable $v^{(k)} \in V$ such that $v_i^{(k)} \prec \cup_i$ for some $i \in \phi_k$. Since there is no blocking taxon $b \in B$ with $b \prec \cup_i$, we have $\cup_i \prec b_{2(n+k)-1}$. Thus, $v_T^{(k)}$ is pruned from the associated caterpillar in $\mathcal{T}_v'$ that contains $v_i^{(k)}$ such that $v_T^{(k)} \prec v_i^{(k)} \prec \cup_i$ (see Fig. 4).
The following can be shown analogously. If there exists a $\neg v_i^{(k)} \prec \cup_i$ for some $i \in \nu_k$, then $v_F^{(k)}$ is also pruned earlier than $\cup_i$. Stated formally:

$$\forall k \in N\ \forall i \in \nu_k \colon \left(\neg v_i^{(k)} \prec \cup_i \implies v_F^{(k)} \prec \cup_i\right). \tag{7}$$

(4) Let $v^{(k)} \in V$ be any variable of I. If $v_T^{(k)}$ is pruned earlier than some taxon in $\{\cup_1, \cup_2, \ldots, \cup_m\}$, then $v_F^{(k)}$ is pruned later than all taxa in $\{\cup_1, \cup_2, \ldots, \cup_m\}$, i.e.,

$$\forall k \in N \colon \left(\left(\exists i \in M \colon v_T^{(k)} \prec \cup_i\right) \implies \left(\forall i \in M \colon v_F^{(k)} \succ \cup_i\right)\right). \tag{8}$$

Argument: Consider a variable $v^{(k)} \in V$ such that $v_T^{(k)} \prec \cup_i$ for some $i \in M$. Assume towards a contradiction that there is some $j \in M$ such that $v_F^{(k)} \prec \cup_j$. Then, one of the two blocking taxa b_{2k-1} and b_{2k} is pruned prior to $v_F^{(k)}$ (see Fig. 4). But this is not possible since there is no blocking taxon $b \in B$ with $b \prec \cup_j$.
As an immediate consequence of statement (8), we get the analogous statement for $v_F^{(k)}$, i.e.,

$$\forall k \in N \colon \left(\left(\exists i \in M \colon v_F^{(k)} \prec \cup_i\right) \implies \left(\forall i \in M \colon v_T^{(k)} \succ \cup_i\right)\right). \tag{9}$$

Now, we show that β indeed satisfies each clause of I. For each clause $C_i = \ell^{(p)} \vee \ell^{(q)} \vee \ell^{(r)}$, we have $\ell_i^{(s)} \prec \cup_i$ for some $\ell_i^{(s)} \in \left\{\ell_i^{(p)}, \ell_i^{(q)}, \ell_i^{(r)}\right\}$ (Condition 2). Since $\ell_i^{(s_i)} \in \left\{v_i^{(k)}, \neg v_i^{(k)}\right\}$ for some $k \in N$, we have $v_T^{(k)} \prec \cup_i$ if $\ell_i^{(s_i)} = v_i^{(k)}$ and $v_F^{(k)} \prec \cup_i$ if $\ell_i^{(s_i)} = \neg v_i^{(k)}$ (Condition 3). Hence, by setting $\beta(v^{(k)}) = T$ if $v_T^{(k)} \prec \cup_i$ and $\beta(v^{(k)}) = F$ if $v_F^{(k)} \prec \cup_i$, we satisfy at least one literal of each clause. Note that we can assign arbitrary truth values to variables $v^{(k)}$ with $v_T^{(k)} \succ \cup_i$ and $v_F^{(k)} \succ \cup_i$ for all $i \in M$. Here, we choose to set all these variables to F. The truth assignment β is consistent, since at least one taxon in $\left\{v_T^{(k)}, v_F^{(k)}\right\}$ is pruned later than all taxa in $\{\cup_1, \cup_2, \ldots, \cup_m\}$ (Condition 4). Hence, the truth assignment β is consistent and satisfies each clause of I.

Folding into two trees on the same set of taxa. The trees in $\mathcal{T}_v \cup \mathcal{T}_c \cup \{T_f\}$ and, similarly, the trees in $\mathcal{T}_v' \cup \mathcal{T}_c' \cup \{T_f'\}$ (see Fig. 4, 5, and 6) have mutually disjoint taxa sets. Hence, by $n + 3m$ applications of Lemma 2, we can construct a compound tree S for all trees in $\mathcal{T}_v \cup \mathcal{T}_c \cup \{T_f\}$ and, by $2n + 2m$ applications of Lemma 2, we can construct a compound tree S' for all trees in $\mathcal{T}_v' \cup \mathcal{T}_c' \cup \{T_f'\}$ such that $\mathcal{T}$ has a cherry-picking sequence if and only if S and S' have a cherry-picking sequence. Lastly, by applying Lemma 1, we obtain two trees T and T' from S and S', respectively, such that $X(T) = X(T')$, and S and S' have a cherry-picking sequence if and only if T and T' have such a sequence. It now follows that I is satisfiable if and only if T and T' have a cherry-picking sequence.

Number of taxa in the final instance. It remains to show that T and T' can be constructed in polynomial time. By Eq. (3), recall that $|X(\mathcal{T})| = 6n + 13m + 3$. Now, since we apply Lemma 2 a total of $3n + 5m$ times and each application introduces two new taxa, we have

$$|X(\{S,\ S'\})| = 6n + 13m + 3 + 2(3n + 5m) = 12n + 23m + 3.$$

Observe that each taxon in $X(\mathcal{T})$ labels a leaf of a unique tree in $\mathcal{T}_v \cup \mathcal{T}_c \cup \{T_f\}$ and a leaf of a unique tree in $\mathcal{T}_v' \cup \mathcal{T}_c' \cup \{T_f'\}$. It therefore follows that each taxon that is contained in exactly one of $X(S)$ and $X(S')$ has been introduced by an application of Lemma 2. Conversely, each application of this lemma introduces two taxa that are both contained in exactly one of $X(S)$ and $X(S')$. Hence, recalling that in obtaining T and T' from S and S', respectively, an additional leaf labeled ρ is introduced (see the third sentence in the proof of Lemma 1), we have

$$|X(T)| = |X(T')| = 12n + 23m + 3 + 2(3n + 5m) + 1 = 18n + 33m + 4.$$

It now follows, that the size of T and T' as well as the time it takes to construct these two trees are polynomial. This completes the proof of the theorem. □

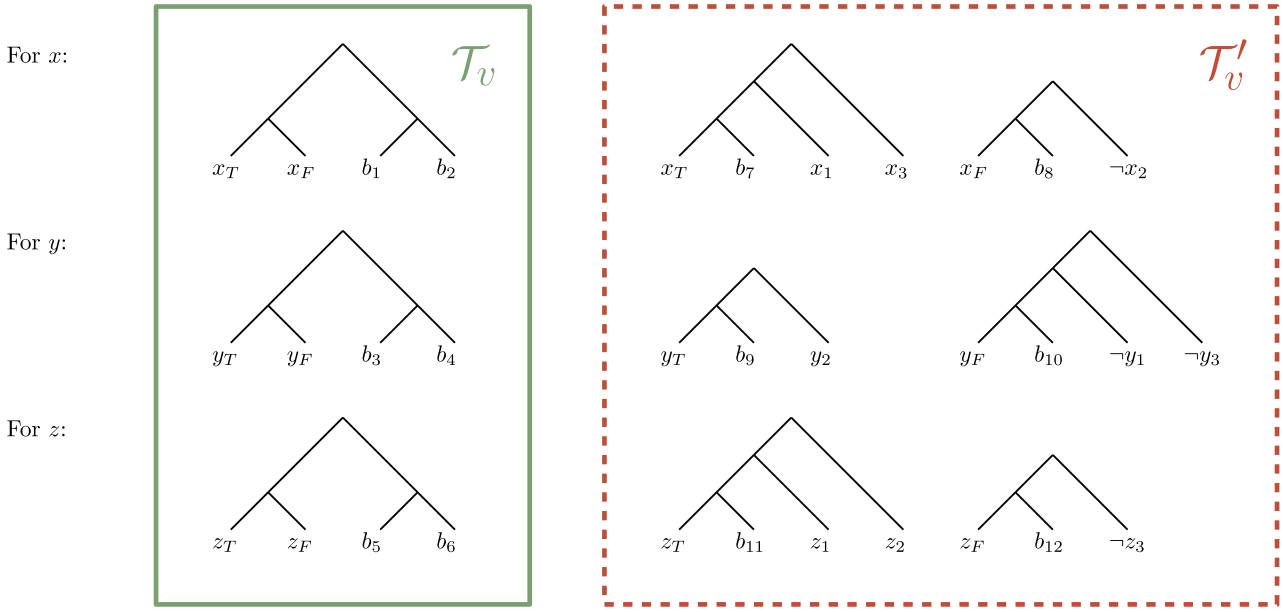


Fig. 7. The variable gadget for $(x \vee \neg y \vee z) \wedge (\neg x \vee y \vee z) \wedge (x \vee \neg y \vee \neg z)$.

To illustrate the proof of Theorem 1, we now give an explicit example of a 3-SAT instance and show how it is reduced to a set of trees by following the construction that is described in the aforementioned proof. Let I be the following instance of 3-SAT

$$\underbrace{(x \vee \neg y \vee z)}_{c_1} \wedge \underbrace{(\neg x \vee y \vee z)}_{c_2} \wedge \underbrace{(x \vee \neg y \vee \neg z)}_{c_3}.$$

For the purpose of ordering the blocking taxa in the same way as described in the proof, we regard variable x as $v^{(1)}$, variable y as $v^{(2)}$, and variable z as $v^{(3)}$. Let $n = 3$ (resp. $m = 3$) be the number of variables (resp. clauses) in I. We construct a set $\mathcal{T}$ of $3n + 5m + 2 = 26$ trees. The 9 trees that represent the variable gadget $\mathcal{T}_v$ and $\mathcal{T}'_v$ are shown in Fig. 7, the 15 trees that represent the clause gadget $\mathcal{T}_c$ and $\mathcal{T}'_c$ are shown in Fig. 8 and the two trees that represent the formula gadget T_f and T'_f are shown in Fig. 9. Note that $|X(\mathcal{T})| = 3n + 13m + 3 = 60$. Clearly, I is satisfied for the truth assignment $\beta : \{x, y, z\} \to \{T, F\}$ with $\beta(x) = \beta(z) = T$, $\beta(y) = F$. To see that $\mathcal{T}$ also has a cherry-picking sequence of length 60, we follow the sequence of pruning operations that is described in Parts 1–3 in the first direction of the proof of Claim 1

$$\begin{aligned}&(x_T, y_F, z_T,\\ &x_1, c_1^1, \cup_a^1, \neg y_1, c_2^1, \cup_b^1, z_1, \cup_c^1, \cup_1, c_1^2, \cup_a^2, c_2^2, \cup_b^2, z_2, \cup_c^2, \cup_2, x_3, \cup_a^3, \neg y_3, c_2^3, \cup_b^3, c_3^3, \cup_c^3, \cup_3,\\ &b_{21}, b_{20}, \ldots, b_{13}, c_3^1, c_3^2, c_1^3, b_{12}, b_{11}, \ldots, b_7, b_X, b_Y, b_6, b_5, \ldots, b_1, x_F, y_T, z_F, \neg x_2, y_2, \neg z_3, b_Z),\end{aligned}$$

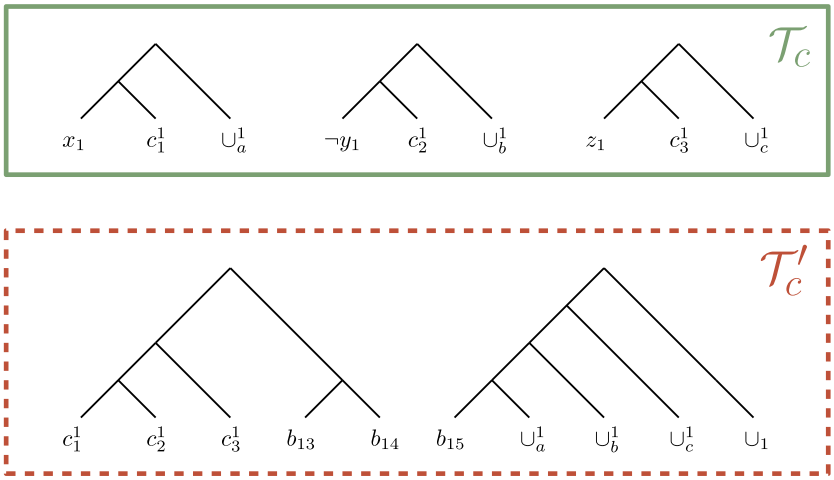
where line 1 corresponds to Part 1, line 2 corresponds to Part 2, and line 3 corresponds to Part 3.

4. Discussion

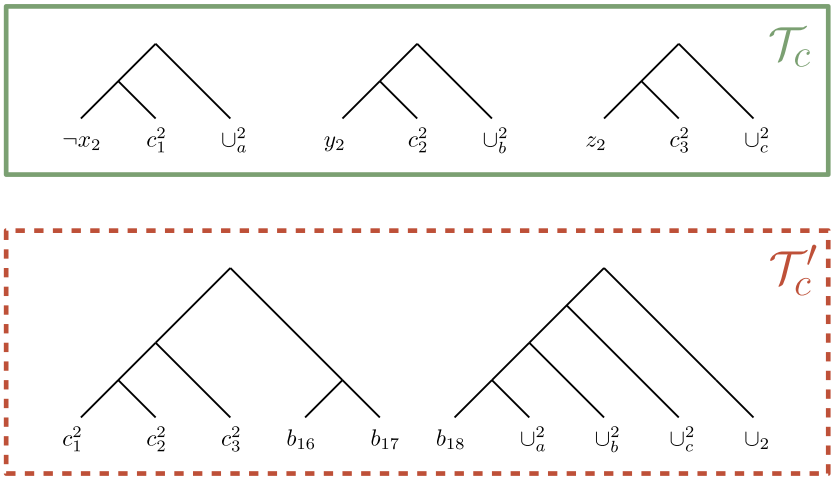
Given any set of input trees, there always exists some phylogenetic network displaying them. Roughly speaking, one can simply merge the input trees at the leaves and at the root. However, what happens when you restrict the network to have some additional, biologically motivated, properties? Then there might not always exist a network displaying the input trees. Moreover, deciding whether or not there exists such a network may be a difficult problem. Indeed, in this paper we have shown that even if the input consists of only two binary trees, it is already NP-complete to decide whether there exists any temporal phylogenetic network displaying them.

One could be tempted to look for approximation algorithms for the associated optimization problem: given a set of phylogenetic trees, find a temporal network that displays them and has smallest possible reticulation number, if such a network exists. Note, however, that an approximation algorithm is required to always output a valid solution, for any valid input. The problem formulation above (based on [13]) does not specify what a valid solution is when there does not exist a temporal network displaying the input trees. Nevertheless, whatever the output in that case is, it can be checked in polynomial time whether the output of the algorithm is a temporal network displaying the input trees. This is because temporal networks are tree-child, and checking whether a tree-child network displays a tree can be achieved in polynomial time [16]. Hence, any approximation algorithm for the problem could be used to decide in polynomial time whether there

For C_1:



For C_2:



For C_3:

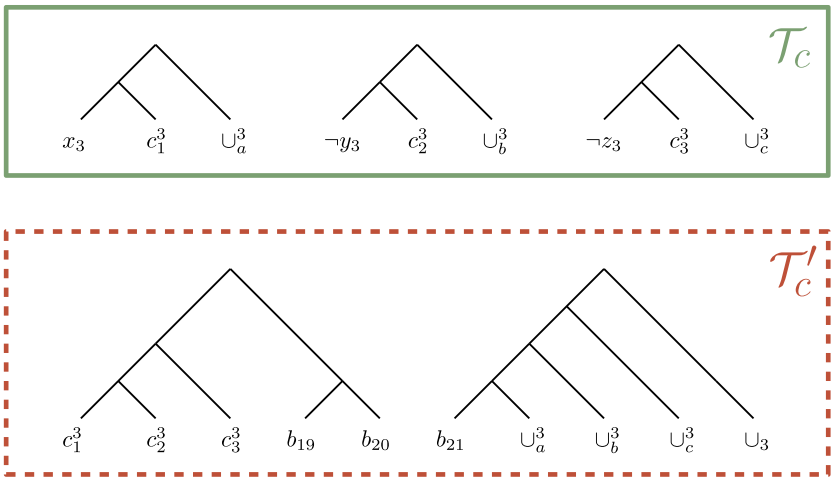


Fig. 8. The clause gadget for $(x \vee \neg y \vee z) \wedge (\neg x \vee y \vee z) \wedge (x \vee \neg y \vee \neg z)$.

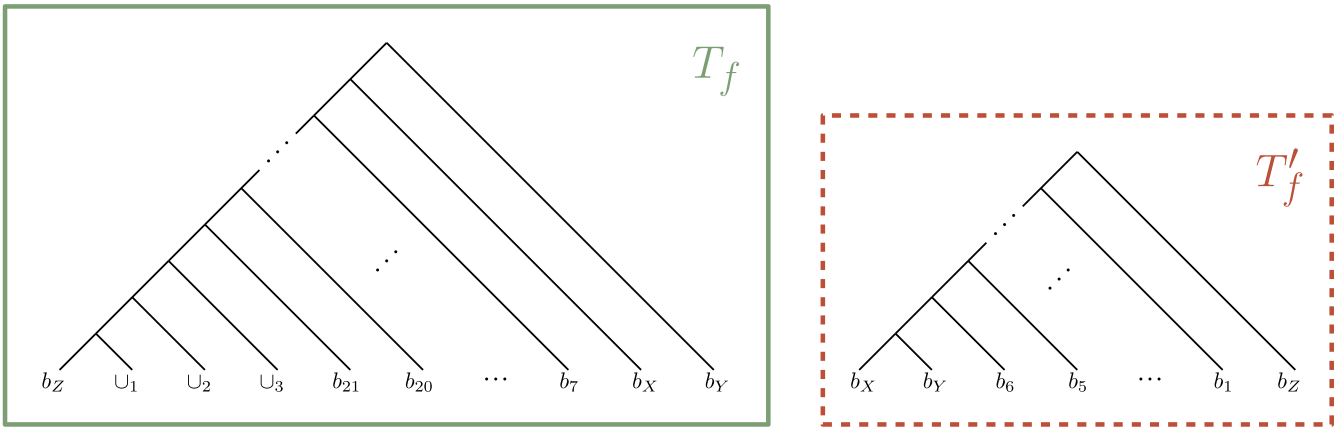


Fig. 9. The formula gadget for $(x \vee \neg y \vee z) \wedge (\neg x \vee y \vee z) \wedge (x \vee \neg y \vee \neg z)$.

exists a temporal network displaying the input trees, which is not possible, unless P = NP, given the NP-completeness shown in this paper.

Therefore, a more promising direction is to consider fixed-parameter algorithms for the associated parameterized version of the problem. Given a set of phylogenetic trees and a parameter k, decide whether there exists a temporal network that displays the input trees and has reticulation number at most k. One then aims at algorithms solving this problem in $O(|X|^{O(1)}f(k))$ time, with f some function of k, preferably of the form c^k with c a small constant. Intuitively, such an FPT algorithm is only exponential in the reticulation number and not in the number of leaves. Indeed, even though it is NP-complete to decide whether there exists a temporal network with unlimited reticulation number, for small reticulation numbers this problem might be much easier. In fact, for instances of two binary trees a fixed-parameter algorithm is already known [12]. Important open problems include the question whether such algorithms exist for instances of more than two trees and whether algorithms can be developed that work well in practice.

It would also be interesting to consider other biologically motivated network classes. For example, binary tree-child (e.g. [5]) or tree-sibling networks (e.g. [3]). Could it be that one of the associated decision problems is nontrivial (for more than two input trees) but polynomial-time solvable? For other network classes, such as tree-based (e.g. [9]) or time-consistent (e.g. [4]) networks, it is known that there always exists a solution [25]. For such classes, it would be interesting to study the optimization version of the problem.

Acknowledgments

We thank Matúš Mihalák for useful discussions. Leo van Iersel was partly supported by the Netherlands Organization for Scientific Research (NWO), including Vidi grant 639.072.602, and partly by the 4TU Applied Mathematics Institute. Simone Linz was supported by the New Zealand Marsden Fund.

References

[1] E. Bapteste, L. van Iersel, A. Janke, S. Kelchner, S. Kelk, J. McInerney, D. Morrison, L. Nakhleh, M. Steel, L. Stougie, J. Whitfield, Networks: expanding evolutionary thinking, Trends Genet. 29 (8) (2013) 439–441.
[2] M. Bordewich, C. Semple, Computing the hybridization number of two phylogenetic trees is fixed-parameter tractable, IEEE/ACM Trans. Comput. Biol. Bioinform. 4 (3) (2007) 458–466.
[3] G. Cardona, M. Llabrés, F. Rosselló, G. Valiente, A distance metric for a class of tree-sibling phylogenetic networks, Bioinformatics 24 (13) (2008) 1481–1488.
[4] G. Cardona, M. Llabrés, F. Rosselló, G. Valiente, Path lengths in tree-child time consistent hybridization networks, Inform. Sci. 180 (3) (2010) 366–383.
[5] G. Cardona, F. Rosselló, G. Valiente, Comparison of tree-child phylogenetic networks, IEEE/ACM Trans. Comput. Biol. Bioinform. 6 (4) (2009) 552–569.
[6] Z.-Z. Chen, L. Wang, Hybridnet: a tool for constructing hybridization networks, Bioinformatics 26 (22) (2010) 2912–2913.
[7] M. Cygan, F. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, S. Saurabh, Parameterized Algorithms, vol. 3., Springer, 2015.
[8] J. Döcker, S. Linz, On the existence of a cherry-picking sequence, Theoret. Comput. Sci. 714 (2018) 36–50.
[9] A. Francis, M. Steel, Which phylogenetic networks are merely trees with additional arcs? Syst. Biol. 64 (5) (2015) 768–777.
[10] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, 1979.
[11] D. Gusfield, ReCombinatorics: The Algorithmics of Ancestral Recombination Graphs and Explicit Phylogenetic Networks, The MIT Press, 2014.
[12] P. Humphries, S. Linz, C. Semple, Cherry picking: a characterization of the temporal hybridization number for a set of phylogenies, Bull. Math. Biol. 75 (10) (2013) 1879–1890.
[13] P. Humphries, S. Linz, C. Semple, On the complexity of computing the temporal hybridization number for two phylogenies, Discrete Appl. Math. 161 (7) (2013) 871–880.
[14] D. Huson, R. Rupp, C. Scornavacca, Phylogenetic Networks: Concepts, Algorithms and Applications, Cambridge University Press, 2011.
[15] L. van Iersel, S. Kelk, C. Scornavacca, Kernelizations for the hybridization number problem on multiple nonbinary trees, J. Comput. System Sci. 82 (6) (2016) 1075–1089.
[16] L. van Iersel, C. Semple, M. Steel, Locating a tree in a phylogenetic network, Inform. Process. Lett. 110 (23) (2010) 1037–1043.

[17] R. Karp, Reducibility among combinatorial problems, in: Complexity of Computer Computations (Proc. Sympos. IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972), Plenum, 1972, pp. 85–103.
[18] S. Kelk, L. van Iersel, S. Linz, N. Lekić, C. Scornavacca, Stougie. Cycle killer. L., Qu'est-ce que c'est? On the comparative approximability of hybridization number and directed feedback vertex set, SIAM J. Discrete Math. 26 (4) (2012) 1635–1656.
[19] B. Moret, L. Nakhleh, T. Warnow, C. Linder, A. Tholse, A. Padolina, J. Sun, R. Timme, Phylogenetic networks: modeling, reconstructibility, and accuracy, IEEE/ACM Trans. Comput. Biol. Bioinform. 1 (1) (2004) 13–23.
[20] C. Papadimitriou, M. Yannakakis, Optimization, approximation, and complexity classes, J. Comput. System Sci. 43 (1991) 425–440.
[21] C. Semple, M. Steel, Phylogenetics, Oxford University Press, 2003.
[22] S. Soucy, J. Huang, J. Gogarten, Horizontal gene transfer: building the web of life, Nature Rev. Genet. 16 (8) (2015) 472–482.
[23] M. Steel, Phylogeny: Discrete and Random Processes in Evolution, SIAM, 2016.
[24] C. Whidden, R. Beiko, N. Zeh, Fixed-parameter algorithms for maximum agreement forests, SIAM J. Comput. 42 (4) (2013) 1431–1466.
[25] L. Zhang, On tree-based phylogenetic networks, J. Comput. Biol. 23 (7) (2016) 553–565.

1.3 Displaying trees across two phylogenetic networks

The following paper [DLS19] is also available online at the following URL: `https://doi.org/10.1016/j.tcs.2019.09.003`.

Theoretical Computer Science 796 (2019) 129–146

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Displaying trees across two phylogenetic networks

Janosch Döcker [a], Simone Linz [b,*], Charles Semple [c]

[a] Department of Computer Science, University of Tübingen, Germany
[b] School of Computer Science, University of Auckland, New Zealand
[c] School of Mathematics and Statistics, University of Canterbury, New Zealand

ARTICLE INFO

Article history:
Received 25 February 2019
Received in revised form 27 July 2019
Accepted 3 September 2019
Available online 6 September 2019
Communicated by T. Calamoneri

Keywords:
Phylogenetic network
Tree-Containment
Polynomial-time hierarchy
Display set
Temporal network
Normal network

ABSTRACT

Phylogenetic networks are a generalization of phylogenetic trees to leaf-labeled directed acyclic graphs that represent ancestral relationships between species whose past includes non-tree-like events such as hybridization and horizontal gene transfer. Indeed, each phylogenetic network embeds a collection of phylogenetic trees. Referring to the collection of trees that a given phylogenetic network $\mathcal{N}$ embeds as the display set of $\mathcal{N}$, several questions in the context of the display set of $\mathcal{N}$ have recently been analyzed. For example, the widely studied Tree-Containment problem asks if a given phylogenetic tree is contained in the display set of a given network. The focus of this paper are two questions that naturally arise in comparing the display sets of two phylogenetic networks. First, we analyze the problem of deciding if the display sets of two phylogenetic networks have a tree in common. Surprisingly, this problem turns out to be NP-complete even for two temporal normal networks. Second, we investigate the question of whether or not the display sets of two phylogenetic networks are equal. While we recently showed that this problem is polynomial-time solvable for a normal and a tree-child network, it is computationally hard in the general case. In establishing hardness, we show that the problem is contained in the second level of the polynomial-time hierarchy. Specifically, it is Π_2^P-complete. Along the way, we show that two other problems are also Π_2^P-complete, one of which being a generalization of Tree-Containment.

1. Introduction

In trying to disentangle the evolutionary history of species, phylogenetic networks, which are leaf-labeled directed acyclic graphs, are becoming increasingly important. From a biological as well as from a mathematical viewpoint, phylogenetic networks are often regarded as a tool to summarize a collection of conflicting phylogenetic trees. Due to processes such as hybridization and lateral gene transfer, the evolution at the species-level is not necessarily tree-like. Nevertheless, individual genes or parts thereof are usually assumed to evolve in a tree-like way. It is consequently of interest to construct phylogenetic networks that embed a collection of phylogenetic trees or, conversely, summarize the phylogenetic trees that are embedded in a given phylogenetic network. These and related types of problems have recently attracted considerable attention from the mathematical community as they lead to a number of challenging questions. One of the most studied questions in this context is called Tree-Containment. Given a phylogenetic network $\mathcal{N}$ and a phylogenetic tree $\mathcal{T}$, this problem asks whether or not $\mathcal{N}$ embeds $\mathcal{T}$. While Tree-Containment is NP-complete in general [7], it has been shown to be polynomial-time solvable for several popular classes of phylogenetic networks, e.g. so-called tree-child and reticulation-

* Corresponding author.
E-mail addresses: janosch.doecker@uni-tuebingen.de (J. Döcker), s.linz@auckland.ac.nz (S. Linz), charles.semple@canterbury.ac.nz (C. Semple).

https://doi.org/10.1016/j.tcs.2019.09.003
0304-3975/

visible networks [1,6,14]. Formal definitions of these classes are given in the next section. Currently, the fastest algorithm that solves Tree-Containment for these latter types of networks has a running time that is linear in the size of $\mathcal{N}$ [15]. Since the number of vertices in a tree-child and a reticulation-visible network is linear in the number of leaves [1,2], it follows that the running time is in fact linear in the number of leaves of $\mathcal{N}$.

Pushing Tree-Containment into a novel direction, Gunawan et al. [6] have recently posed the question of how one can check if two reticulation-visible networks embed the same set of phylogenetic trees. Since the number of trees that a phylogenetic network $\mathcal{N}$ embeds grows exponentially with the number k of vertices in $\mathcal{N}$ whose in-degree is at least two, there is no immediate check that can be performed in polynomial time. In particular, the number of phylogenetic trees that $\mathcal{N}$ embeds is bounded above by 2^k, and it was shown independently in [14, Theorem 1] and [17, Corollary 3.4] that this upper bound is sharp for the class of normal networks.

Referring to the collection of phylogenetic trees that a given phylogenetic network embeds as its *display set* (formally defined in Section 2), we investigate two questions that naturally arise in comparing the display sets of two phylogenetic networks. The first question asks if the display sets of two phylogenetic networks have a common element. We call this problem Common-Tree-Containment and show in Section 3 that it is NP-complete even when the two input networks are both temporal and normal. The class of temporal and normal networks is a strict subclass of the class of tree-child and, hence, reticulation-visible networks for which Tree-Containment is polynomial-time solvable. The second problem, which we refer to as Display-Set-Equivalence, is the problem of Gunawan et al. [6] mentioned above that asks, without restricting to a particular class of phylogenetic networks, if the display sets of two networks are equal. While we recently showed that this problem has a polynomial-time algorithm for when the input consists of a normal and a tree-child network [3], we show in Section 4 that the problem is computationally hard for two arbitrary phylogenetic networks. Specifically, we show that Display-Set-Equivalence is Π_2^P-complete or, in other words, complete for the second level of the polynomial-time hierarchy [13]. In particular, unless there is a collapse in the polynomial hierarchy, such a problem has no polynomial-time reduction from itself to any NP-complete or co-NP-complete problem. From a practical viewpoint, this means that the frequently-taken approach of applying SAT and ILP solvers to find solutions to NP-complete problems is going to be of limited use when applied to a Π_2^P-complete problem. In establishing that Display-Set-Equivalence is Π_2^P-complete, we also show that deciding if the display set of one phylogenetic network is contained in the display set of another network is Π_2^P-complete.

The paper is organized as follows. The next section contains preliminaries that are used throughout the paper, formal statements of the decision problems that are mentioned in the previous paragraph, and some relevant details about the polynomial-time hierarchy. Section 3 establishes NP-completeness of Common-Tree-Containment and Section 4 establishes Π_2^P-completeness of Display-Set-Equivalence. Lastly, Section 5 contains some concluding remarks and highlights two corollaries that follow from the results in Sections 3.

2. Preliminaries

This section provides notation and terminology that is used in the remaining sections. Throughout this paper, X denotes a non-empty finite set. Let G be a directed acyclic graph. For two distinct vertices u and v in G, we say that u is an *ancestor* of v and v is a *descendant* of u, if there is a directed path from u to v in G. If (u, v) is an edge in G, then u is a *parent* of v and v is a *child* of u. Moreover, a vertex of G with in-degree one and out-degree zero is a *leaf* of G.

Phylogenetic networks and trees. A *rooted binary phylogenetic network $\mathcal{N}$ on X* is a (simple) rooted acyclic digraph that satisfies the following properties:

(i) the (unique) root has in-degree zero and out-degree two,
(ii) the set X is the set of vertices of out-degree zero, each of which has in-degree one, and
(iii) all other vertices have either in-degree one and out-degree two, or in-degree two and out-degree one.

The set X is the *leaf set* of $\mathcal{N}$. Furthermore, the vertices of in-degree one and out-degree two are *tree vertices*, while the vertices of in-degree two and out-degree one are *reticulations*. An edge directed into a reticulation is called a *reticulation edge* while each non-reticulation edge is called a *tree edge*.

Let $\mathcal{N}$ be a rooted binary phylogenetic network on X. If $\mathcal{N}$ has no reticulations, then $\mathcal{N}$ is said to be a *rooted binary phylogenetic X-tree*. To ease reading and since all phylogenetic networks considered in this paper are rooted and binary, we refer to a rooted binary phylogenetic network (resp. a rooted binary phylogenetic tree) simply as a *phylogenetic network* (resp. *a phylogenetic tree*).

Now let $\mathcal{T}$ be a phylogenetic X-tree. If $Y = \{y_1, y_2, \ldots, y_m\}$ is a subset of X, then $\mathcal{T}[-y_1, y_2, \ldots, y_m]$ and, equivalently, $\mathcal{T}|(X - Y)$ denote the phylogenetic tree with leaf set $X - Y$ that is obtained from the minimal rooted subtree of $\mathcal{T}$ that connects all leaves in $X - Y$ by suppressing all vertices of in-degree one and out-degree one.

Remark. Throughout the paper, we frequently detail constructions of phylogenetic networks. To this end, we sometimes need labels of internal vertices. Their only purpose is to make references. Indeed, they should not be regarded as genuine labels as those used for the leaves of a phylogenetic network.

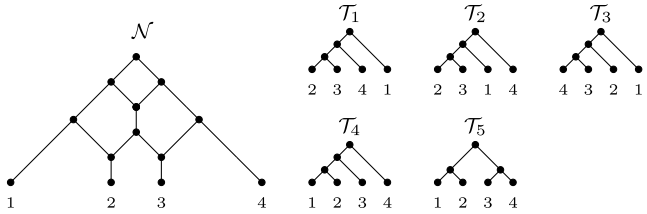


Fig. 1. A phylogenetic network $\mathcal{N}$ and the display set of $\mathcal{N}$ that consists of the five trees shown on the right-hand side.

Classes of phylogenetic networks. Let $\mathcal{N}$ be a phylogenetic network on X with vertex set V. An edge $e = (u, v)$ is a *shortcut* if there is a directed path from u to v whose set of edges does not contain e. A vertex v of $\mathcal{N}$ is called *visible* if there exists a leaf $\ell \in X$ such that each directed path from the root of $\mathcal{N}$ to ℓ passes through v. Now $\mathcal{N}$ is *reticulation-visible* if each reticulation in $\mathcal{N}$ is visible, and $\mathcal{N}$ is *tree-child* if each non-leaf vertex in $\mathcal{N}$ has a child that is a leaf or a tree vertex. Lastly, $\mathcal{N}$ is *normal* if it is tree-child and does not contain any shortcuts. Clearly, by definition, each normal network is also tree-child. Furthermore, it follows from the next well-known equivalence result [2] that each tree-child network is also reticulation-visible.

Lemma 2.1. *Let $\mathcal{N}$ be a phylogenetic network. Then $\mathcal{N}$ is tree-child if and only if each vertex of $\mathcal{N}$ is visible.*

Thus, the class of normal networks is a subclass of tree-child networks. Furthermore, if there exists a map $t : V \to \mathbb{R}^+$ that assigns a time stamp to each vertex of $\mathcal{N}$ and satisfies the following two properties:

(i) $t(u) = t(v)$ whenever (u, v) is a reticulation edge and
(ii) $t(u) < t(v)$ whenever (u, v) is a tree edge,

then we say that $\mathcal{N}$ is *temporal*, in which case we call t a *temporal labeling* of $\mathcal{N}$. Note that, although normal networks have no shortcuts, a normal network need not be temporal. Tree-child, normal, and temporal networks were first introduced by Cardona et al. [2], Willson [16], and Moret et al. [10], respectively.

Caterpillars. Let $\mathcal{C}$ be a phylogenetic tree with leaf set $\{\ell_1, \ell_2, \ldots, \ell_n\}$. Furthermore, for each $i \in \{1, 2, \ldots, n\}$ let p_i denote the parent of ℓ_i. Then $\mathcal{C}$ is called a *caterpillar* if $n \geq 2$ and the elements in the leaf set of $\mathcal{C}$ can be ordered, say $\ell_1, \ell_2, \ldots, \ell_n$, so that $p_1 = p_2$ and, for all $i \in \{3, 4, \ldots, n\}$, we have (p_i, p_{i-1}) as an edge in $\mathcal{C}$. In this case, we denote $\mathcal{C}$ by $(\ell_1, \ell_2, \ldots, \ell_n)$. Additionally, we say that a phylogenetic X-tree $\mathcal{T}$ *contains a caterpillar* $\mathcal{C} = (\ell_1, \ell_2, \ldots, \ell_n)$ if $\mathcal{T}$ has a subtree that is a subdivision of $\mathcal{C}$.

Displaying. Let $\mathcal{N}$ be a phylogenetic network on X and let $\mathcal{T}$ be a phylogenetic Y-tree such that $Y \subseteq X$. Then $\mathcal{N}$ *displays* $\mathcal{T}$ if, up to suppressing vertices of in-degree one and out-degree one, $\mathcal{T}$ can be obtained from $\mathcal{N}$ by deleting edges and vertices, in which case, the edge set, denoted by $E_{\mathcal{T}}$, of the resulting acyclic directed graph is called an *embedding* of $\mathcal{T}$ in $\mathcal{N}$. Note that, if $\mathcal{N}$ displays $\mathcal{T}$, then the root of an embedding of $\mathcal{T}$ in $\mathcal{N}$ need not coincide with the root of $\mathcal{N}$. Moreover, the *display set* of $\mathcal{N}$, denoted by $T(\mathcal{N})$, consists of all phylogenetic X-trees that are displayed by $\mathcal{N}$. As mentioned in the introduction, the size of $T(\mathcal{N})$ is bounded above by 2^k, where k is the number of reticulations in $\mathcal{N}$. To illustrate, Fig. 1 shows a phylogenetic network $\mathcal{N}$ with $T(\mathcal{N}) = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_5\}$, where the five trees in $T(\mathcal{N})$ are shown on the right-hand side of the same figure. In this as well as in all other figures throughout the paper, edges are directed downwards.

Again, let $\mathcal{N}$ be a phylogenetic network on X, and let S be a subset of the edges of $\mathcal{N}$. Then S is a *switching* of $\mathcal{N}$ if, for each reticulation v of $\mathcal{N}$, S contains precisely one of the two reticulation edges that are directed into v. Now, let S be a switching of $\mathcal{N}$. If we delete each reticulation edge in $\mathcal{N}$ that is not in S and, repeatedly, suppress each resulting vertex with in-degree one and out-degree one, delete each vertex with in-degree one and out-degree zero that is not in X, and delete each vertex with in-degree zero and out-degree one, we obtain a phylogenetic X-tree $\mathcal{T}$, in which case, we say that S *yields* $\mathcal{T}$. Note that $\mathcal{T}$ is displayed by $\mathcal{N}$. Conversely, observe that, if $\mathcal{T}$ is a phylogenetic X-tree that is displayed by $\mathcal{N}$, then there exists a switching of $\mathcal{N}$ that yields $\mathcal{T}$. We summarize this in the following observation.

Observation 2.2. *A phylogenetic network $\mathcal{N}$ on X displays a phylogenetic X-tree $\mathcal{T}$ if and only if there exists a switching of $\mathcal{N}$ that yields $\mathcal{T}$.*

Problem statements. Tree-Containment is a well known problem in the study of phylogenetic networks and its computational complexity has extensively been analyzed for various network classes. In the language of this paper, it can be stated as follows.

Tree-Containment
Input. A phylogenetic X-tree $\mathcal{T}$ and phylogenetic network $\mathcal{N}$ on X.
Question. Is $\mathcal{T} \in T(\mathcal{N})$?

While Tree-Containment is concerned with a single display set, it is natural to compare display sets across phylogenetic networks, e.g. in the context of comparing networks. To make a first step in this direction, the focus of this paper are the following three decision problems that compare the display sets of two phylogenetic networks.

Common-Tree-Containment
Input. Two phylogenetic networks $\mathcal{N}$ and $\mathcal{N}'$ on X.
Question. Is $T(\mathcal{N}) \cap T(\mathcal{N}') \neq \emptyset$?

Display-Set-Containment
Input. Two phylogenetic networks $\mathcal{N}$ and $\mathcal{N}'$ on X.
Question. Is $T(\mathcal{N}) \subseteq T(\mathcal{N}')$?

Display-Set-Equivalence
Input. Two phylogenetic networks $\mathcal{N}$ and $\mathcal{N}'$ on X.
Question. Is $T(\mathcal{N}) = T(\mathcal{N}')$?

We note that Tree-Containment is a special case of both Display-Set-Containment and Common-Tree-Containment. Hence, NP-hardness of the two latter problems follows immediately for when $\mathcal{N}$ and $\mathcal{N}'$ are two arbitrary phylogenetic networks. Nevertheless, as we will see in Sections 3 and 4, we pinpoint the complexity of Common-Tree-Containment and Display-Set-Containment exactly. In particular, we will show that (i) Common-Tree-Containment is NP-complete even for when $\mathcal{N}$ and $\mathcal{N}'$ are both temporal and normal and (ii) Display-Set-Containment is complete for the second level of the polynomial-time hierarchy. This last result turns out to be a key ingredient in showing that Display-Set-Equivalence is also complete for the second level of the polynomial-time hierarchy.

The polynomial hierarchy. An *oracle* for a complexity class A is a black box that, in constant time, outputs the answer to any given instance of a decision problem contained in A. The *polynomial-time hierarchy* (or short, *polynomial hierarchy*) [5,13] consists of a system of nested complexity classes that are defined recursively and generalize the classes P, NP, and co-NP. In particular, for any integer $k \geq 0$, we have

$$\Sigma_0^P = \Pi_0^P = \mathrm{P},$$

$$\Sigma_{k+1}^P = \mathrm{NP}^{\Sigma_k^P} \quad \text{and} \quad \Pi_{k+1}^P = \text{co-NP}^{\Sigma_k^P},$$

where a problem is in $\mathrm{NP}^{\Sigma_k^P}$ (resp. $\text{co-NP}^{\Sigma_k^P}$) if we can verify an appropriate certificate of a yes-instance (resp. no-instance) in polynomial-time when given access to an oracle for Σ_k^P. By definition, $\Sigma_1^P = \mathrm{NP}$ and $\Pi_1^P = \text{co-NP}$, and $\Pi_2^P = \text{co-NP}^{\mathrm{NP}}$.

For all $k \geq 0$, we say that the classes Σ_k^P and Π_k^P are on the *k-th level* of the polynomial hierarchy. Although, Σ_{k+1}^P (resp. Π_{k+1}^P) generalizes Σ_k^P (resp. Π_k^P), it is an open problem whether $\Sigma_k^P = \Sigma_{k+1}^P$ or $\Pi_k^P = \Pi_{k+1}^P$ for any $k \geq 0$. Specifically, for $k = 0$, this is the fundamental P versus NP problem. If $\Sigma_k^P = \Sigma_{k+1}^P$ or $\Pi_k^P = \Pi_{k+1}^P$ for some $k \geq 0$, then this would result in a collapse of the polynomial hierarchy to the k-th level.

In Section 4, we show that Display-Set-Containment and Display-Set-Equivalence are both Π_2^P-complete. Intuitively, problems that are complete for the second level of the polynomial hierarchy are more difficult than problems that are complete for the first level. Recall that a decision problem is in co-NP if a no-instance can be verified in polynomial time given an appropriate certificate. Now, similar to showing that a problem is co-NP-complete, a proof that establishes Π_2^P-completeness consists of two steps: (i) show that a problem is in Π_2^P, and (ii) establish a polynomial-time reduction from a problem that is known to be Π_2^P-complete to the problem at hand. With regards to (i), a decision problem is *in* Π_2^P if a no-instance can be verified in polynomial time when one is given an appropriate certificate and has access to an NP-*oracle*, that is, an oracle that can solve NP-complete problems in constant time.

3. Hardness of Common-Tree-Containment

As noted in the introduction, Tree-Containment is NP-complete in general, but polynomial-time solvable for several popular classes of phylogenetic networks such as tree-child and reticulation-visible networks. In this section, we show that no such dichotomy holds for Common-Tree-Containment. In particular, we will show that this problem is NP-complete even if the input consists of two temporal normal networks. To establish the result, we use a reduction from the classical computational problem 3-SAT.

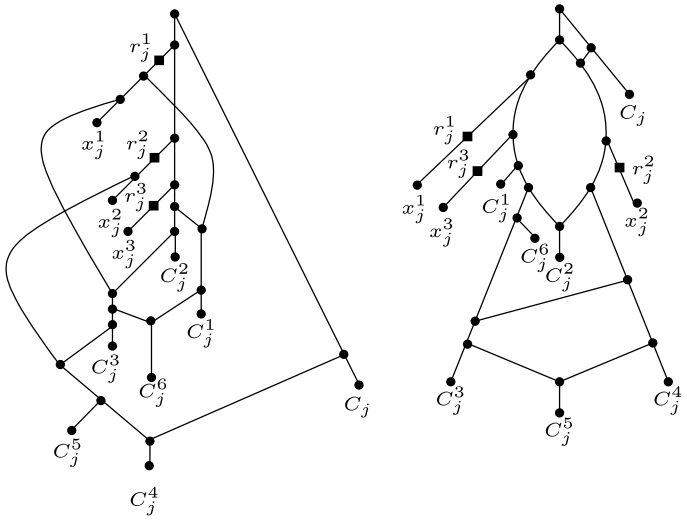


Fig. 2. For a clause $C_j = (x_j^1 \vee x_j^2 \vee x_j^3)$, the clause gadget G_j^A (left) and the clause gadget G_j^B (right). Leaves are bijectively labeled with the elements in $\{C_j, C_j^1, C_j^2, \ldots, C_j^6, x_j^1, x_j^2, x_j^3\}$. Furthermore, each gadget has three vertices of in-degree one and out-degree one indicated by small squares labeled r_j^1, r_j^2, and r_j^3.

3-SAT
Input. A set $V = \{v_1, v_2, \ldots, v_n\}$ of variables, and a set $\{C_1, C_2, \ldots, C_m\}$ of clauses such that each clause is a disjunction of exactly three literals and each literal is an element in $\{v_i, \bar{v}_i : i \in \{1, 2, \ldots, n\}\}$.
Question. Does there exist a truth assignment for V that satisfies each clause C_j with $j \in \{1, 2, \ldots, m\}$?

Let I be an instance of 3-SAT, and let $C_j = (x_j^1 \vee x_j^2 \vee x_j^3)$ be a clause of I for $j \in \{1, 2, \ldots, m\}$. Then, for some indices k, k', and k'' in $\{1, 2, \ldots, n\}$, we have $x_j^1 \in \{v_k, \bar{v}_k\}$, $x_j^2 \in \{v_{k'}, \bar{v}_{k'}\}$, and $x_j^3 \in \{v_{k''}, \bar{v}_{k''}\}$. Without loss of generality, we impose the following two restrictions on I:

(R1) for each $v_i \in V$ with $i \in \{1, 2, \ldots, n\}$, at most one element in $\{v_i, \bar{v}_i\}$ is a literal of C_j and
(R2) $k < k' < k''$.

Now, for each clause C_j, we construct the two clause gadgets G_j^A and G_j^B that are shown in Fig. 2. We next establish a simple lemma.

Lemma 3.1. *Let G_j^A and G_j^B be the two clause gadgets that are shown in Fig. 2. Obtain two phylogenetic networks $\mathcal{G}_j^A$ and $\mathcal{G}_j^B$ from G_j^A and G_j^B, respectively, by suppressing the three vertices r_j^1, r_j^2, and r_j^3 of in-degree one and out-degree one. Then $T(\mathcal{G}_j^A) \cap T(\mathcal{G}_j^B) = \emptyset$.*

Proof. To see that $T(\mathcal{G}_j^A) \cap T(\mathcal{G}_j^B) = \emptyset$, observe that each tree in $T(\mathcal{G}_j^A)$ contains the caterpillar (x_j^2, x_j^3, x_j^1), whereas each tree in $T(\mathcal{G}_j^B)$ contains the caterpillar (x_j^1, x_j^3, x_j^2). □

Following on from Lemma 3.1, let $L = \{x_j^1, x_j^2, x_j^3\}$. Although $\mathcal{G}_j^A$ and $\mathcal{G}_j^B$ display no common phylogenetic tree with leaf set $\{C_j, C_j^1, C_j^2, \ldots, C_j^6\} \cup L$, they do display a common phylogenetic with leaf set $\{C_j, C_j^1, C_j^2, \ldots, C_j^6\} \cup L'$ for each proper subset L' of L. In the proof of Theorem 3.2, for some truth assignment β, the latter corresponds to at least one literal in C_j being satisfied by β, while the former corresponds to no literal in C_j being satisfied by β.

Let $S = (s_1, s_2, \ldots, s_n)$ be an arbitrary tuple, and let r be an element that is not contained in S. We write $(r)||S$ to denote the tuple $(r, s_1, s_2, \ldots, s_n)$ obtained by concatenating r and S. With this definition in hand, we are now in a position to establish the main result of this section.

Theorem 3.2. Common-Tree-Containment *is* NP*-complete when the input consists of two temporal normal networks.*

Proof. For two normal networks, van Iersel et al. [14] showed that the running time of Tree-Containment is polynomial in the size of this leaf set. Hence, it follows that Common-Tree-Containment is in NP for two normal networks.

Let I be an instance of 3-SAT with n variables and m clauses. Using the same notation as in the formal statement of 3-SAT, we construct two phylogenetic networks $\mathcal{N}$ and $\mathcal{N}'$ on

$$X = \left\{C_j, C_j^1, C_j^2, \ldots, C_j^6, x_j^1, x_j^2, x_j^3 : j \in \{1, 2, \ldots, m\}\right\} \cup \left\{v_i : i \in \{1, 2, \ldots, n\}\right\}$$

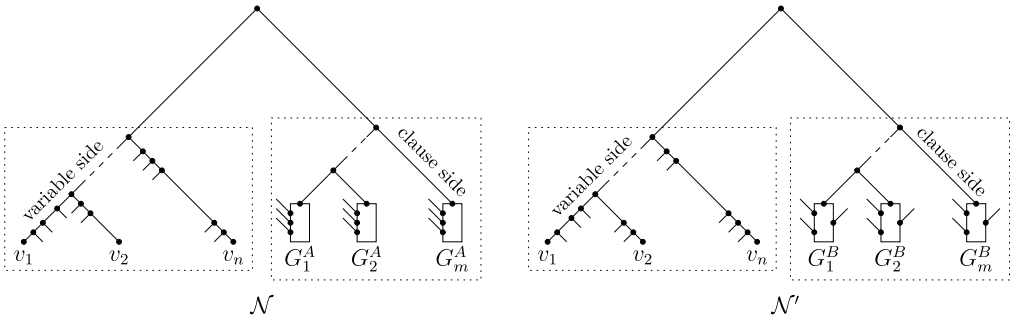


Fig. 3. Overview of the construction of the two temporal normal networks $\mathcal{N}$ and $\mathcal{N}'$ in the proof of Theorem 3.2. Dangling edges on the clause and variable side of $\mathcal{N}$ and $\mathcal{N}'$, respectively, are paired up depending on I. For details, see Step 3 of the construction.

as follows. Let $\mathcal{T}$ be the phylogenetic tree obtained by creating a vertex ρ, adding an edge that joins ρ with the root of the caterpillar $(v_1, v_2, \ldots, v_n)$, and adding an edge that joins ρ with the root of the caterpillar $(c_1, c_2, \ldots, c_m)$. Now, setting $\mathcal{M} = \mathcal{M}' = \mathcal{T}$, let $\mathcal{N}$ and $\mathcal{N}'$ be the two phylogenetic networks obtained from $\mathcal{M}$ and $\mathcal{M}'$, respectively, by applying the following four-step process.

1. For all $j \in \{1, 2, \ldots, m\}$, replace c_j with G_j^A in $\mathcal{M}$ and replace c_j with G_j^B in $\mathcal{M}'$.
2. For all $i \in \{1, 2, \ldots, n\}$, subdivide the edge directed into v_i with a new vertex d_i in $\mathcal{M}$ and $\mathcal{M}'$.
3. For each $j \in \{1, 2, \ldots, m\}$ in increasing order, consider $C_j = (x_j^1 \vee x_j^2 \vee x_j^3)$. Let v_{k_ℓ} be the unique element in V such that $x_j^\ell \in \{v_{k_\ell}, \bar{v}_{k_\ell}\}$ for each $\ell \in \{1, 2, 3\}$. If $x_j^\ell = v_{k_\ell}$, subdivide the edge directed into v_{k_ℓ} with a new vertex u_j^ℓ in $\mathcal{M}$ and subdivide the edge directed into d_{k_ℓ} with a new vertex u_j^ℓ in $\mathcal{M}'$. Otherwise, subdivide the edge directed into d_{k_ℓ} with a new vertex u_j^ℓ in $\mathcal{M}$ and subdivide the edge directed into v_{k_ℓ} with a new vertex u_j^ℓ in $\mathcal{M}'$. Add a new edge (u_j^ℓ, r_j^ℓ) in $\mathcal{M}$ and $\mathcal{M}'$.
4. For each $i \in \{1, 2, \ldots, n\}$, suppress the vertex d_i of in-degree one and out-degree one in $\mathcal{M}$ and $\mathcal{M}'$.

To illustrate, Fig. 3 gives a high-level overview of the construction of $\mathcal{N}$ and $\mathcal{N}'$. Observe that, for each $j \in \{1, 2, \ldots, m\}$, the three vertices r_j^1, r_j^2, and r_j^3 in $\mathcal{N}$ and $\mathcal{N}'$ are reticulations.

We next show that $\mathcal{N}$ and $\mathcal{N}'$ are both temporal and normal.

3.2.1. *Both $\mathcal{N}$ and $\mathcal{N}'$ are temporal and normal.*

Proof. We first show that $\mathcal{N}$ is temporal and normal. Let

$$V_r = \{r_j^\ell : j \in \{1, 2, \ldots, m\} \text{ and } \ell \in \{1, 2, 3\}\}.$$

Furthermore, for each $i \in \{1, 2, \ldots, n\}$, let V_i consist of all vertices that lie on the unique directed path from the root of $\mathcal{N}$ to v_i, and let

$$V_v = \bigcup_{i=1}^{n} V_i.$$

We begin by assigning a positive real-valued labeling t to each vertex in $V_v \cup V_r$ as follows. First, under t, each vertex in V_v is assigned a labeling such that the following two properties are satisfied.

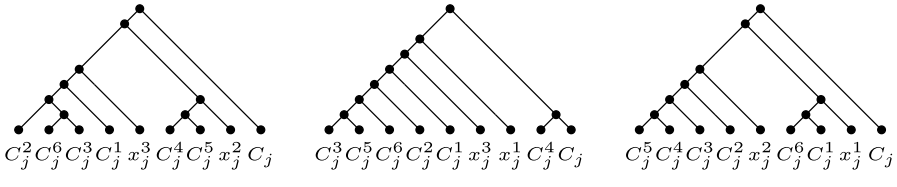
(i) If $u, v \in V_v$ and u is an ancestor of v, then $t(u) < t(v)$.
(ii) For all $i \in \{1, 2, \ldots, n-1\}$, the temporal labeling of each vertex in V_i that is not contained in V_{i+1} is smaller than the minimum temporal labeling over all vertices that are contained in V_{i+1} and not in V_i.

By construction of $\mathcal{N}$, note that such a labeling always exists. Second, under t, each vertex in V_r is assigned the same labeling as its unique parent that is contained in V_v. Because of restrictions (R1) and (R2) that we have imposed on I and the way we have assigned temporal labelings to the vertices in V_v, we have

$$t(r_j^1) < t(r_j^2) < t(r_j^3)$$

for each $j \in \{1, 2, \ldots, m\}$. A routine check now shows that t can be extended to a temporal labeling of $\mathcal{N}$ and, thus, $\mathcal{N}$ is temporal.

Now, since $\mathcal{N}$ is temporal, it follows that $\mathcal{N}$ has no shortcuts. Hence, to show that $\mathcal{N}$ is normal, it suffices to show that $\mathcal{N}$ is tree-child. It is straightforward to check that $\mathcal{N}$ has no edge (u, v) such that u and v are both reticulations.



(1) $\mathcal{T}_j^{TFF}$ (2) $\mathcal{T}_j^{FTF}$ (3) $\mathcal{T}_j^{FFT}$

(4) $\mathcal{T}_j^{FTT} = \mathcal{T}_j^{FTF}[-x_j^3]$ (5) $\mathcal{T}_j^{TFT} = \mathcal{T}_j^{FFT}[-x_j^1]$

(6) $\mathcal{T}_j^{TTF} = \mathcal{T}_j^{FTF}[-x_j^1]$ (7) $\mathcal{T}_j^{TTT} = \mathcal{T}_j^{FTF}[-x_j^1, x_j^3]$

Fig. 4. The seven trees that are used in the proof of Theorem 3.2.

Hence, each reticulation in $\mathcal{N}$ has a child that is a tree vertex or a leaf. Furthermore, by construction, each tree vertex of $\mathcal{N}$ that is a vertex of some G_j^A with $j \in \{1, 2, \ldots, m\}$ has a child that is a tree vertex or a leaf. Lastly, for each non-leaf vertex v of $\mathcal{N}$ that is neither a reticulation nor a vertex of some G_j^A, consider a directed path P from v to an element in $\{v_1, v_2, \ldots, v_n, C_1, C_2, \ldots, C_m\}$. By construction, P exists. It is now easily seen that the second vertex of P is a child of v that is either a tree vertex or a leaf. This establishes that $\mathcal{N}$ is normal. An analogous argument that uses G_j^B instead of G_j^A can be used to show that $\mathcal{N}'$ is temporal and normal, thereby completing the proof of (3.2.1). □

Since the number of vertices of a normal network is polynomial in the size of X [9] and $|X| = 10m + n$, it follows that $\mathcal{N}$ and $\mathcal{N}'$ can be constructed in time polynomial in the size of X.

3.2.2. *The instance I is a yes-instance if and only if $T(\mathcal{N}) \cap T(\mathcal{N}') \neq \emptyset$.*

Proof. First, suppose that I is a yes-instance. We construct a *variable tree* $\mathcal{T}_v$ and a *clause tree* $\mathcal{T}_c$ that, joined together, result in a phylogenetic X-tree that is displayed by $\mathcal{N}$ and $\mathcal{N}'$. Let $\beta : V \to \{F, T\}$ be a truth assignment that satisfies each clause, and let

$$Y = \{x_j^\ell : j \in \{1, 2, \ldots, m\} \text{ and } \ell \in \{1, 2, 3\}\}.$$

Furthermore, for each $i \in \{1, 2, \ldots, n\}$, let Y_i (resp. $\bar{Y}_i$) be the tuple consisting of the elements in Y that equal v_i (resp. $\bar{v}_i$) such that, for any two elements x_j^ℓ and $x_{j'}^{\ell'}$ in Y_i (resp. $\bar{Y}_i$), x_j^ℓ precedes $x_{j'}^{\ell'}$ precisely if $j > j'$. By construction, note that the two caterpillars $(v_i)||Y_i$ and $(v_i)||\bar{Y}_i$ are displayed by $\mathcal{N}$ and $\mathcal{N}'$. Now, obtain $\mathcal{T}_v$ from the caterpillar $(v_1, v_2, \ldots, v_n)$ by doing the following for each $i \in \{1, 2, \ldots, n\}$. If $\beta(v_i) = T$, replace v_i with the caterpillar $(v_i)||Y_i$; otherwise, replace v_i with the caterpillar $(v_i)||\bar{Y}_i$. Again, by construction, it is easily checked that $\mathcal{T}_v$ is displayed by $\mathcal{N}$ and $\mathcal{N}'$. We next construct $\mathcal{T}_c$. Consider a clause $C_j = (x_j^1 \vee x_j^2 \vee x_j^3)$. For each $\ell \in \{1, 2, 3\}$, set $z_\ell = T$ if x_j^ℓ is satisfied by β and, otherwise, set $z_\ell = F$. Depending on which elements in $\{z_1, z_2, z_3\}$ equal F and T, respectively, and noting that there exists some ℓ for which $z_\ell = T$, we define the *clause tree* $\mathcal{T}_j^{z_1 z_2 z_3}$ relative to C_j to be one of the seven trees that are listed in Fig. 4. Intuitively, x_j^ℓ is a leaf in $\mathcal{T}_j^{z_1 z_2 z_3}$ precisely if $z_\ell = F$. Now, obtain $\mathcal{T}_c$ from the caterpillar $(c_1, c_2, \ldots, c_m)$ by replacing, for each $j \in \{1, 2, \ldots, m\}$, the leaf c_j with the clause tree relative to C_j. As $\mathcal{T}_j^{z_1 z_2 z_3}$ is displayed by the two phylogenetic networks obtained from G_j^A and G_j^B by suppressing the three vertices r_j^1, r_j^2, and r_j^3 of in-degree one and out-degree one, it follows that $\mathcal{T}_j^{z_1 z_2 z_3}$ is also displayed by $\mathcal{N}$ and $\mathcal{N}'$. In turn, this implies that, by construction, $\mathcal{T}_c$ is displayed by $\mathcal{N}$ and $\mathcal{N}'$. Lastly, we construct a phylogenetic tree $\mathcal{T}$ on X by creating a vertex ρ, adding a new edge that joins ρ with the root of $\mathcal{T}_v$, and a new edge that joins ρ with the root of $\mathcal{T}_c$. As $\mathcal{T}_v$ and $\mathcal{T}_c$ are displayed by $\mathcal{N}$ and $\mathcal{N}'$, it is easily checked that $\mathcal{T}$ is displayed by $\mathcal{N}$ and $\mathcal{N}'$, and so $T(\mathcal{N}) \cap T(\mathcal{N}') \neq \emptyset$.

Second, suppose that $T(\mathcal{N}) \cap T(\mathcal{N}') \neq \emptyset$. Let $\mathcal{T}$ be a phylogenetic X-tree that is displayed by $\mathcal{N}$ and $\mathcal{N}'$. Furthermore, let $j, j' \in \{1, 2, \ldots, m\}$, and let $\ell, \ell' \in \{1, 2, 3\}$. For each reticulation r_j^ℓ in $\mathcal{N}$ (resp. $\mathcal{N}'$), we say that $\mathcal{T}$ *picks* x_j^ℓ *from the clause side* of $\mathcal{N}$ (resp. $\mathcal{N}'$) if $\mathcal{T}$ has a vertex whose set of descendants contains x_j^ℓ and C_j but does not contain any element in V; otherwise, we say that $\mathcal{T}$ *picks* x_j^ℓ *from the variable side* of $\mathcal{N}$ (resp. $\mathcal{N}'$). Intuitively, x_j^ℓ is picked from the clause side of $\mathcal{N}$ (resp. $\mathcal{N}'$) precisely if the embedding of $\mathcal{T}$ in $\mathcal{N}$ (resp. $\mathcal{N}'$) contains the reticulation edge directed into r_j^ℓ whose two end vertices are vertices of G_j^A (resp. G_j^B). Note that, as $\mathcal{T}$ is displayed by $\mathcal{N}$ and $\mathcal{N}'$, we have that $\mathcal{T}$ picks x_j^ℓ from the variable side of $\mathcal{N}$ if and only if $\mathcal{T}$ picks x_j^ℓ from the variable side of $\mathcal{N}'$. We next make two observations:

(O1) For each clause $C_j = (x_j^1 \vee x_j^2 \vee x_j^3)$, it follows from Lemma 3.1 that $\mathcal{T}$ picks at most two of x_j^1, x_j^2, and x_j^3 from the clause side of $\mathcal{N}$ and $\mathcal{N}'$.

(O2) It follows from Step 3 in the construction of $\mathcal{N}$ and $\mathcal{N}'$, and the fact that $\mathcal{T}$ is displayed by $\mathcal{N}$ and $\mathcal{N}'$ that, if $\mathcal{T}$ picks x_j^ℓ from the variable side of $\mathcal{N}$ and $\mathcal{N}'$, and $x_j^\ell = v_i$ for some $i \in \{1, 2, \ldots, n\}$, then each $x_{j'}^{\ell'}$ with $x_{j'}^{\ell'} = \bar{v}_i$ is picked from the clause side of $\mathcal{N}$ and $\mathcal{N}'$. Similarly, if $\mathcal{T}$ picks x_j^ℓ from the variable side of $\mathcal{N}$ and $\mathcal{N}'$, and $x_j^\ell = \bar{v}_i$ for some $i \in \{1, 2, \ldots, n\}$, then each $x_{j'}^{\ell'}$ with $x_{j'}^{\ell'} = v_i$ is picked from the clause side of $\mathcal{N}$ and $\mathcal{N}'$.

Now, let β be the truth assignment that is defined as follows. For each $i \in \{1, 2, \ldots, n\}$, we set $v_i = T$ if there exists an element x_j^ℓ with $x_j^\ell = v_i$ that is picked from the variable side of $\mathcal{N}$ and $\mathcal{N}'$. On the other hand, we set $v_i = F$ if either there exists an element x_j^ℓ with $x_j^\ell = \bar{v}_i$ that is picked from the variable side of $\mathcal{N}$ and $\mathcal{N}'$ or there is no x_j^ℓ with $x_j^\ell \in \{v_i, \bar{v}_i\}$ that is picked from the variable side of $\mathcal{N}$ and $\mathcal{N}'$. Because of (O2), β is well defined. Moreover, by (O1) it follows that β satisfies at least one literal of each clause and, hence, I is a yes-instance. □

This completes the proof of Theorem 3.2. □

The next corollary is an immediate consequence of Theorem 3.2.

Corollary 3.3. *Let $\mathcal{N}$ and $\mathcal{N}'$ be two temporal normal networks on X. It is* co-NP*-complete to decide if $T(\mathcal{N}) \cap T(\mathcal{N}') = \emptyset$.*

4. Hardness of Display-Set-Equivalence

In this section, we show that Display-Set-Equivalence is Π_2^P-complete, that is, the problem is complete for the second level of the polynomial hierarchy. To establish this result, we use a chain of three polynomial-time reductions that are described in Subsections 4.1, 4.2, and 4.3. Before detailing the reductions, we introduce two more decision problems that play an important role in this section.

Recall the (ordinary) 3-SAT problem as introduced in Section 3. The input to an instance of 3-SAT consists of a boolean formula over a set of variables. Importantly, each variable is existentially quantified since we are asking whether or not there *exists* a truth assignment to each variable that satisfies each clause of the formula. In contrast, the following quantified version of 3-SAT has two different types of variables, i.e. each variable is either existentially or universally quantified.

∀∃ 3-SAT
Input. A quantified boolean formula

$$\Psi = \forall v_1 \forall v_2 \cdots \forall v_p \exists v_{p+1} \exists v_{p+2} \cdots \exists v_n \bigwedge_{j=1}^{m} C_j$$

over a set of variables $V = \{v_1, v_2, \ldots, v_n\}$ such that each clause C_j is a disjunction of exactly three literals and each literal is an element in $\{v_i, \bar{v}_i : i \in \{1, 2, \ldots, n\}\}$.
Question. For each truth assignment $\beta^\forall : \{v_1, v_2, \ldots, v_p\} \to \{F, T\}$, does there exist a truth assignment $\beta^\exists : \{v_{p+1}, v_{p+2}, \ldots, v_p\} \to \{F, T\}$ such that, collectively, $\beta^\forall$ and $\beta^\exists$ satisfy each clause in Ψ?

It was shown in [13] that ∀∃ 3-SAT is Π_2^P-complete. Let I be an instance of ∀∃ 3-SAT. Note that each clause of I has at least one literal that is an element in $\{x_i, \bar{x}_i : i \in \{p+1, p+2, \ldots, n\}\}$ since, otherwise, I is a no-instance. Furthermore, if all variables are existentially quantified, then I is an instance of the (ordinary) 3-SAT problem. Hence, we may assume throughout this section that $1 \leq p < n$.

We next formally state a quantified version of the well-known NP-complete decision problem Directed-Disjoint-Connecting-Paths [5,11]. Let G be a directed graph with vertex set V, and let $\{(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)\}$ be a collection of disjoint pairs of vertices in V. In what follows, we write π_i to denote a directed path in G from s_i to t_i with $i \in \{1, 2, \ldots, k\}$.

∀∃ Directed-Disjoint-Connecting-Paths
Input. A directed graph G and two collections

$$P^\forall = \{(s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p)\},$$
$$P^\exists = \{(s_{p+1}, t_{p+1}), (s_{p+2}, t_{p+2}), \ldots, (s_k, t_k)\}$$

of disjoint pairs of vertices in G such that $1 \leq p < k$ and, for each $(s_i, t_i) \in P^\forall$, there exists a directed path from s_i to t_i in G.
Question. For each set $\Pi^\forall = \{\pi_1, \pi_2, \ldots, \pi_p\}$ of directed paths, does there exist a set $\Pi^\forall \cup \{\pi_{p+1}, \pi_{p+2}, \ldots, \pi_k\}$ of mutually vertex-disjoint directed paths in G?

4.1. ∀∃ Directed-Disjoint-Connecting-Paths is Π_2^P-complete

To show that ∀∃ Directed-Disjoint-Connecting-Paths is complete for the second level of the polynomial hierarchy, we use a polynomial-time reduction from ∀∃ 3-SAT. This reduction constructs a special instance of ∀∃ Directed-Disjoint-Connecting-Paths for which the input graph is a particular type of phylogenetic network.

Let $\mathcal{N}$ be a phylogenetic network on X, let $S = \{s_1, s_2, \ldots, s_k\}$ and $T = \{t_1, t_2, \ldots, t_k\}$ be two disjoint subsets of the vertices of $\mathcal{N}$ such that $T = X$, and let $p \in \{1, 2, \ldots, k\}$. We call $\mathcal{N}$ a *caterpillar-inducing* network with respect to S if the network obtained from $\mathcal{N}$ by deleting each vertex that lies on a directed path from a child of a vertex in S to a leaf of $\mathcal{N}$ is a caterpillar up to deleting all leaf labels. Moreover, we say that $\mathcal{N}$ has the *two-path property relative to* p if, for each $i \in \{1, 2, \ldots, p\}$, there are two directed paths, say π_i and π_i', from s_i to t_i such that the following three properties are satisfied:

(i) π_i and π_i' are the only directed paths from s_i to t_i in $\mathcal{N}$,
(ii) π_i and π_i' only have the three vertices s_i, t_i, and the (unique) parent of t_i as well as the edge directed into t_i in common, and
(iii) no path in $\{\pi_i, \pi_i' : i \in \{1, 2, \ldots, p\}\}$ intersects with any path in $\{\pi_j, \pi_j' : j \in \{1, 2, \ldots, p\} - \{i\}\}$.

Using the same notation as in the statement of ∀∃ Directed-Disjoint-Connecting-Paths, we now introduce a similar problem whose input graph is a phylogenetic network.

∀∃ Phylo-Directed-Disjoint-Connecting-Paths
Input. A phylogenetic network $\mathcal{N}$ on X, two disjoint sets $S = \{s_1, s_2, \ldots, s_k\}$ and $T = X = \{t_1, t_2, \ldots, t_k\}$ of vertices of $\mathcal{N}$, and an integer p with $1 \leq p < k$ such that $\mathcal{N}$ is caterpillar-inducing with respect to S and has the two-path property relative to p. Furthermore, the two collections

$$P^{\forall} = \{(s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p)\},$$
$$P^{\exists} = \{(s_{p+1}, t_{p+1}), (s_{p+2}, t_{p+2}), \ldots, (s_k, t_k)\}$$

of pairs of elements in S and T.
Question. For each set $\Pi^{\forall} = \{\pi_1, \pi_2, \ldots, \pi_p\}$ of directed paths, does there exist a set $\Pi^{\forall} \cup \{\pi_{p+1}, \pi_{p+2}, \ldots, \pi_k\}$ of mutually vertex-disjoint directed paths in $\mathcal{N}$?

The next theorem establishes the Π_2^P-completeness of ∀∃ Phylo-Directed-Disjoint-Connecting-Paths. The reduction that we use for the proof has a flavor that is similar to that in [8, page 86].

Theorem 4.1. *The decision problem* ∀∃ Phylo-Directed-Disjoint-Connecting-Paths *is* Π_2^P*-complete.*

Proof. We first show that ∀∃ Phylo-Directed-Disjoint-Connecting-Paths is in Π_2^P. Using the same notation as in the formal statement of this problem, let $\Pi^{\forall} = \{\pi_1, \pi_2, \ldots, \pi_p\}$ be a set of directed paths in $\mathcal{N}$. Since $\mathcal{N}$ has the two-path property relative to p, the paths in $\Pi^{\forall}$ are mutually vertex disjoint. Next obtain the directed graph G from $\mathcal{N}$ by deleting all vertices that lie on a path in $\Pi^{\forall}$. Lastly, use an NP-oracle for the unquantified version of Directed-Disjoint-Connecting-Paths to decide if there exists a set $\Pi^{\exists} = \{\pi_{p+1}, \pi_{p+2}, \ldots, \pi_k\}$ of mutually vertex-disjoint directed paths in G. Since a given instance of ∀∃ Phylo-Directed-Disjoint-Connecting-Paths is a no-instance precisely if there exists some set $\Pi^{\forall}$ for which no choice of $\Pi^{\exists}$ results in a set $\Pi^{\forall} \cup \Pi^{\exists}$ of mutually vertex-disjoint directed paths in $\mathcal{N}$, it follows that this problem is in co-$\mathrm{NP}^{\mathrm{NP}} = \Pi_2^P$.

We now establish a polynomial-time reduction from the quantified 3-SAT problem. Let I be an instance of ∀∃ 3-SAT with boolean formula

$$\Psi = \forall v_1 \forall v_2 \cdots \forall v_p \exists v_{p+1} \exists v_{p+2} \cdots \exists v_n \bigwedge_{j=1}^{m} C_j$$

over a set $V = \{v_1, v_2, \ldots, v_n\}$ of variables. Throughout the proof, we use $C_j = (x_{3j-2} \vee x_{3j-1} \vee x_{3j})$ to refer to the three literals in C_j for each $j \in \{1, 2, \ldots, m\}$. Now, for each $i \in \{1, 2, \ldots, n\}$, let $\mathcal{J}_i^+$ be the set that consists of the indices of the literals that are equal to v_i and, similarly, let $\mathcal{J}_i^-$ be the set that consists of the indices of the literals that are equal to $\bar{v}_i$. Without loss of generality, we may assume that $\mathcal{J}_i^+ \neq \emptyset$ or $\mathcal{J}_i^- \neq \emptyset$ since, otherwise, v_i can be deleted from V.

For each variable v_i, we construct a variable gadget G_i^v as follows:

1. Create three vertices s_i^v, t_i^v, and y_i.
2. Create the (possibly empty) set of vertices $\bigcup_{l \in \mathcal{J}_i^+} \{p_l^{\mathrm{in}}, p_l^{\mathrm{out}}\}$ and construct the directed path

$$\pi_i^+ = (s_i^v,\ p_{l_1}^{\mathrm{in}},\ p_{l_1}^{\mathrm{out}},\ p_{l_2}^{\mathrm{in}},\ p_{l_2}^{\mathrm{out}},\ \ldots,\ p_{l_q}^{\mathrm{in}},\ p_{l_q}^{\mathrm{out}},\ y_i, t_i^v)$$

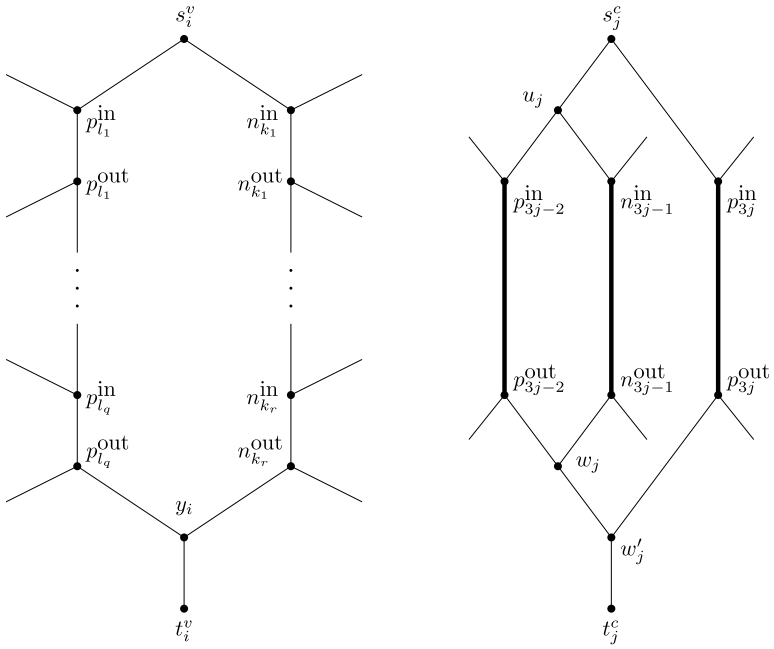


Fig. 5. Left: Variable gadget for a variable v_i. Right: Clause gadget for a clause $C_j = (x_{3j-2} \vee x_{3j-1} \vee x_{3j})$, where the second literal equals a negated variable and each of the other two literals equals an unnegated variable. The three thick edges are edges of a variable gadget. The complete construction is detailed in the proof of Theorem 4.1.

with $\{l_1, l_2, \ldots, l_q\} = \mathcal{J}_i^+$.

3. Create the (possibly empty) set of vertices $\bigcup_{k \in \mathcal{J}_i^-} \{n_k^{\text{in}}, n_k^{\text{out}}\}$ and construct the directed path

$$\pi_i^- = (s_i^v, n_{k_1}^{\text{in}}, n_{k_1}^{\text{out}}, n_{k_2}^{\text{in}}, n_{k_2}^{\text{out}}, \ldots, n_{k_r}^{\text{in}}, n_{k_r}^{\text{out}}, y_i, t_i^v)$$

with $\{k_1, k_2, \ldots, k_r\} = \mathcal{J}_i^-$.

Note that, since we do not allow for parallel edges, the last edge (y_i, t_i^v) of π_i^+ and π_i^- only appears once in G_i^v. Intuitively, the two paths π_i^+ and π_i^- correspond to the two possible truth assignments for the variable v_i. To illustrate, a generic variable gadget for v_i is shown on the left-hand side of Fig. 5. The additional edges in this figure that are directed into vertices of the variable gadget and directed out of vertices of this gadget will be defined as part of the clause gadget construction which we describe next.

For a clause $C_j = (x_{3j-2} \vee x_{3j-1} \vee x_{3j})$, let i_j, i'_j, and i''_j be the elements in $\{1, 2, \ldots, n\}$ such that $x_{3j-2} \in \{v_{i_j}, \bar{v}_{i_j}\}$, $x_{3j-1} \in \{v_{i'_j}, \bar{v}_{i'_j}\}$, and $x_{3j} \in \{v_{i''_j}, \bar{v}_{i''_j}\}$. Now, for each $j \in \{1, 2, \ldots, m\}$, add the following vertices and edges to the variable gadgets.

1. Create the vertices $\{s_j^c, t_j^c, u_j, w_j, w'_j\}$.
2. Add the edges in $\{(s_j^c, u_j), (w_j, w'_j), (w'_j, t_j^c)\}$.
3. If $x_{3j-2} = v_{i_j}$, add the edges $(u_j, p_{3j-2}^{\text{in}})$ and $(p_{3j-2}^{\text{out}}, w_j)$. Otherwise, add the edges $(u_j, n_{3j-2}^{\text{in}})$ and $(n_{3j-2}^{\text{out}}, w_j)$.
4. If $x_{3j-1} = v_{i'_j}$, add the edges $(u_j, p_{3j-1}^{\text{in}})$ and $(p_{3j-1}^{\text{out}}, w_j)$. Otherwise, add the edges $(u_j, n_{3j-1}^{\text{in}})$ and $(n_{3j-1}^{\text{out}}, w_j)$.
5. If $x_{3j} = v_{i''_j}$, add the edges $(s_j^c, p_{3j}^{\text{in}})$ and $(p_{3j}^{\text{out}}, w'_j)$. Otherwise, add the edges $(s_j^c, n_{3j}^{\text{in}})$ and $(n_{3j}^{\text{out}}, w'_j)$.

In what follows, we refer to the edges and vertices that get added in the aforementioned five-step construction relative to a given C_j as the clause gadget for C_j. For each clause $C_j = (x_{3j-2} \vee x_{3j-1} \vee x_{3j})$, there are three directed paths from s_j^c to t_j^c each of which corresponds to one of the three literals in C_j. For example, for the first literal x_{3j-2}, there is a directed path from s_j^c to t_j^c that intersects with the edge $(p_{3j-2}^{\text{in}}, p_{3j-2}^{\text{out}})$ on $\pi_{i_j}^+$ if $x_{3j-2} = v_{i_j}$ and that intersects with the edge $(n_{3j-2}^{\text{in}}, n_{3j-2}^{\text{out}})$ on $\pi_{i_j}^-$ if $x_{3j-2} = \bar{v}_{i_j}$. To illustrate, assume that $x_{3j-2} = v_{i_j}$, $x_{3j-1} = \bar{v}_{i'_j}$, and $x_{3j} = v_{i''_j}$. For this specific case, the clause gadget for C_j is shown on the right-hand side of Fig. 5.

Now, let G be the directed graph that results from the construction of all variable and all clause gadgets. Observe that G is acyclic. We next set up an instance I' of ∀∃ Phylo-Directed-Disjoint-Connecting-Paths. Let $\mathcal{T}$ be the caterpillar $(\ell_1^v, \ell_2^v, \ldots, \ell_n^v, \ell_1^c, \ell_2^c, \ldots, \ell_m^c)$. We obtain a directed acyclic graph $\mathcal{N}$ from $\mathcal{T}$ and G by identifying ℓ_i^v with s_i^v for each $i \in \{1, 2, \ldots, n\}$ and identifying ℓ_j^c with s_j^c for each $j \in \{1, 2, \ldots, m\}$. Clearly, $\mathcal{N}$ is connected and has no parallel edges.

Moreover, except for the root, since each vertex of G has in-degree one and out-degree two, in-degree two and out-degree one, or in-degree one and out-degree zero, it follows that $\mathcal{N}$ is a phylogenetic network on $T = \{t_1^v, t_2^v, \ldots, t_n^v, t_1^c, t_2^c, \ldots, t_m^c\}$. Let $S = \{s_1^v, s_2^v, \ldots, s_n^v, s_1^c, s_2^c, \ldots, s_m^c\}$. Since every vertex of G that is not contained in S lies on a directed path from a child of a vertex in S to a leaf in $\mathcal{N}$, it follows that $\mathcal{N}$ is caterpillar-inducing with respect to S. Moreover, for each $i \in \{1, 2, \ldots, n\}$, there are exactly two directed paths from s_i^v to t_i^v in G_i^v and, hence, in $\mathcal{N}$ that only intersect in the vertices s_i^v, t_i^v, and y_i, and the edge (y_i, t_i^v). Recalling that $1 \leq p < n$, it follows from the construction that $\mathcal{N}$ has the two-path property relative to p, and that both $P^\forall$ and $P^\exists$ are non-empty. We now set

$$P^\forall = \{(s_1^v, t_1^v), (s_2^v, t_2^v), \ldots, (s_p^v, t_p^v)\} \text{ and}$$
$$P^\exists = \{(s_{p+1}^v, t_{p+1}^v), (s_{p+2}^v, t_{p+2}^v), \ldots, (s_n^v, t_n^v)\} \cup \{(s_1^c, t_1^c), (s_2^c, t_2^c), \ldots, (s_m^c, t_m^c)\}.$$

This completes the description of I'.

Since the number of vertices of G is $3n + 11m$, the number of vertices of $\mathcal{T}$ is $2(n + m) - 1$, and G and $\mathcal{T}$ have $n + m$ vertices in common, it follows that $\mathcal{N}$ has size $O(n + m)$ and can be constructed in polynomial time.

We complete the proof by establishing the following sublemma.

4.1.1. *The instance I is a yes-instance if and only if the instance I' is a yes-instance.*

Proof. First, suppose that I is a yes-instance. Let $\Pi^\forall = \{\pi_1^v, \pi_2^v, \ldots, \pi_p^v\}$ be a set of directed paths in $\mathcal{N}$ such that each π_i^v begins at s_i^v and ends at t_i^v. As $p < n$, we have $\pi_i^v \in \{\pi_i^+, \pi_i^-\}$. Moreover, since $\mathcal{N}$ has the two-path property relative to p, the paths in $\Pi^\forall$ are mutually vertex disjoint in $\mathcal{N}$. Now, let $\beta : V \to \{F, T\}$ be a truth assignment that satisfies each clause of Ψ such that, if $\pi_i^v = \pi_i^+$, then $v_i = F$ and, otherwise, $v_i = T$ for each $i \in \{1, 2, \ldots, p\}$. Since I is a yes-instance, β exists. We next construct a directed path for each pair of vertices in $P^\exists$ such that, collectively, these paths together with the elements in $\Pi^\forall$ form a solution to I'. For each $i \in \{p + 1, p + 2, \ldots, n\}$, set $\pi_i^v = \pi_i^+$ if $v_i = F$ and set $\pi_i^v = \pi_i^-$ if $v_i = T$. Furthermore, for each $j \in \{1, 2, \ldots, m\}$, let $x_{j'}$, with $j' \in \{3j - 2, 3j - 1, 3j\}$, be a literal in C_j that is satisfied by β, and let i be the element in $\{1, 2, \ldots, n\}$ such that $x_{j'} \in \{v_i, \bar{v}_i\}$. By construction of the clause gadget, there is a directed path, say π_j^c, from s_j^c to t_j^c in $\mathcal{N}$ such that one of the following properties applies.

(i) If $x_{j'} = v_i$, then π_j^c contains the edge $(p_{j'}^{\text{in}}, p_{j'}^{\text{out}})$.
(ii) If $x_{j'} = \bar{v}_i$, then π_j^c contains the edge $(n_{j'}^{\text{in}}, n_{j'}^{\text{out}})$.

In Case (i), as $v_i = T$, we have $\pi_i^v = \pi_i^-$, and it follows that π_j^c does not intersect π_i^v. Similar in Case (ii), as $v_i = F$, we have $\pi_i^v = \pi_i^+$, and it again follows that π_j^c does not intersect π_i^v. By construction of $\mathcal{N}$, it is now straightforward to check that

$$\Pi^\forall \cup \{\pi_{p+1}^v, \pi_{p+2}^v, \ldots, \pi_n^v, \pi_1^c, \pi_2^c, \ldots, \pi_m^c\}$$

is a collection of mutually vertex-disjoint directed-paths in $\mathcal{N}$ that connect each pair of vertices in $P^\forall \cup P^\exists$. In particular, since the argument presented in this paragraph applies to all choices of directed paths in $\Pi^\forall$, we conclude that I' is a yes-instance.

Second, suppose that I' is a yes-instance. Let $\beta^\forall : \{v_1, v_2, \ldots, v_p\} \to \{F, T\}$ be a truth assignment. Furthermore, let

$$\Pi = \{\pi_1^v, \pi_2^v, \ldots, \pi_p^v\} \cup \{\pi_{p+1}^v, \pi_{p+2}^v, \ldots, \pi_n^v, \pi_1^c, \pi_2^c, \ldots, \pi_m^c\}$$

be a collection of mutually vertex-disjoint directed paths in $\mathcal{N}$ such that $\pi_i^v = \pi_i^-$ if $v_i = T$ and $\pi_i^v = \pi_i^+$ if $v_i = F$ for each $i \in \{1, 2, \ldots, p\}$. Since I' is a yes-instance, Π exists. Now, let $\beta : V \to \{F, T\}$ such that

(i) for each $i \in \{1, 2, \ldots, p\}$, we have $\beta(v_i) = \beta^\forall(v_i)$ and,
(ii) for each $i \in \{p + 1, p + 2, \ldots, n\}$, we have $\beta(v_i) = F$ if $\pi_i^v = \pi_i^+$ and, $\beta(v_i) = T$ if $\pi_i^v = \pi_i^-$.

We next show that β satisfies each clause of Ψ. Let $C_j = (x_{3j-2} \vee x_{3j-1} \vee x_{3j})$ be a clause of Ψ with $j \in \{1, 2, \ldots, m\}$. Consider the directed path $\pi_j^c \in \Pi$ from s_j^c to t_j^c in $\mathcal{N}$. Let j' be the unique element in $\{3j - 2, 3j - 1, 3j\}$ such that π_j^c contains either the edge $(p_{j'}^{\text{in}}, p_{j'}^{\text{out}})$ or the edge $(n_{j'}^{\text{in}}, n_{j'}^{\text{out}})$, and let i be the element in $\{1, 2, \ldots, n\}$ such that $x_{j'} \in \{v_i, \bar{v}_i\}$. First, assume that π_j^c contains $(p_{j'}^{\text{in}}, p_{j'}^{\text{out}})$. Then, as $x_{j'} = v_i$ and the paths in Π are mutually vertex disjoint in $\mathcal{N}$, it follows that $\pi_i^v = \pi_i^-$. Hence $\beta(v_i) = T$. Second, assume that π_j^c contains $(n_{j'}^{\text{in}}, n_{j'}^{\text{out}})$. Then, as $x_{j'} = \bar{v}_i$ and the paths in Π are mutually vertex disjoint, it follows that $\pi_i^v = \pi_i^+$. Hence $\beta(v_i) = F$. Under both assumptions, β satisfies C_j because $\beta(x_{j'}) = T$. It now follows that β satisfies Ψ and, as the argument applies to all choices of truth assignments for the elements in $\{v_1, v_2, \ldots, v_p\}$, we conclude that I is a yes-instance. □

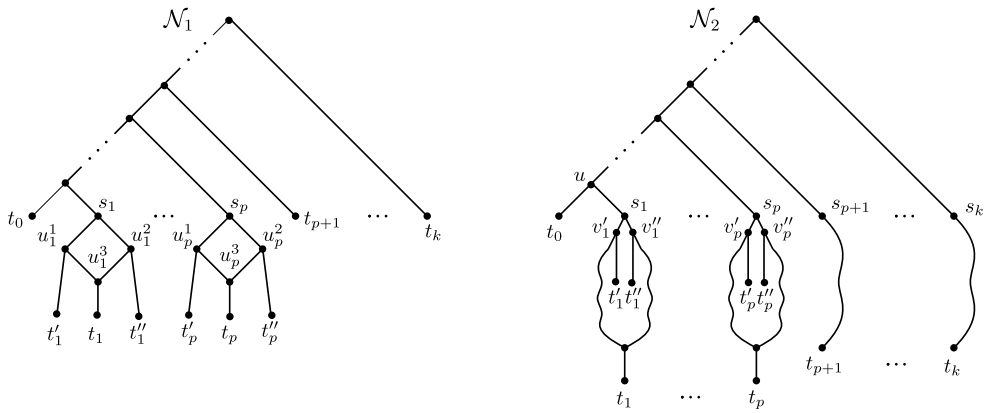


Fig. 6. The two phylogenetic networks $\mathcal{N}_1$ and $\mathcal{N}_2$ that are constructed in the proof of Theorem 4.3. For reasons of simplicity, not all edges of $\mathcal{N}_2$ are shown. In particular, each squiggly line is a directed path and, depending on the given instance of ∀∃ Phylo-Directed-Disjoint-Connecting-Paths, squiggly paths may intersect with each other and may be further interconnected by paths that are not shown.

This completes the proof of Theorem 4.1. □

While the next corollary is not needed for the remainder of the paper, it may be of independent interest in the theoretical computer science community.

Corollary 4.2. *The decision problem* ∀∃ Directed-Disjoint-Connecting-Paths *is* Π_2^P*-complete.*

Proof. Since every instance of ∀∃ Phylo-Directed-Disjoint-Connecting-Paths is also an instance of ∀∃ Directed-Disjoint-Connecting-Paths, it follows from Theorem 4.1 that the latter problem is Π_2^P-hard. To establish that ∀∃ Directed-Disjoint-Connecting-Paths is in Π_2^P, we use the same argument as in the first paragraph of the proof of Theorem 4.1 and, additionally, check in polynomial time if the paths in $\Pi^\forall$ are vertex disjoint. □

4.2. Display-Set-Containment *is* Π_2^P*-complete*

In this section, we show that Display-Set-Containment is complete for the second level of the polynomial hierarchy. This problem is a generalization of the well-known NP-complete Tree-Containment problem [7].

Theorem 4.3. Display-Set-Containment *is* Π_2^P*-complete.*

Proof. We first show that Display-Set-Containment is in Π_2^P. Let $\mathcal{N}$ and $\mathcal{N}'$ be two phylogenetic networks on X. To decide if $T(\mathcal{N}) \subseteq T(\mathcal{N}')$, let S be a switching of $\mathcal{N}$, and let $\mathcal{T}$ be the phylogenetic X-tree yielded by S. Then use an NP-oracle for Tree-Containment to decide if $\mathcal{T}$ is displayed by $\mathcal{N}'$. Since $\mathcal{N}$ and $\mathcal{N}'$ form a no-instance precisely if there exists some switching for $\mathcal{N}$ that yields a phylogenetic tree that is not displayed by $\mathcal{N}'$, it follows that Display-Set-Containment is in co-$\mathrm{NP}^{\mathrm{NP}} = \Pi_2^P$.

To complete the proof, we establish a reduction from ∀∃ Phylo-Directed-Disjoint-Connecting-Paths. Using the same notation as in the formal statement of ∀∃ Phylo-Directed-Disjoint-Connecting-Paths, let I be the following instance of this problem. Let $\mathcal{N}$ be a phylogenetic network on X, let $S = \{s_1, s_2, \ldots, s_k\}$ and $T = X = \{t_1, t_2, \ldots, t_k\}$ be two disjoint sets of vertices of $\mathcal{N}$, and let p be an integer with $1 \leq p < k$ such that $\mathcal{N}$ is caterpillar-inducing with respect to S and has the two-path property relative to p. Furthermore, let

$$P^\forall = \{(s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p)\},$$
$$P^\exists = \{(s_{p+1}, t_{p+1}), (s_{p+2}, t_{p+2}), \ldots, (s_k, t_k)\}$$

be two collections of pairs of elements in S and T. This completes the description of I.

Now, let $\mathcal{N}_1$ be the phylogenetic network obtained from the caterpillar $(t_0, s_1, s_2, \ldots, s_p, t_{p+1}, t_{p+2}, \ldots, t_k)$ by adding the following edges and vertices for each $i \in \{1, 2, \ldots, p\}$. Create three vertices u_i^1, u_i^2, and u_i^3 and add the set

$$\{(s_i, u_i^1), (s_i, u_i^2), (u_i^1, u_i^3), (u_i^2, u_i^3), (u_i^3, t_i), (u_i^1, t_i'), (u_i^2, t_i'')\}$$

of edges. Observe that the leaf set of $\mathcal{N}_1$ is

$$X' = \{t_0, t_1, t_2, \ldots, t_k\} \cup \{t_i', t_i'' : i \in \{1, 2, \ldots, p\}\}.$$

The construction of $\mathcal{N}_1$ is shown on the left-hand side of Fig. 6. We complete the reduction to an instance of Display-Set-Containment by describing a second phylogenetic network $\mathcal{N}_2$. For each $i \in \{1, 2, \ldots, p\}$, let w_i' and w_i'' be the two children of s_i in $\mathcal{N}$. As $\mathcal{N}$ has the two-path property relative to p, recall that there are exactly two directed paths from s_i to t_i in $\mathcal{N}$, and these two paths only have s_i, t_i, and the parent of t_i in common. In the remainder of the proof, we denote the directed path from s_i to t_i that contains w_i' with π_i' and, similarly, we denote the directed path from s_i to t_i that contains w_i'' with π_i''. Lastly, we denote the parent of s_1 with p_1. Now, obtain $\mathcal{N}_2$ from $\mathcal{N}$ in the following way.

(i) Subdivide the edge (p_1, s_1) with a new vertex u and add the edge (u, t_0).
(ii) For each $i \in \{1, 2, \ldots, p\}$, subdivide (s_i, w_i') with a new vertex v_i', subdivide (s_i, w_i'') with a new vertex v_i'', and add the two edges (v_i', t_i') and (v_i'', t_i'').

Clearly, the leaf set of $\mathcal{N}_2$ is X'. To illustrate, $\mathcal{N}_2$ is shown on the right-hand side in Fig. 6.

As the size of X' is polynomial in the size of X, it follows that the size of $\mathcal{N}_1$ and $\mathcal{N}_2$ is polynomial in the size of $\mathcal{N}$. Furthermore, the construction of $\mathcal{N}_1$ and $\mathcal{N}_2$ takes polynomial time.

4.3.1. *The instance I is a yes-instance if and only if $T(\mathcal{N}_1) \subseteq T(\mathcal{N}_2)$.*

Proof. First, suppose that I is a yes-instance. Let $\mathcal{T}'$ be a phylogenetic X'-tree that is displayed by $\mathcal{N}_1$. For each $i \in \{1, 2, \ldots, p\}$, note that $\mathcal{T}'$ contains one of the two caterpillars (t_i, t_i', t_i'') or (t_i, t_i'', t_i'). Let $\mathcal{J}'$ be the set that consists of each element $i \in \{1, 2, \ldots, p\}$ for which $\mathcal{T}'$ contains (t_i, t_i', t_i'') and, similarly, let $\mathcal{J}''$ be the set that consists of each element $i \in \{1, 2, \ldots, p\}$ for which $\mathcal{T}'$ contains (t_i, t_i'', t_i'). Furthermore, let $\Pi^\forall = \{\pi_1, \pi_2, \ldots, \pi_p\}$ be the set of directed paths in $\mathcal{N}$ such that $\pi_i = \pi_i'$ if $i \in \mathcal{J}'$ and $\pi_i = \pi_i''$ if $i \in \mathcal{J}''$. Since I is a yes-instance, there exists a set $\Pi = \Pi^\forall \cup \{\pi_{p+1}, \pi_{p+2}, \ldots, \pi_k\}$ of mutually vertex-disjoint directed paths in $\mathcal{N}$, where π_j is a directed path from s_j to t_j for each $j \in \{p+1, p+2, \ldots, k\}$. Moreover, as $\mathcal{N}$ is caterpillar-inducing with respect to S, it is straightforward to check that there exists a phylogenetic X-tree $\mathcal{T}$ such that the following three properties are satisfied:

(i) $\mathcal{T}$ is displayed by $\mathcal{N}$,
(ii) $\mathcal{T} = \mathcal{T}'|X$, and
(iii) there exists an embedding of $\mathcal{T}$ in $\mathcal{N}$ that contains all edges of paths in Π.

Let $E_\mathcal{T}$ be an embedding of $\mathcal{T}$ in $\mathcal{N}$ that satisfies (iii). By construction of $\mathcal{N}_2$ from $\mathcal{N}$, there exists an embedding of $\mathcal{T}$ in $\mathcal{N}_2$ whose set of edges is

$$\begin{aligned} E'_\mathcal{T} = (E_\mathcal{T} - (\{(p_1, s_1)\} \cup \{(s_i, w_i') : i \in \mathcal{J}'\} \cup \{(s_i, w_i'') : i \in \mathcal{J}''\})) \cup \\ \{(p_1, u), (u, s_1)\} \cup \{(s_i, v_i'), (v_i', w_i') : i \in \mathcal{J}'\} \cup \\ \{(s_i, v_i''), (v_i'', w_i'') : i \in \mathcal{J}''\}. \end{aligned}$$

For each $i \in \{1, 2, \ldots, p\}$, let E_i' be the subset $\{(v_i', t_i'), (v_i'', t_i''), (s_i, v_i'')\}$ of edges in $\mathcal{N}_2$ if $i \in \mathcal{J}'$, and the subset $\{(v_i'', t_i''), (v_i', t_i'), (s_i, v_i')\}$ of edges in $\mathcal{N}_2$ if $i \in \mathcal{J}''$. Since $E'_\mathcal{T}$ is an embedding of $\mathcal{T}$ in $\mathcal{N}_2$, it now follows that

$$E'_\mathcal{T} \cup E_1' \cup E_2' \cup \cdots \cup E_p' \cup \{(u, t_0)\}$$

is an embedding of $\mathcal{T}'$ in $\mathcal{N}_2$. Hence, $T(\mathcal{N}_1) \subseteq T(\mathcal{N}_2)$.

Second, suppose that I is a no-instance. Throughout this part of the proof, we use π_i to denote a directed path from s_i to t_i in $\mathcal{N}$ for each $i \in \{1, 2, \ldots, k\}$. Then, as $\mathcal{N}$ has the two-path property relative to p, there is a set $\Pi^\forall = \{\pi_1, \pi_2, \ldots, \pi_p\}$ of mutually vertex-disjoint directed paths in $\mathcal{N}$ for which every set $\Pi = \Pi^\forall \cup \{\pi_{p+1}, \pi_{p+2}, \ldots, \pi_k\}$ of directed paths in $\mathcal{N}$ contains two elements that are not vertex disjoint. For each $i \in \{1, 2, \ldots, k\}$, let E_i be the set of edges of π_i in $\mathcal{N}$. Furthermore, for each $i \in \{1, 2, \ldots, p\}$, let E_i' be the subset

$$(E_i - \{(s_i, w_i')\}) \cup \{(s_i, v_i'), (v_i', w_i'), (v_i', t_i'), (s_i, v_i''), (v_i'', t_i'')\}$$

of edges in $\mathcal{N}_2$ if $\pi_i = \pi_i'$, and the subset

$$(E_i - \{(s_i, w_i'')\}) \cup \{(s_i, v_i''), (v_i'', w_i''), (v_i'', t_i''), (s_i, v_i'), (v_i', t_i')\}$$

of edges in $\mathcal{N}_2$ if $\pi_i = \pi_i''$, where π_i' or π_i'' are as described in the construction of $\mathcal{N}_2$ from $\mathcal{N}$. Clearly, there is a phylogenetic tree $\mathcal{T}_p$ with leaf set $\{t_i, t_i', t_i'' : i \in \{1, 2, \ldots, p\}\}$ for which there exists an embedding in $\mathcal{N}_2$ that contains all edges in $E_1' \cup E_2' \cup \cdots \cup E_p'$. Observe that $\mathcal{T}_p$ can be obtained from the caterpillar $(\ell_1, \ell_2, \ldots, \ell_p)$ by replacing each $\ell_i \in \{\ell_1, \ell_2, \ldots, \ell_p\}$ with the caterpillar (t_i, t_i', t_i'') if $\pi_i = \pi_i'$ and with the caterpillar (t_i, t_i'', t_i') if $\pi_i = \pi_i''$. By construction, it now follows that $\mathcal{N}_1$ displays $\mathcal{T}_p$. Let $\mathcal{T}$ be the unique phylogenetic X'-tree that is displayed by $\mathcal{N}_1$ such that $\mathcal{T}|\{t_i, t_i', t_i'' : i \in \{1, 2, \ldots, p\}\} = \mathcal{T}_p$.

We complete the argument by showing that $\mathcal{T}$ is not displayed by $\mathcal{N}_2$. Towards a contradiction, assume that $\mathcal{T}$ is displayed by $\mathcal{N}_2$. Let $E'_{\mathcal{T}}$ be an embedding of $\mathcal{T}$ in $\mathcal{N}_2$. Then, since $\mathcal{T}$ contains (t_i, t'_i, t''_i) or (t_i, t''_i, t'_i) for each $i \in \{1, 2, \ldots, p\}$ and $\mathcal{N}$ satisfies the two-path property relative to p, it follows from the construction of $\mathcal{N}_2$ that $E'_{\mathcal{T}}$ contains all edges in $E'_1 \cup E'_2 \cup \cdots \cup E'_p$. Furthermore, observe that there is a unique directed path from the root, say ρ, of $\mathcal{N}_2$ to t_0, and so the edges on this path are elements of $E'_{\mathcal{T}}$. For each pair i and i' of distinct elements in $\{1, 2, \ldots, k\}$, it therefore follows that the directed path from ρ to t_i in $E'_{\mathcal{T}}$ and the directed path from ρ to $t_{i'}$ in $E'_{\mathcal{T}}$ only intersect in vertices that are ancestors of t_0 in $\mathcal{N}_2$. Hence, as $\mathcal{N}_2$ is caterpillar-inducing with respect to S, there exist directed paths $\pi_1^*, \pi_2^*, \ldots, \pi_p^*, \pi_{p+1}^*, \ldots, \pi_k^*$ in $E'_{\mathcal{T}}$ such that the following three properties are fulfilled.

(i) For each $i \in \{1, 2, \ldots, p\}$, π_i^* is the unique directed path from s_i to t_i in $\mathcal{N}_2$ that contains v'_i if $\pi_i = \pi'_i$ and that contains v''_i if $\pi_i = \pi''_i$.
(ii) For each $i \in \{p+1, p+2, \ldots, k\}$, π_i^* is a directed path from s_i to t_i in $\mathcal{N}_2$.
(iii) The elements in $\Pi^* = \{\pi_1^*, \pi_2^*, \ldots, \pi_k^*, \}$ are mutually vertex disjoint.

Now, by construction, observe that π_i^* is also a directed path from s_i to t_i in $\mathcal{N}$ for each $i \in \{p+1, p+2, \ldots, k\}$. As Π^* is a set of mutually vertex-disjoint directed paths in $\mathcal{N}_2$, it now follows that, $\Pi^\forall \cup \{\pi_{p+1}^*, \pi_{p+2}^*, \ldots, \pi_k^*\}$ is a set of mutually vertex-disjoint directed paths in $\mathcal{N}$. In turn, this implies that I is a yes-instance; a contradiction. Hence, $\mathcal{T} \notin T(\mathcal{N}_2)$, and so $T(\mathcal{N}_1) \not\subseteq T(\mathcal{N}_2)$. □

This establishes Theorem 4.3. □

We end this section with a brief discussion of the structural properties of the phylogenetic network $\mathcal{N}_1$ that is constructed in the proof of Theorem 4.3. These properties will play an important role in the next section when we establish Π_2^P-completeness of Display-Set-Equivalence. Let $\mathcal{N}$ be a phylogenetic network on X. We say that $\mathcal{N}$ is a *caterpillar network* if it can be obtained from a caterpillar $(\ell_1, \ell_2, \ldots, \ell_k)$ with $2 \leq k \leq |X|$ by replacing each ℓ_i with a phylogenetic network $\mathcal{N}_i$ on X_i such that the elements in $\{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k\}$ are pairwise vertex disjoint and

$$\bigcup_{i=1}^{k} X_i = X.$$

By construction, $\mathcal{N}_1$ is a caterpillar network. Moreover, it is easily seen that $\mathcal{N}_1$ is temporal and tree-child.

The next corollary now immediately follows from Theorem 4.3.

Corollary 4.4. *Let $\mathcal{N}_1$ be a temporal tree-child caterpillar network on X, and let $\mathcal{N}_2$ be a phylogenetic network on X. Then deciding whether $T(\mathcal{N}_1) \subseteq T(\mathcal{N}_2)$ is Π_2^P-complete.*

4.3. Display-Set-Equivalence is Π_2^P-complete

With the result of Corollary 4.4 in hand, we are now in a position to establish the main result of Section 4 which is the following theorem.

Theorem 4.5. Display-Set-Equivalence *is Π_2^P-complete.*

Proof. Let $\mathcal{N}$ and $\mathcal{N}'$ be two phylogenetic networks on X. By Theorem 4.3, the problem of deciding whether or not $T(\mathcal{N}) \subseteq T(\mathcal{N}')$ is in Π_2^P. Similarly, the problem of deciding whether or not $T(\mathcal{N}') \subseteq T(\mathcal{N})$ is in Π_2^P. Hence, Display-Set-Equivalence is in Π_2^P.

We next establish a polynomial-time reduction from Display-Set-Containment to Display-Set-Equivalence. Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be two phylogenetic networks on $X = \{\ell_1, \ell_2, \ldots, \ell_n\}$ that form the input to an instance of Display-Set-Containment that asks if $T(\mathcal{N}_1) \subseteq T(\mathcal{N}_2)$. By Corollary 4.4, we may assume that $\mathcal{N}_1$ is a caterpillar network. Then there exist two vertex-disjoint phylogenetic networks $\mathcal{M}_1$ and $\mathcal{M}_{1'}$ with leaf sets W_1 and $W_{1'}$, respectively, such that $W_1 \cup W_{1'} = X$, and $\mathcal{N}_1$ can be obtained from the caterpillar $\{x_1, x_2\}$ by replacing x_1 with $\mathcal{M}_1$ and x_2 with $\mathcal{M}_{1'}$. To ease reading, let $\mathcal{N}'_1$ and $\mathcal{N}'_2$ be the two phylogenetic networks on $X' = \{\ell'_1, \ell'_2, \ldots, \ell'_n\}$ that are obtained from $\mathcal{N}_1$ and $\mathcal{N}_2$, respectively, by replacing ℓ_i with ℓ'_i in both networks for each $i \in \{1, 2, \ldots, n\}$. Similarly, let $\mathcal{M}'_1$ and $\mathcal{M}'_{1'}$ be the two phylogenetic networks obtained from $\mathcal{M}_1$ and $\mathcal{M}_{1'}$, respectively, by replacing ℓ_i with ℓ'_i in exactly one of $\mathcal{M}_1$ and $\mathcal{M}_{1'}$ for each $i \in \{1, 2, \ldots, n\}$. If W'_1 (resp. $W'_{1'}$) denotes the leaf set of $\mathcal{M}'_1$ (resp. $\mathcal{M}'_{1'}$), then $W'_1 \cup W'_{1'} = X'$.

Set $\mathcal{T}$ as well as $\mathcal{T}'$ to be the caterpillar $(w_1, w_2, \ldots, w_{2n+3})$. Furthermore, let $u_{2n+3}, u_{2n+2}, \ldots, u_2$ be the directed path in $\mathcal{T}$ (and $\mathcal{T}'$) such that, for all $j \in \{2, 3, \ldots, 2n+3\}$, u_j is the parent of w_j. Now, let G_1^* and G_2^* be the two directed acyclic graphs that are obtained from $\mathcal{T}$ and $\mathcal{T}'$, respectively, by applying the following six-step process.

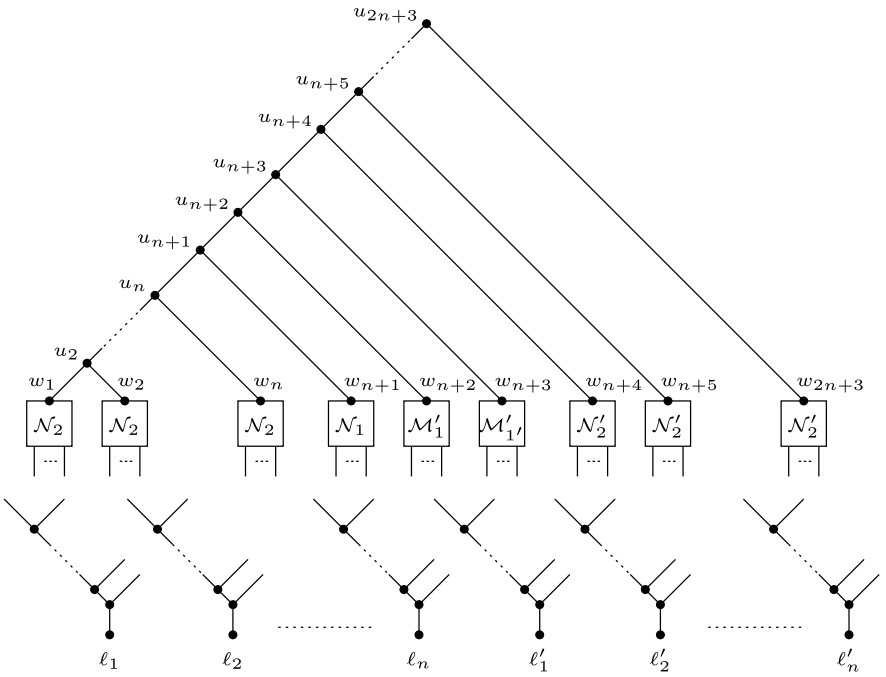


Fig. 7. The phylogenetic network $\mathcal{N}_1^*$ on $X \cup X'$ as constructed in the proof of Theorem 4.5. Each of the squares labeled $\mathcal{N}_2$, $\mathcal{N}_1$, $\mathcal{M}'_1$, $\mathcal{M}'_{1'}$, and $\mathcal{N}'_2$ refers to the network obtained from its namesake phylogenetic network by deleting all leaf labels. The dangling edges of each square are paired up with the edges shown in the bottom part of the figure as described in Step 6 of the construction in the proof of Theorem 4.5.

1. For all $j \in \{1, 2, \ldots, n\}$, replace w_j with $\mathcal{N}_2$ in $\mathcal{T}$ and $\mathcal{T}'$ by identifying w_j with the root of $\mathcal{N}_2$.
2. Replace w_{n+1} with the root of $\mathcal{N}_1$ in $\mathcal{T}$ by identifying w_{n+1} with the root of $\mathcal{N}_1$, and replace w_{n+1} with the root of $\mathcal{M}_1$ in $\mathcal{T}'$ by identifying w_{n+1} with the root of $\mathcal{M}_1$
3. Replace w_{n+2} with $\mathcal{M}'_1$ in $\mathcal{T}$ by identifying w_{n+2} with the root of $\mathcal{M}'_1$, and replace w_{n+2} with $\mathcal{M}_{1'}$ in $\mathcal{T}'$ by identifying w_{n+2} with the root of $\mathcal{M}_{1'}$
4. Replace w_{n+3} with $\mathcal{M}'_{1'}$ in $\mathcal{T}$ by identifying w_{n+3} with the root of $\mathcal{M}'_{1'}$, and replace w_{n+3} with $\mathcal{N}'_1$ in $\mathcal{T}'$ by identifying w_{n+3} with the root of $\mathcal{N}'_1$.
5. For all $j \in \{n+4, n+5, \ldots, 2n+3\}$, replace w_j with $\mathcal{N}'_2$ in $\mathcal{T}$ and $\mathcal{T}'$ by identifying w_j with the root of $\mathcal{N}'_2$.
6. For each $i \in \{1, 2, \ldots, n\}$, identify all leaves labeled ℓ_i (resp. ℓ'_i) in $\mathcal{T}$ with a new vertex v_i (resp. v'_i), add a new edge (v_i, ℓ_i) (resp. (v'_i, ℓ'_i)). Do the same for all leaves labeled ℓ_i (resp. ℓ'_i) in $\mathcal{T}'$.

To complete the construction, let $\mathcal{N}_1^*$ and $\mathcal{N}_2^*$ be two phylogenetic networks such that G_1^* and G_2^* can be obtained from $\mathcal{N}_1^*$ and $\mathcal{N}_2^*$, respectively, by contracting edges. Clearly, the leaf set of $\mathcal{N}_1^*$ and $\mathcal{N}_2^*$ is $X \cup X'$. Moreover, the directed path $u_{2n+3}, u_{2n+2}, \ldots, u_2$ of $\mathcal{T}$ and $\mathcal{T}'$ is also a directed path of $\mathcal{N}_1^*$ and $\mathcal{N}_2^*$. We refer to this path as the *backbone* of $\mathcal{N}_1^*$ and $\mathcal{N}_2^*$. The phylogenetic networks $\mathcal{N}_1^*$ and $\mathcal{N}_2^*$ are shown in Figs. 7 and 8, respectively. Lastly, observe that the size of both $\mathcal{N}_1^*$ and $\mathcal{N}_2^*$ is $O(n(|E_1| + |E_2|))$, where E_1 and E_2 is the edge set of $\mathcal{N}_1$ and $\mathcal{N}_2$, respectively. Hence, the construction of $\mathcal{N}_1^*$ and $\mathcal{N}_2^*$ takes polynomial time.

4.5.1. *$T(\mathcal{N}_1) \subseteq T(\mathcal{N}_2)$ if and only if $T(\mathcal{N}_1^*) = T(\mathcal{N}_2^*)$.*

Proof. Throughout this proof, let $U = \{u_2, u_3, \ldots, u_{2n+3}\}$ be the vertex set of the backbone of $\mathcal{N}_1^*$ and $\mathcal{N}_2^*$, and let

$$E_U = \{(u_2, w_1), (u_2, w_2), (u_3, w_3), \ldots, (u_{2n+3}, w_{2n+3})\}$$

be the set of edges in $\mathcal{N}_1^*$ and $\mathcal{N}_2^*$ that are directed from a vertex in U to a vertex not in U. Furthermore, for a vertex v and an embedding E, we say that v is *in* E if there exists an edge in E that is incident with v. If v is in E, then we denote this by $v \in E$.

First, suppose that $T(\mathcal{N}_1) \nsubseteq T(\mathcal{N}_2)$. Let $\mathcal{T}_1$ be a phylogenetic X-tree such that $\mathcal{T}_1 \in T(\mathcal{N}_1)$ and $\mathcal{T}_1 \notin T(\mathcal{N}_2)$. Let $\mathcal{T}'_1$ be the phylogenetic X'-tree obtained from $\mathcal{T}_1$ by replacing ℓ_i with ℓ'_i for each $i \in \{1, 2, \ldots, n\}$. Furthermore, let $\mathcal{T}$ be the phylogenetic $(X \cup X')$-tree obtained from $\mathcal{T}_1$ and $\mathcal{T}'_1$ by creating a new vertex ρ, adding an edge that joins ρ with the root of $\mathcal{T}_1$, and adding an edge that joins ρ with the root of $\mathcal{T}'_1$. As $\mathcal{N}_1$ displays $\mathcal{T}_1$ and $\mathcal{N}'_1$ displays $\mathcal{T}'_1$, it is easy to check that an embedding of $\mathcal{T}$ in $\mathcal{N}_2^*$ can be obtained from adding edges of $\mathcal{N}_2^*$ to

$$\{(u_{n+3}, u_{n+2}), (u_{n+2}, u_{n+1}), (u_{n+1}, w_{n+1}), (u_{n+2}, w_{n+2}), (u_{n+3}, w_{n+3})\}$$

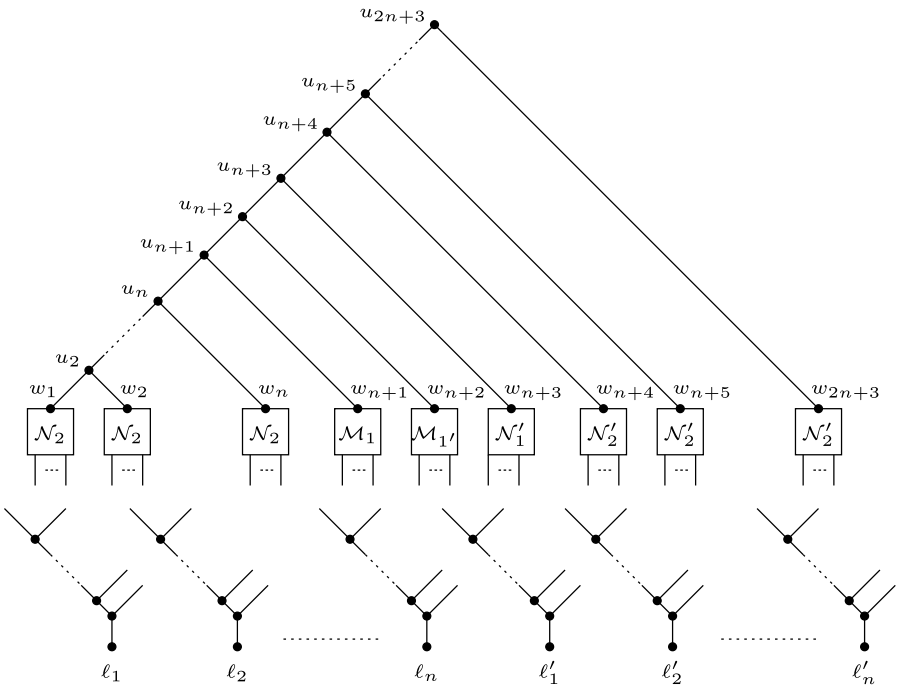


Fig. 8. The phylogenetic network $\mathcal{N}_2^*$ on $X \cup X'$ as constructed in the proof of Theorem 4.5. Each of the squares labeled $\mathcal{N}_2$, $\mathcal{M}_1$, $\mathcal{M}_{1'}$, $\mathcal{N}'_1$ and $\mathcal{N}'_2$ refers to the network obtained from its namesake phylogenetic network by deleting all leaf labels. The dangling edges of each square are paired up with the edges shown in the bottom part of the figure as described in Step 6 of the construction in the proof of Theorem 4.5.

such that each element in X is a descendant of u_{n+2}, each element in X' is a descendant of w_{n+3}. Hence, $\mathcal{T}$ is displayed by $\mathcal{N}_2^*$.

We next show that $\mathcal{T}$ is not displayed by $\mathcal{N}_1^*$. Towards a contradiction, assume that $\mathcal{T}$ is displayed by $\mathcal{N}_1^*$. Let E_1 be an embedding of $\mathcal{T}$ in $\mathcal{N}_1^*$. Furthermore, let k be the maximum element in $\{1, 2, \ldots, 2n+3\}$ such that $w_k \in E_1$. By construction of $\mathcal{T}$, either each element in X is a descendant of w_k in E_1 or each element in X' is a descendant of w_k in E_1. Thus, as $\mathcal{N}_2$ does not display $\mathcal{T}_1$ and $\mathcal{N}'_2$ does not display $\mathcal{T}'_1$, we have $k = n+1$. In particular, each element in X is a descendant of w_k in E_1. But no element in X' is a descendant of u_k in E_1; a contradiction. Hence, $\mathcal{T}$ is not displayed by $\mathcal{N}_1^*$, and so $T(\mathcal{N}_1^*) \neq T(\mathcal{N}_2^*)$.

Second, suppose that $T(\mathcal{N}_1) \subseteq T(\mathcal{N}_2)$. Let $\mathcal{T}$ be a phylogenetic $(X \cup X')$-tree that is displayed by $\mathcal{N}_1^*$, and let E_1 be an embedding of $\mathcal{T}$ in $\mathcal{N}_1^*$. For each $j \in \{1, 2, \ldots, 2n+3\}$ with $w_j \in E_1$, let Y_j be the set that consists of all leaves that are descendants of w_j in E_1, and let $\mathcal{T}_j$ be the phylogenetic tree obtained from the minimal rooted subtree of E_1 that connects all leaves in Y_j by suppressing all vertices with in-degree one and out-degree one. If $w_{n+1} \in E_1$, then, by the pigeonhole principle, there exists an element $j \in \{1, 2, \ldots, n\}$ such that $w_j \notin E_1$. Similarly, if $w_{n+3} \in E_1$, then there exists an element $j' \in \{n+4, n+5, \ldots, 2n+3\}$ such that $w_{j'} \notin E_1$. Without loss of generality, we may therefore assume by the construction of $\mathcal{N}_1^*$ that E_1 satisfies the following property.

(P) If $w_{n+1} \in E_1$, then $w_n \notin E_1$ and, if $w_{n+3} \in E_1$, then $w_{n+4} \notin E_1$.

Intuitively, property (P) allows enough 'play' so that any embedding of a phylogenetic $(X \cup X')$-tree in N_1^* can be replicated in N_2^*.

Since $T(\mathcal{N}_1) \subseteq T(N_2)$, each tree in $T(\mathcal{N}_1)$ is displayed by $\mathcal{N}_2$, and so each tree in $T(\mathcal{M}'_1)$ is displayed by $\mathcal{N}'_1$, and each tree in $T(\mathcal{M}'_{1'})$ is displayed by $\mathcal{N}'_2$. Hence, there exists an embedding E_2 of a phylogenetic $(X \cup X')$-tree in $\mathcal{N}_2^*$ such that the following conditions are satisfied:

(i) For each $j \in \{1, 2, \ldots, n, n+4, n+5, \ldots, 2n+3\}$, if $w_j \in E_1$, then w_j is the root of a subtree in E_2 that is a subdivision of $\mathcal{T}_j$.
(ii) If $w_{n+1} \in E_1$ and so $w_n \notin E_1$, then w_n is the root of a subtree in E_2 that is a subdivision of $\mathcal{T}_{n+1}$.
(iii) If $w_{n+3} \in E_1$ and so $w_{n+4} \notin E_1$, then w_{n+4} is the root of a subtree in E_2 that is a subdivision of $\mathcal{T}_{n+3}$.
(iv) If $w_{n+2} \in E_1$, then w_{n+3} is the root of a subtree in E_2 that is a subdivision of $\mathcal{T}_{n+2}$.

Since E_1 satisfies (P), E_2 is well defined. By construction of $\mathcal{N}_1^*$ and $\mathcal{N}_2^*$, it now follows that the edges in E_2 are an embedding of $\mathcal{T}$ in $\mathcal{N}_2^*$. Thus $T(\mathcal{N}_1^*) \subseteq T(\mathcal{N}_2^*)$.

Now, let $\mathcal{T}$ be a phylogenetic $(X \cup X')$-tree that is displayed by $\mathcal{N}_2^*$. To see that $\mathcal{T}$ is displayed by $\mathcal{N}_1^*$, we can effectively use the same argument as the one to show that $T(\mathcal{N}_1^*) \subseteq T(\mathcal{N}_2^*)$ even though the assumption that $T(\mathcal{N}_1) \subseteq T(\mathcal{N}_2)$ is not

symmetric. In particular, let E_2 be an embedding of $\mathcal{T}$ in $\mathcal{N}_2^*$. For each $j \in \{1, 2, \ldots, 2n+3\}$ with $w_j \in E_2$, let Z_j be the set that consists of all leaves that are descendants of w_j in E_2, and let T_j be the phylogenetic tree obtained from the minimal rooted subtree of E_2 that connects all the leaves in Z_j by suppressing all vertices with in-degree one and out-degree one. If $w_{n+1} \in E_2$, then, by the pigeonhole principle, there exists an element $j \in \{1, 2, \ldots, n\}$ such that $w_j \notin E_2$. Similarly, if $w_{n+3} \in E_2$, then there exists an element $j' \in \{n+4, n+5, \ldots, 2n+3\}$ such that $w_{j'} \notin E_2$. Thus, by construction, we may assume without loss of generality that

(P)′ if $w_{n+1} \in E_2$, then $w_n \notin E_2$ and, if $w_{n+3} \in E_2$, then $w_{n+4} \notin E_2$.

As $T(\mathcal{N}_1) \subseteq T(N_2)$, each tree in $T(\mathcal{M}_1)$ is displayed by $\mathcal{N}_2$, each tree in $T(\mathcal{M}_{1'})$ is displayed by $\mathcal{N}_1$, and each tree in $T(\mathcal{N}_1')$ is displayed by $\mathcal{N}_2'$. Thus there exists an embedding E_1 of a phylogenetic $(X \cup X')$-tree in $\mathcal{N}_1^*$ satisfying the following conditions:

(i)′ For each $j \in \{1, 2, \ldots, n, n+4, n+5, \ldots, 2n+3\}$, if $w_j \in E_2$, then w_j is the root of a subtree in E_1 that is a subdivision of $\mathcal{T}_j$.
(ii)′ If $w_{n+1} \in E_2$ and so $w_n \notin E_2$, then w_n is the root of a subtree in E_1 that is a subdivision of $\mathcal{T}_{n+1}$.
(iii)′ If $w_{n+3} \in E_2$ and so $w_{n+4} \notin E_2$, then w_{n+4} is the root of a subtree of E_1 that is a subdivision of $\mathcal{T}_{n+3}$.
(iv)′ If $w_{n+2} \in E_2$, then w_{n+1} is the root of a subtree in E_1 that is a subdivision of $\mathcal{T}_{n+2}$.

It is now easily checked that E_1 is well defined and, by construction of $\mathcal{N}_1^*$ and $\mathcal{N}_2^*$, is an embedding of $\mathcal{T}$ in $\mathcal{N}_1^*$. So $T(\mathcal{N}_2^*) \subseteq T(\mathcal{N}_1^*)$. Combining both cases establishes that $T(\mathcal{N}_2^*) = T(\mathcal{N}_1^*)$. □

This completes the proof of Theorem 4.5. □

5. Conclusion

We end this paper, with two corollaries that are implied by the results presented in Section 3 and two open problems. In 2015, Francis and Steel [4] introduced tree-based networks. A phylogenetic network $\mathcal{N}$ on X is *tree-based* if, up to suppressing vertices of in-degree one and out-degree one, $\mathcal{N}$ displays a phylogenetic X-tree $\mathcal{T}$ that can be obtained by only deleting reticulation edges, in which case, $\mathcal{T}$ is a *base tree* of $\mathcal{N}$. If $\mathcal{N}$ is tree-based, it is well known that not every phylogenetic X-tree displayed by $\mathcal{N}$ is a base tree. However, noting that each tree-child network is also a tree-based network, it is shown in [12] that a phylogenetic tree $\mathcal{T}$ is displayed by a tree-child network $\mathcal{N}$ if and only if $\mathcal{T}$ is a base tree of $\mathcal{N}$. Hence, for two tree-child networks $\mathcal{N}$ and $\mathcal{N}'$, the problem of deciding whether or not $T(\mathcal{N}) \cap T(\mathcal{N}') \neq \emptyset$ is equivalent to deciding whether or not $\mathcal{N}$ and $\mathcal{N}'$ have a common base tree.

Corollary 5.1. *Let $\mathcal{N}$ and $\mathcal{N}'$ be two tree-based networks on X. Then deciding if $\mathcal{N}$ and $\mathcal{N}'$ have a common base tree is* NP*-complete.*

Proof. Let S be a switching of $\mathcal{N}$, and let $\mathcal{T}$ be a phylogenetic X-tree. We say that S is a *base-tree switching* if, for each non-leaf vertex u in $\mathcal{N}$ that is the parent of only reticulations, there exists an edge (u, v) in S. By the definition of a tree-based network it follows that $\mathcal{T}$ is a base tree of $\mathcal{N}$ if and only if there exists a base-tree switching S of $\mathcal{N}$ that yields $\mathcal{T}$. Now, let S be a switching of $\mathcal{N}$, and let S' be a switching of $\mathcal{N}'$. If S is a base-tree switching of $\mathcal{N}$ and S' is a base-tree switching of $\mathcal{N}'$, and S and S' yield the same tree, then $\mathcal{N}$ and $\mathcal{N}'$ have a common base tree. Since it can be checked in polynomial time if S (resp. S') is a base-tree switching of $\mathcal{N}$ (resp. $\mathcal{N}'$), and if S and S' yield the same tree, it follows that deciding whether or not $\mathcal{N}$ and $\mathcal{N}'$ have a common base tree is in NP. The corollary now follows from Theorem 3.2. □

Using (ordinary) switchings instead of base-tree switching, ideas analogous to the ones described in the proof of Corollary 5.1 can be used to show that Common-Tree-Containment is in NP for two arbitrary phylogenetic networks. The next corollary is now an immediate consequence of Theorem 3.2.

Corollary 5.2. Common-Tree-Containment *is* NP*-complete for two arbitrary phylogenetic networks.*

In possible contrast to the last two corollaries and, in particular, Theorem 3.2, we leave it as an open problem to decide on the complexity of Common-Tree-Containment for two level-one networks.

Lastly, let C be a class of phylogenetic networks for which Tree-Containment is solvable in polynomial time such as tree-child or, more generally, reticulation-visible networks [1,6,14]. Furthermore, let $\mathcal{N}$ and $\mathcal{N}'$ be two networks in C. Then deciding if $T(\mathcal{N}) = T(\mathcal{N}')$ is in co-NP because, given a tree $\mathcal{T}$ that is displayed by $\mathcal{N}$ or $\mathcal{N}'$, it can be checked in polynomial time, if $\mathcal{T}$ is also displayed by the other network. If this is not the case, then $\mathcal{N}$ and $\mathcal{N}'$ form a no-instance of Display-Set-Equivalence. Whether Display-Set-Equivalence for $\mathcal{N}$ and $\mathcal{N}'$ is co-NP-complete remains an open problem. Nevertheless, it is unlikely that Display-Set-Equivalence for $\mathcal{N}$ and $\mathcal{N}'$ is Π_2^P-complete since a problem that is Π_2^P-complete and in co-NP would imply that co-NP $= \Pi_2^P$ which, in turn, would result in a collapse of the polynomial hierarchy to the first level.

Acknowledgements

We thank Britta Dorn for insightful discussions as well as the anonymous referees for their careful reading of the paper. The second and third authors thank the New Zealand Marsden Fund for their financial support.

References

[1] M. Bordewich, C. Semple, Reticulation-visible networks, Adv. Appl. Math. 76 (2016) 114–141.
[2] G. Cardona, F. Rosselló, G. Valiente, Comparison of tree-child phylogenetic networks, IEEE/ACM Trans. Comput. Biol. Bioinform. 6 (2009) 552–569.
[3] J. Döcker, S. Linz, C. Semple, Display sets of normal and tree-child networks, submitted for publication. A preprint is available at arXiv:1901.06725.
[4] A. Francis, M. Steel, Which phylogenetic networks are merely trees with additional arcs?, Syst. Biol. 64 (2015) 768–777.
[5] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Company, 1979.
[6] A.D.M. Gunawan, B. DasGupta, L. Zhang, A decomposition theorem and two algorithms for reticulation-visible networks, Inf. Comput. 252 (2017) 161–175.
[7] I.A. Kanj, L. Nakhleh, C. Than, G. Xia, Seeing the trees and their branches in the network is hard, Theor. Comput. Sci. 401 (2008) 153–164.
[8] S. Khuller, Design and analysis of algorithms: course notes, available at https://drum.lib.umd.edu/bitstream/handle/1903/592/CS-TR-3113.ps?sequence=1, 1994.
[9] C. McDiarmid, C. Semple, D. Welsh, Counting phylogenetic networks, Ann. Comb. 19 (2015) 205–224.
[10] B.M.E. Moret, L. Nakhleh, T. Warnow, C.R. Linder, A. Tholse, A. Padolina, J. Sun, R. Timme, Phylogenetic networks: modeling, reconstructibility, and accuracy, IEEE/ACM Trans. Comput. Biol. Bioinform. 1 (2004) 13–23.
[11] Y. Perl, Y. Shiloach, Finding two disjoint paths between two pairs of vertices in a graph, J. Assoc. Comput. Mach. 25 (1978) 1–9.
[12] C. Semple, Phylogenetic networks with every embedded phylogenetic tree a base tree, Bull. Math. Biol. 78 (2016) 132–137.
[13] L.J. Stockmeyer, The polynomial-time hierarchy, Theor. Comput. Sci. 3 (1976) 1–22.
[14] L. van Iersel, C. Semple, M. Steel, Locating a tree in a phylogenetic network, Inf. Process. Lett. 110 (2010) 1037–1043.
[15] M. Weller, Linear-time tree containment in phylogenetic networks, in: M. Blanchette, A. Ouangraoua (Eds.), RECOMB-CG 2018, in: Lecture Notes in Computer Science, vol. 11183, Springer, Cham, 2018.
[16] S.J. Willson, Properties of normal phylogenetic networks, Bull. Math. Biol. 72 (2010) 340–358.
[17] S.J. Willson, Tree-average distances on certain phylogenetic networks have their weights uniquely determined, Algorithms Mol. Biol. 7 (2012) 13.

1.4 Display sets of normal and tree-child networks

The following paper [DLS21] is also available online at the following URL: `https://doi.org/10.37236/9128`.

Display sets of normal and tree-child networks

Janosch Döcker
Department of Computer Science
University of Tübingen
Tübingen, Germany
janosch.doecker@uni-tuebingen.de

Simone Linz *
School of Computer Science
University of Auckland
Auckland, New Zealand
s.linz@auckland.ac.nz

Charles Semple †
School of Mathematics and Statistics
University of Canterbury
Christchurch, New Zealand
charles.semple@canterbury.ac.nz

Submitted: Nov 9, 2019; Accepted: Dec 7, 2020; Published: Jan 15, 2021

Abstract

Phylogenetic networks are leaf-labelled directed acyclic graphs that are used in computational biology to analyse and represent the evolutionary relationships of a set of species or viruses. In contrast to phylogenetic trees, phylogenetic networks have vertices of in-degree at least two that represent reticulation events such as hybridisation, lateral gene transfer, or reassortment. By systematically deleting various combinations of arcs in a phylogenetic network $\mathcal{N}$, one derives a set of phylogenetic trees that are embedded in $\mathcal{N}$. We recently showed that the problem of deciding if two binary phylogenetic networks embed the same set of phylogenetic trees is computationally hard, in particular, we showed it to be Π_2^P-complete. In this paper, we establish a polynomial-time algorithm for this decision problem if the initial two networks consist of a normal network and a tree-child network; two well-studied topologically restricted subclasses of phylogenetic networks, with normal networks being more structurally constrained than tree-child networks. The running time of the algorithm is quadratic in the size of the leaf sets.

Mathematics Subject Classifications: 05C85, 92D15

*Supported by the New Zealand Marsden Fund.
†Supported by the New Zealand Marsden Fund.

1 Introduction

Phylogenetic (evolutionary) networks rather than phylogenetic trees provide a more faithful representation of the ancestral history of certain collections of extant species. The reason for this is the existence of non-treelike (reticulate) evolutionary processes such as lateral gene transfer and hybridisation. Similar to the study of phylogenetic trees, the development of tools and algorithms to reconstruct phylogenetic networks from biological sequence data is an active area of research [14, 17, 22]. However, in this paper, we focus on the combinatorial properties of phylogenetic networks. A precise understanding of these properties is indispensable for the analysis and comparison of phylogenetic networks as well as for the advancement of network reconstruction algorithms.

At the species-level, evolution is not necessarily treelike. But, at the level of genes, we typically assume treelike evolution. Consequently, as phylogenetic networks are frequently viewed as an amalgamation of the ancestral history of genes, we are interested in the phylogenetic trees embedded (displayed) in a given phylogenetic network. From this viewpoint, there has been a variety of studies including the small maximum parsimony problem for phylogenetic networks [15], deciding if a phylogenetic network is (uniquely) determined by the phylogenetic trees it embeds [6, 20], counting the number of phylogenetic trees displayed by a phylogenetic network [12], and determining if a phylogenetic network embeds a phylogenetic tree more than once [4]. In this context, one of the most well-known studied computational problems is Tree-Containment. Here, the problem is deciding whether or not a given phylogenetic tree is embedded in a given phylogenetic network. In general, the problem is NP-complete [11], but it has been shown to be decidable in polynomial-time for several prominent classes of phylogenetic networks [1, 8, 10].

Recently posed in [8] for reticulation-visible networks, in this paper we study a natural variation of Tree-Containment. In particular, we consider the problem of deciding whether or not two given binary phylogenetic networks embed the same set of phylogenetic trees. Called Display-Set-Equivalence, we recently showed that, in general, this problem is Π_2^P-complete [5], that is, complete for the second level of the polynomial hierarchy. A related problem that is also Π_2^P-complete and that we investigated in the same paper asks whether or not the set of trees embedded in a phylogenetic network is a subset of the set of trees embedded in another network. Problems on the second level of the polynomial hierarchy are computationally more difficult than problems on the first level which include all NP- and co-NP-complete problems. For further details, see [18]. In contrast, the main result of this paper shows that there is a polynomial-time algorithm for Display-Set-Equivalence if one of the two given networks is normal and the other one is tree-child.

Normal [19] and tree-child networks [3] are two structurally constrained subclasses of phylogenetic networks. While formal definitions are given below, we informally mention here that a tree-child network has the property that every non-leaf vertex has a child that does not represent a reticulation event. Moreover, a normal network is tree-child with an additional property concerning the arcs directed into a vertex representing a reticulation event, which we refer to as "no shortcuts". Both subclasses have actively been studied

for the last ten years. Indeed, studying subclasses of phylogenetic networks is particularly appealing from a mathematical perspective because (a) several decision problems that are computationally hard in general can be solved in polynomial time for certain subclasses, and (b) algorithms that reconstruct phylogenetic networks from smaller building blocks, such as networks on three leaves, often only uniquely encode phylogenetic networks of restricted subclasses [4, 7, 9, 10, 20]. The rest of the introduction formally defines Display-Set-Equivalence, states the main result, and provides additional details.

A *binary phylogenetic network* $\mathcal{N}$ *on* X is a rooted acyclic directed graph with no arcs in parallel and satisfying the following properties:

(i) the (unique) root has out-degree two;

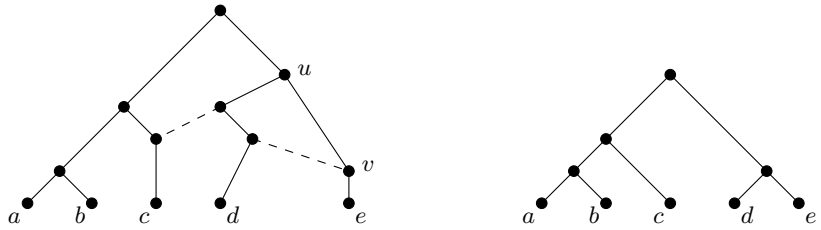
(ii) a vertex with out-degree zero has in-degree one, and the set of vertices with out-degree zero is X; and

(iii) all other vertices have either in-degree one and out-degree two, or in-degree two and out-degree one.

For technical reasons, if $|X| = 1$, we additionally allow a single vertex labelled by the element in X to be a binary phylogenetic network. The vertices in $\mathcal{N}$ of out-degree zero are called *leaves*, and so X is referred to as the *leaf set* of $\mathcal{N}$. Furthermore, vertices of in-degree one and out-degree two are *tree vertices*, while vertices of in-degree two and out-degree one are *reticulations*. The arcs directed into a reticulation are *reticulation arcs*, all other arcs are *tree arcs*. A *binary phylogenetic* X*-tree* is a binary phylogenetic network on X with no reticulations. To ease reading and since all phylogenetic networks considered in this paper are binary, we refer to a binary phylogenetic network (resp. binary phylogenetic tree) as a phylogenetic network (resp. phylogenetic tree).

Let $\mathcal{N}$ be a phylogenetic network. A reticulation arc (u, v) of $\mathcal{N}$ is a *shortcut* if there is a directed path in $\mathcal{N}$ from u to v that does not traverse (u, v). We say that $\mathcal{N}$ is a *tree-child network* if every non-leaf vertex is the parent of a tree vertex or a leaf. If, in addition, $\mathcal{N}$ has no shortcuts, then $\mathcal{N}$ is *normal*. To illustrate, in Fig. 1(i), $\mathcal{N}$ is a tree-child network but it is not normal as the arc (u, v) is a shortcut. As with all other figures in the paper, arcs are directed down the page.

Now let $\mathcal{N}$ be a phylogenetic network on X and let $\mathcal{T}$ be a phylogenetic X-tree. Then $\mathcal{N}$ *displays* $\mathcal{T}$ if $\mathcal{T}$ can be obtained from $\mathcal{N}$ by deleting arcs and vertices, and suppressing the resulting vertices of in-degree one and out-degree one. An equivalent and useful way to view the notion of displaying is as follows. The *root extension* of $\mathcal{T}$ is obtained by adjoining a new vertex, u say, to the root of $\mathcal{T}$ via a new arc directed away from u. It is easily checked that $\mathcal{N}$ displays $\mathcal{T}$ precisely if a subdivision of either $\mathcal{T}$ or the root extension of $\mathcal{T}$ can be obtained from $\mathcal{N}$ by deleting arcs and non-root vertices. We refer to such a subdivision as an *embedding*, $\mathcal{S}$ say, of $\mathcal{T}$ in $\mathcal{N}$. Observe that it follows from the definition of an embedding that the unique vertex of $\mathcal{S}$ with in-degree zero is also the root vertex of $\mathcal{N}$. Having these two vertices coincide is particularly convenient when writing arguments, and so we frequently adopt this viewpoint in the proofs of the paper.



(i) $\mathcal{N}$ (ii) $\mathcal{T}$

Figure 1: (i) A tree-child network $\mathcal{N}$ on $\{a, b, c, d, e\}$ and (ii) a phylogenetic tree $\mathcal{T}$ displayed by $\mathcal{N}$.

To illustrate the notion of display, in Fig. 1, $\mathcal{N}$ displays $\mathcal{T}$, where an embedding of $\mathcal{T}$ in $\mathcal{N}$ is shown as solid arcs. Note that there is one other distinct embedding of $\mathcal{T}$ in $\mathcal{N}$. Furthermore, the root extension $\mathcal{T}'$ of a phylogenetic tree $\mathcal{T}$ is shown in Fig. 2, where $\mathcal{T}$ is displayed by the tree-child network $\mathcal{N}'$ in the same figure since an embedding of $\mathcal{T}$ in $\mathcal{N}'$ can be obtained by deleting the two arcs (u_1, v_1) and (u_2, v_2), the two arcs (u_1, v_1) and (u'_2, v_2), or the two arcs (u'_1, v_1) and (u_2, v_2) in $\mathcal{N}'$. Now, suppose that $\mathcal{S}$ is an embedding of $\mathcal{T}$ in $\mathcal{N}$. If (u, v) is an arc in $\mathcal{N}$, we say $\mathcal{S}$ *uses* (u, v) if (u, v) is an arc in $\mathcal{S}$. The set of phylogenetic X-trees displayed by $\mathcal{N}$, called the *display set* of $\mathcal{N}$, is denoted by $T(\mathcal{N})$.

The problem of interest in this paper is the following decision problem:

Display-Set-Equivalence
Input. Two phylogenetic networks $\mathcal{N}$ and $\mathcal{N}'$ on X.
Output. Is $T(\mathcal{N}) = T(\mathcal{N}')$?
It is shown in [5] that, in general, Display-Set-Equivalence is Π_2^P-complete. In contrast, the main result of this paper shows that this decision problem is solvable in polynomial time if $\mathcal{N}$ is normal and $\mathcal{N}'$ is tree-child. In particular, we have

Theorem 1. *Let $\mathcal{N}$ and $\mathcal{N}'$ be normal and tree-child networks on X, respectively. Then deciding if $T(\mathcal{N}) = T(\mathcal{N}')$ can be done in time quadratic in the size of X.*

Before continuing, we add some remarks. The proof of Theorem 1 turned out to be much longer than we originally anticipated. If $\mathcal{N}'$ has no shortcuts, that is, $\mathcal{N}'$ is normal, then $T(\mathcal{N}) = T(\mathcal{N}')$ if and only if $\mathcal{N}$ is isomorphic to $\mathcal{N}'$ [20]. However, if $\mathcal{N}'$ is allowed to have shortcuts, then it is possible that $T(\mathcal{N}) = T(\mathcal{N}')$, but $\mathcal{N}$ is not isomorphic to $\mathcal{N}'$. For example, consider the normal and tree-child networks $\mathcal{N}$ and $\mathcal{N}'$, respectively, shown in Fig. 2. Clearly, $\mathcal{N}$ is not isomorphic to $\mathcal{N}'$, but it is easily checked that $T(\mathcal{N}) = T(\mathcal{N}')$. While we already knew of instances like that shown in Fig. 2, the allowance of shortcuts raised many more hurdles than we expected. We next explain briefly what causes at least some of these hurdles. Let v be a reticulation vertex of a tree-child network $\mathcal{N}$, and let u and u' be the two parents of v. Since $\mathcal{N}$ is tree-child, it follows from the definition (see Lemma 2) that there is a directed path from u (resp. u') to a leaf ℓ (resp. ℓ') that does not contain a reticulation arc. Importantly, if $\mathcal{N}$ is also normal, then $\ell \neq \ell'$ and the local

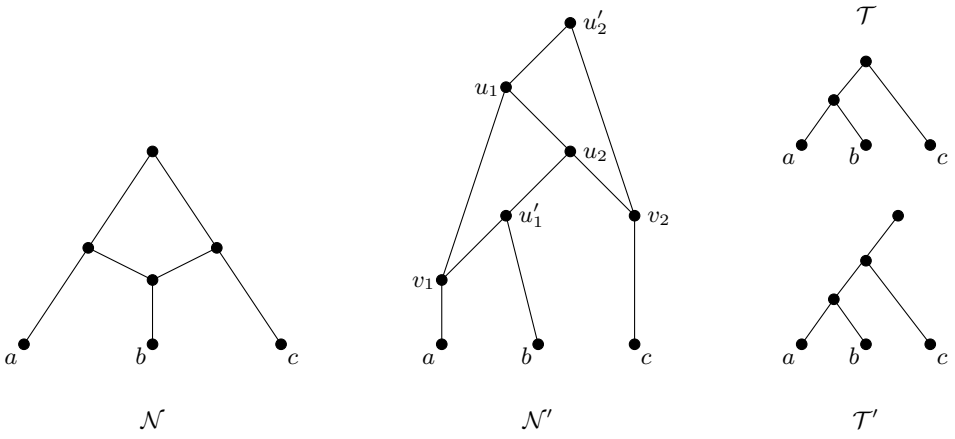


Figure 2: A normal network $\mathcal{N}$ and a tree-child network $\mathcal{N}'$, where $T(\mathcal{N}) = T(\mathcal{N}')$ but $\mathcal{N}$ is not isomorphic to $\mathcal{N}'$. Furthermore, the root extension $\mathcal{T}'$ of a phylogenetic tree $\mathcal{T}$.

structure of $\mathcal{N}$ around v is quite restricted. On the other hand, if $\mathcal{N}$ is not normal and either (u, v) or (u', v) is a shortcut, then it is possible for ℓ and ℓ' to coincide. In turn, this implies that the local structure of $\mathcal{N}$ around v is much less restrictive. To establish that Display-Set-Equivalence is solvable in polynomial time for when the input consists of a normal network $\mathcal{N}$ and a tree-child network $\mathcal{N}'$, we have used a detailed analysis of the local structures of $\mathcal{N}$ and $\mathcal{N}'$ relative to a reticulation in $\mathcal{N}$ under the assumption that $T(\mathcal{N}) = T(\mathcal{N}')$.

Now, let $\mathcal{N}$ be a phylogenetic network, and let u be a vertex of $\mathcal{N}$. We say that u is *visible* if there is a leaf, ℓ say, in $\mathcal{N}$ such that every directed path from the root of $\mathcal{N}$ to ℓ traverses u, in which case, ℓ *verifies the visibility of* u. Furthermore, $\mathcal{N}$ is *reticulation-visible* if every reticulation is visible. To summarise, note that normal networks are a proper subclass of tree-child networks and tree-child networks are a proper subclass of reticulation-visible networks. In particular, tree-child networks are precisely the class of networks in which every vertex is visible [3]. As mentioned in the third paragraph of the introduction, Display-Set-Equivalence was recently posed for when $\mathcal{N}$ and $\mathcal{N}'$ are both reticulation-visible and the computational complexity of this problem remains open. Knowing the hurdles that had to be overcome in the proof of Theorem 1, perhaps an easier problem to consider (depending on its complexity) is Display-Set-Equivalence for when $\mathcal{N}$ and $\mathcal{N}'$ are both tree-child.

The paper is organised as follows. Section 2 contains some additional concepts as well as several lemmas concerning tree-child networks. The proof of Theorem 1 is algorithmic and relies on comparing the structures of $\mathcal{N}$ and $\mathcal{N}'$ local to a common pair of leaves. Section 3 establishes the necessary structural results to make these comparisons. Depending on the outcomes of the comparisons, the algorithm recurses in one of three ways. The lemmas associated with these recursions are given in Section 4. The algorithm, its correctness, and its running time, and thus the proof of Theorem 1, are given in the last section. A more detailed overview of the algorithm underlying the proof of Theorem 1 is given at the end of the next section.

2 Preliminaries

Throughout the paper, X denotes a non-empty finite set and all paths are directed. Furthermore, if D is a set and b is an element, we write $D \cup b$ for $D \cup \{b\}$ and $D - b$ for $D - \{b\}$.

Cluster and visibility sets. Let $\mathcal{N}$ be a phylogenetic network on X with root ρ, and let u be a vertex of $\mathcal{N}$. A vertex v is *reachable* from u if there is a path from u to v. The set of leaves reachable from u, denoted C_u, is the *cluster (set) of* u. Furthermore, the set of leaves verifying the visibility of u, denoted V_u, is the *visibility set of* u. Note that the visibility set of u is a subset of the cluster set of u.

Let $\mathcal{T}$ be a phylogenetic X-tree. A non-empty subset C of X is a *cluster of* $\mathcal{T}$ if there is a vertex u in $\mathcal{T}$ such that $C = C_u$. For non-empty (disjoint) subsets Y and Z of X, we say that $\{Y, Z\}$ is a *generalised cherry* of $\mathcal{T}$ if Y, Z, and $Y \cup Z$ are all clusters of $\mathcal{T}$.

Normal and tree-child networks. Let u be a vertex of a phylogenetic network $\mathcal{N}$ on X. A path P starting at u and ending at a leaf is a *tree-path* if every non-terminal vertex is a tree vertex, in which case, *u has a tree-path* and P is a *tree-path for u*. Observe that u may or may not be a reticulation and that every arc in a tree-path is a tree arc. The next lemma is freely-used throughout the paper. Part (ii) is well-known and follows immediately from the definition of a tree-child network, and (iii) was noted in the introduction.

Lemma 2. *Let $\mathcal{N}$ be a phylogenetic network. Then the following statements are equivalent:*

(i) *$\mathcal{N}$ is tree-child,*

(ii) *every vertex of $\mathcal{N}$ has a tree-path, and*

(iii) *every vertex of $\mathcal{N}$ is visible.*

It follows from Lemma 2 that all visibility sets of a tree-child network are non-empty.

Let a and b be distinct leaves of a phylogenetic network $\mathcal{N}$, and let p_a and p_b denote the parents of a and b, respectively. Then $\{a, b\}$ is a *cherry* if $p_a = p_b$. Furthermore, $\{a, b\}$ is a *reticulated cherry* if the parent of one of the leaves, say b, is a reticulation and (p_a, p_b) is an arc in $\mathcal{N}$. Note that, if this holds, then p_a is a tree vertex. The arc (p_a, p_b) is the *reticulation arc* of the reticulated cherry $\{a, b\}$. As with the previous lemma, the next lemma [2] is freely-used throughout the paper.

Lemma 3. *Let $\mathcal{N}$ be a tree-child network on X, where $|X| \geqslant 2$. Then $\mathcal{N}$ has either a cherry or a reticulated cherry.*

The next lemma is established in [19].

Lemma 4. *Let $\mathcal{N}$ be a normal network on X, and let t and u be vertices in $\mathcal{N}$. Then $C_u \subseteq C_t$ if and only if u is reachable from t.*

Let $\mathcal{N}$ be a phylogenetic network on X. Let $\mathcal{S}$ be an embedding in $\mathcal{N}$ of a phylogenetic X-tree $\mathcal{T}$ and let C be a cluster of $\mathcal{T}$. Analogous to cluster sets of $\mathcal{N}$, each vertex w of $\mathcal{S}$ has a *cluster set* and this set consists of the elements in X at the end of a path in $\mathcal{S}$ starting at w. Of course, the cluster set of w relative to $\mathcal{S}$ is a subset of the cluster set of w relative to $\mathcal{N}$. The vertex in $\mathcal{S}$ *corresponding* to C is the (unique) vertex u whose cluster set relative to $\mathcal{S}$ is C and with the property that every other vertex with cluster set C in $\mathcal{S}$ is on a path from the root of $\mathcal{S}$ to u.

Lemma 5. *Let $\mathcal{N}$ be a normal network on X and let u be a tree vertex of $\mathcal{N}$. Let $\mathcal{T}$ be a phylogenetic X-tree having cluster C_u. If $\mathcal{S}$ is an embedding of $\mathcal{T}$ in $\mathcal{N}$, then the vertex in $\mathcal{S}$ corresponding to C_u is u.*

Proof. Suppose that $\mathcal{S}$ is an embedding of $\mathcal{T}$ in $\mathcal{N}$. Let t be the vertex in $\mathcal{S}$ corresponding to C_u, and observe that t is a tree vertex. Clearly, $C_u \subseteq C_t$ and so, by Lemma 4, u is reachable from t on a path P in $\mathcal{N}$. If $t \neq u$, then, as $\mathcal{N}$ is normal and therefore has no shortcuts, t is the parent of a vertex, v say, that is not on P. Now, there is a tree-path from v to a leaf ℓ. By construction, $\ell \notin C_u$. In turn, regardless of whether or not v is a reticulation, this implies that the cluster in $\mathcal{S}$ corresponding to t contains ℓ, a contradiction. Thus $t = u$, thereby completing the proof of the lemma. □

Deleting arcs and leaves. Let $\mathcal{N}$ be a phylogenetic network on X, and let (u, v) be an arc of $\mathcal{N}$. We denote the directed graph obtained from $\mathcal{N}$ by deleting (u, v) and suppressing any resulting vertices with in-degree one and out-degree one by $\mathcal{N}\backslash(u, v)$. Note that $\mathcal{N}\backslash(u, v)$ may have arcs in parallel. If u is the root of $\mathcal{N}$, we additionally delete u (and its incident arc) after deleting (u, v). Extending this notation in the obvious way, we use $\mathcal{N}\backslash\{(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)\}$ to denote the directed graph obtained from $\mathcal{N}$ by deleting the arcs $(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)$ and suppressing any resulting vertices with in-degree one and out-degree one. Moreover, if b is a leaf of $\mathcal{N}$, then the directed graph obtained from $\mathcal{N}$ by deleting b (and its incident arc), and suppressing any resulting vertex of in-degree one and out-degree one is denoted by $\mathcal{N}\backslash b$. Again, if the parent of b is the root of $\mathcal{N}$, we additionally delete the root (and its incident arc) after deleting b.

Deleting an arc or a leaf of a phylogenetic network does not necessarily result in another phylogenetic network. The next two lemmas, which are also freely used in the paper, give some sufficient conditions for when these operations result in a phylogenetic network. The proof of the first lemma is straightforward and omitted.

Lemma 6. *Let $\mathcal{N}$ be a tree-child network on X, and suppose that $\{a, b\}$ is a cherry of $\mathcal{N}$. Then $\mathcal{N}\backslash b$ is a tree-child network on $X - b$. Moreover, if $\mathcal{N}$ is normal, then $\mathcal{N}\backslash b$ is normal.*

The next lemma generalises a result in [2]. A shortcut (u, v) in a phylogenetic network $\mathcal{N}$ is *trivial* if the parent of v that is not u is a child of u.

Lemma 7. *Let $\mathcal{N}$ be a tree-child network on X, and suppose that (u, v) is a reticulation arc of $\mathcal{N}$.*

(i) *Then* $\mathcal{N}\backslash(u,v)$ *is a tree-child network on* X*. Moreover, if* $\mathcal{N}$ *is normal, then* $\mathcal{N}\backslash(u,v)$ *is normal.*

(ii) *If* (u,v) *is trivial, then* $T(\mathcal{N}) = T(\mathcal{N}\backslash(u,v))$.

Proof. We first prove (i). Since $\mathcal{N}$ is tree-child, u is a tree vertex, the child of u that is not v is either a tree vertex or a leaf, and the unique child of v is either a tree vertex or a leaf. Therefore, as $\mathcal{N}$ has no parallel arcs, $\mathcal{N}\backslash(u,v)$ has no parallel arcs, and so $\mathcal{N}\backslash(u,v)$ is a phylogenetic network. Moreover, it also follows that no new shortcut is created in deleting (u,v) from $\mathcal{N}$. Furthermore, if w is an arbitrary vertex of $\mathcal{N}$, then no tree-path for w in $\mathcal{N}$ traverses (u,v) and so, every vertex in $\mathcal{N}\backslash(u,v)$ has a tree-path. Part (i) now follows.

For (ii), regardless of whether (u,v) is trivial, $T(\mathcal{N}\backslash(u,v)) \subseteq T(\mathcal{N})$. So assume that (u,v) is trivial, in which case (u,u') is an arc in $\mathcal{N}$, and let $\mathcal{T}$ be a phylogenetic tree displayed by $\mathcal{N}$. Let $\mathcal{S}$ be an embedding of $\mathcal{T}$ in $\mathcal{N}$. If $\mathcal{S}$ does not use (u,v), then it is clear that $\mathcal{N}\backslash(u,v)$ displays $\mathcal{T}$. On the other hand, if $\mathcal{S}$ uses (u,v), then by replacing (u,v) with (u',v) we obtain an embedding of $\mathcal{T}$ in $\mathcal{N}$ that does not use (u,v), and so $\mathcal{N}\backslash(u,v)$ displays $\mathcal{T}$. Note that, as $\mathcal{N}$ is tree-child, $\mathcal{S}$ uses (u,u'). Hence $T(\mathcal{N}) \subseteq T(\mathcal{N}\backslash(u,v))$. This completes the proof of (ii). □

We end this section by briefly outlining the algorithm associated with the proof of Theorem 1. Called SameDisplaySet, the algorithm takes as its input normal and tree-child networks $\mathcal{N}$ and $\mathcal{N}'$, respectively, and proceeds by first finding a cherry or a reticulated cherry, $\{a,b\}$ say, in $\mathcal{N}$. It then considers the structure of $\mathcal{N}'$ (and if necessary $\mathcal{N}$) local to leaves a and b, and decides whether to return $T(\mathcal{N}) \neq T(\mathcal{N}')$ or to continue. This decision is based on three Propositions 8, 10, and 11. These propositions give necessary structural properties if $T(\mathcal{N}) = T(\mathcal{N}')$. If the algorithm continues, it deletes certain arcs and leaves in $\mathcal{N}$ and $\mathcal{N}'$. Lemmas 12–14 show that the resulting normal and tree-child networks after the deletions, $\mathcal{N}_1$ and $\mathcal{N}_1'$ say, display the same set of phylogenetic trees, that is $T(\mathcal{N}_1) = T(\mathcal{N}_1')$, if and only if $T(\mathcal{N}) = T(\mathcal{N}')$. The algorithm now recurses on $\mathcal{N}_1$ and $\mathcal{N}_1'$ by finding a cherry or a reticulated cherry of $\mathcal{N}_1$. Eventually, SameDisplaySet either stops and returns $T(\mathcal{N}) \neq T(\mathcal{N}')$ or it reduces $\mathcal{N}$ and $\mathcal{N}'$ to a phylogenetic network consisting of two leaves, in which case $T(\mathcal{N}) = T(\mathcal{N}')$. The formal description of SameDisplaySet is given at the start of Section 5. The reader may choose to refer to that while reading through Sections 3 and 4.

3 Structural Properties

The purpose of this section is to establish three structural results, namely, Propositions 8, 10, and 11. Let $\mathcal{N}$ and $\mathcal{N}'$ be a normal and a tree-child network on X, respectively. Relative to either a cherry or a reticulated cherry, $\{a,b\}$ say, of $\mathcal{N}$, these results determine the structure of $\mathcal{N}'$ local to a and b if $T(\mathcal{N}) = T(\mathcal{N}')$. The first proposition considers when $\{a,b\}$ is a cherry of $\mathcal{N}$, while the second and third propositions consider when $\{a,b\}$

is a reticulated cherry of $\mathcal{N}$ in which the parent of b is a reticulation and the parent of b in $\mathcal{N}'$ is either a reticulation or a tree vertex, respectively. Each of the proofs first considers the parent vertex of b in $\mathcal{N}'$ and establishes sufficient structure of $\mathcal{N}'$ close to b under the assumption that $T(\mathcal{N}) = T(\mathcal{N}')$, so that the iterative algorithm in Section 5 works correctly.

Throughout the proofs in this section, we repeatedly use the following which immediately follows from results in [16]. If $\mathcal{N}$ is a tree-child network on X, then every embedding of a phylogenetic X-tree displayed by $\mathcal{N}$ uses all of the tree arcs and, for each reticulation v, exactly one of the reticulation arcs directed into v. In particular, this implies that if $\mathcal{S}$ is an embedding of a phylogenetic X-tree in $\mathcal{N}$ and P is a tree-path in $\mathcal{N}$, then $\mathcal{S}$ uses every arc in P. Moreover, if $\mathcal{S}'$ is a subset of arcs of $\mathcal{N}$ consisting of all tree arcs and precisely one reticulation arc directed into each reticulation, then $\mathcal{S}'$ is an embedding of a phylogenetic X-tree that is displayed by $\mathcal{N}$. Hence, in the proofs, when we consider an embedding of a phylogenetic X-tree, the focus is on stating which of the two reticulation arcs directed into a reticulation is used.

Proposition 8. *Let $\mathcal{N}$ and $\mathcal{N}'$ be normal and tree-child networks on X, respectively, and suppose $\mathcal{N}'$ has no trivial shortcuts. Let $\{a, b\}$ be a cherry of $\mathcal{N}$. Then $T(\mathcal{N}) = T(\mathcal{N}')$ only if $\{a, b\}$ is a cherry of $\mathcal{N}'$.*

Proof. Suppose $T(\mathcal{N}) = T(\mathcal{N}')$. Note that $\{a, b\}$ is a cherry of every phylogenetic X-tree displayed by $\mathcal{N}$. Let p'_a and p'_b denote the parents of a and b in $\mathcal{N}'$, respectively. First assume that p'_b is a tree vertex. Then, as $T(\mathcal{N}) = T(\mathcal{N}')$, it follows that $C_{p'_b} = \{a, b\}$; otherwise, there is a phylogenetic X-tree displayed by $\mathcal{N}'$ that is not displayed by $\mathcal{N}$. Thus the child vertex of p'_b in $\mathcal{N}'$ that is not b is either a or p'_a. In particular, $\{a, b\}$ is either a cherry or a reticulated cherry with reticulation leaf a in $\mathcal{N}'$. Consider the latter. If q' denotes the parent of p'_a that is not p'_b in $\mathcal{N}'$, then (q', p'_a) is a shortcut. Otherwise, there is a tree-path from q' to a leaf that is not b, and so, using (q', p'_a) and not (p'_b, p'_a) in an embedding of a phylogenetic X-tree, it follows that $\mathcal{N}'$ displays a phylogenetic X-tree in which $\{a, b\}$ is not a cherry. Now let t' denote the child vertex of q' that is not p'_a. Since $\mathcal{N}'$ is tree-child and (q', p'_a) is a shortcut, t' is a tree vertex and $\{a, b\} \subseteq C_{t'}$. If $C_{t'} - a \neq \{b\}$, then, using (q', p'_a) and not (p'_b, p'_a), it follows that $\mathcal{N}'$ displays a phylogenetic X-tree in which $C_{t'} - a$ is a cluster of size at least two containing b, and thus it is not displayed by $\mathcal{N}$. So $C_{t'} - a = \{b\}$ and, in particular, $t' = p'_b$. Thus (q', p'_a) is a trivial shortcut, a contradiction. Therefore if p'_b is a tree vertex, then $\{a, b\}$ is a cherry of $\mathcal{N}'$. If p'_b is a reticulation in $\mathcal{N}'$, then a similar argument leads to the conclusion that $\mathcal{N}'$ has a trivial shortcut. This completes the proof of the proposition. □

We next consider the relative structure local to leaves a and b in $\mathcal{N}$, where $\{a, b\}$ is a reticulated cherry of $\mathcal{N}$. For the next three results, we suppose that $\{a, b\}$ is a reticulated cherry of $\mathcal{N}$ with reticulation leaf b as shown in Fig. 3. Note that, although not shown, if $C_q - (V_q \cup b)$ is nonempty, then $\mathcal{N}$ contains paths from the root ρ to leaves in $C_q - (V_q \cup b)$ avoiding q. Furthermore, in viewing Fig. 3 as well as the other figures in the remainder of the paper, the structure of the phylogenetic network within a box is unknown. However,

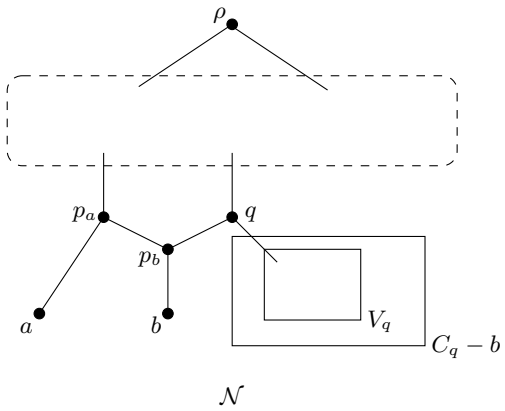


Figure 3: The structure of $\mathcal{N}$ local to the reticulated cherry $\{a, b\}$. Note that, for each leaf $\ell \in C_q - (V_q \cup b)$, there is a path from ρ to ℓ avoiding q.

the label of the box indicates the location of the visibility or cluster set of some particular vertex.

The proof of the next lemma is straightforward and omitted.

Lemma 9. *Let $\mathcal{N}$ be a normal network on X, and suppose that $\{a, b\}$ is a reticulated cherry of $\mathcal{N}$ as shown in Fig. 3. If $\mathcal{T}$ is a phylogenetic X-tree displayed by $\mathcal{N}$, then either*

(i) *$\{a, b\}$ is a cherry of $\mathcal{T}$, or*

(ii) *$\{b, C'_q\}$ is a generalised cherry of $\mathcal{T}$, where $V_q \subseteq C'_q \subseteq C_q - b$ and $a \notin C_q$.*

Moreover, for each $\{A, B\} \in \big\{\{a, b\}, \{b, V_q\}, \{b, C_q - b\}\big\}$, there is a phylogenetic X-tree displayed by $\mathcal{N}$ in which $\{A, B\}$ is a generalised cherry.

Proposition 10. *Let $\mathcal{N}$ and $\mathcal{N}'$ be normal and tree-child networks on X, respectively, and suppose that $\{a, b\}$ is a reticulated cherry of $\mathcal{N}$ as shown in Fig. 3. If the parent of b in $\mathcal{N}'$ is a reticulation, then $T(\mathcal{N}) = T(\mathcal{N}')$ only if, up to isomorphism, $\{a, b\}$ is a reticulated cherry of $\mathcal{N}'$ as shown in Fig. 4, where $V_{q'_2} = V_q$ and $C_{q'_2} = C_q$.*

Proof. Let $\{a, b\}$ be a reticulated cherry of $\mathcal{N}$ as shown in Fig. 3. Thus p_a and p_b denote the parents of a and b in $\mathcal{N}$, respectively, where p_b is a reticulation, and q denotes the parent of p_b in $\mathcal{N}$ that is not p_a. Since $\mathcal{N}$ is normal, (q, p_b) is not a shortcut and $a \notin C_q$. Suppose $T(\mathcal{N}) = T(\mathcal{N}')$, and consider $\mathcal{N}'$. Let p'_a and p'_b denote the parents of a and b in $\mathcal{N}'$, respectively, where p'_b is a reticulation. Let q'_1 and q'_2 denote the parents of p'_b in $\mathcal{N}'$. We will eventually show that one of q'_1 and q'_2, say q'_1, is p'_a.

10.1. *Neither (q'_1, p'_b) nor (q'_2, p'_b) is a shortcut.*

Proof. Assume at least one of (q'_1, p'_b) and (q'_2, p'_b) is a shortcut. Without loss of generality, we may assume (q'_2, p'_b) is a shortcut, and so (q'_1, p'_b) is not a shortcut. Observe that

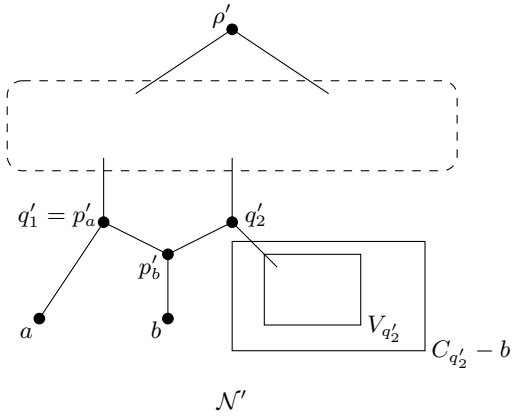


Figure 4: The structure of $\mathcal{N}'$ local to the leaves a and b as established in Proposition 10 when $\mathcal{N}$ is as shown in Fig. 3, the parent of b in $\mathcal{N}'$ is a reticulation, and $T(\mathcal{N}) = T(\mathcal{N}')$. It is also shown that $V_{q'_2} = V_q$ and $C_{q'_2} = C_q$. Note that, for each leaf $\ell \in C_{q'_2} - (V_{q'_2} \cup b)$, there is a path from ρ' to ℓ avoiding q'_2.

$C_{q'_1} \subseteq C_{q'_2}$. Since $T(\mathcal{N}) = T(\mathcal{N}')$, it follows by Lemma 9 that $\mathcal{N}'$ displays a phylogenetic X-tree with $\{a, b\}$ as a cherry and a phylogenetic X-tree with $\{b, V_q\}$ as a generalised cherry. As q'_1 and q'_2 each have a tree-path and every embedding of a phylogenetic X-tree in $\mathcal{N}'$ uses either (q'_1, p'_b) or (q'_2, p'_b), it follows that $V_q \cup a \subseteq C_{q'_2}$. But then, there is an embedding of a phylogenetic X-tree in $\mathcal{N}'$ using (q'_2, p'_b) and not (q'_1, p'_b) which has a generalised cherry $\{b, C_{q'_2} - b\}$. But, $C_{q'_2} - b$ contains $V_q \cup a$ and, by the first part of Lemma 9, $\mathcal{N}$ displays no such tree. Hence neither (q'_1, p'_b) nor (q'_2, p'_b) is a shortcut. □

By (10.1), neither (q'_1, p'_b) nor (q'_2, p'_b) is a shortcut. Therefore, for some $i \in \{1, 2\}$, we have $C_{q'_i} - b = \{a\}$ as $T(\mathcal{N}) = T(\mathcal{N}')$. If not, then one of the following two cases applies.

(i) If $a \notin C_{q'_1} - b$ and $a \notin C_{q'_2} - b$, then, as each of q'_1 and q'_2 has a tree-path, there is no phylogenetic X-tree displayed by $\mathcal{N}'$ with $\{a, b\}$ as a cherry.

(ii) If, for some $i \in \{1, 2\}$, we have $a \in C_{q'_i} - b$ and $|C_{q'_i} - b| \geqslant 2$, then there is a phylogenetic X-tree displayed by $\mathcal{N}'$ in which $\{b, C_{q'_i} - b\}$ is a generalised cherry.

Both cases contradict Lemma 9. Hence, without loss of generality, we may assume that $C_{q'_1} - b = \{a\}$ and so, as $\mathcal{N}'$ is tree-child, $q'_1 = p'_a$. That is, $\{a, b\}$ is a reticulated cherry of $\mathcal{N}'$. Observe that, as (q'_2, p'_b) is not a shortcut by (10.1), we have $a \notin C_{q'_2} - b$.

By Lemma 9, $\mathcal{N}$ displays a phylogenetic X-tree with generalised cherry $\{b, V_q\}$ and so, as $T(\mathcal{N}) = T(\mathcal{N}')$, it follows that $V_q \subseteq C_{q'_2} - b$ and $V_{q'_2} \subseteq V_q$. In turn, as $\mathcal{N}'$ displays a phylogenetic X-tree with generalised cherry $\{b, V_{q'_2}\}$ and $T(\mathcal{N}) = T(\mathcal{N}')$, we have $V_{q'_2} \subseteq C_q - b$ and $V_q \subseteq V_{q'_2}$. Thus $V_q = V_{q'_2}$. Similarly, as $\mathcal{N}$ displays a phylogenetic X-tree with generalised cherry $\{b, C_q - b\}$, and $\mathcal{N}'$ displays a phylogenetic X-tree with generalised cherry $\{b, C_{q'_2} - b\}$, we deduce that $C_q - b \subseteq C_{q'_2} - b$ and $C_{q'_2} - b \subseteq C_q - b$, so $C_q - b = C_{q'_2} - b$. Thus $\{a, b\}$ is a reticulated cherry of $\mathcal{N}'$ as shown in Fig. 4 with $V_{q'_2} = V_q$ and $C_{q'_2} = C_q$, and this completes the proof of the proposition. □

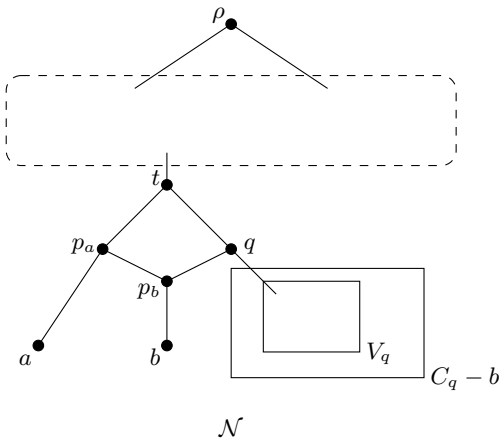


Figure 5: Additional structure of $\mathcal{N}$ local to the leaves a and b as shown in Proposition 11 when $\{a, b\}$ is a reticulated cherry of $\mathcal{N}$ as shown in Fig. 3, the parent of b in $\mathcal{N}'$ is a tree vertex, and $T(\mathcal{N}) = T(\mathcal{N}')$. Note that, for each leaf $\ell \in C_q - (V_q - b)$, there is a path from ρ to ℓ avoiding q.

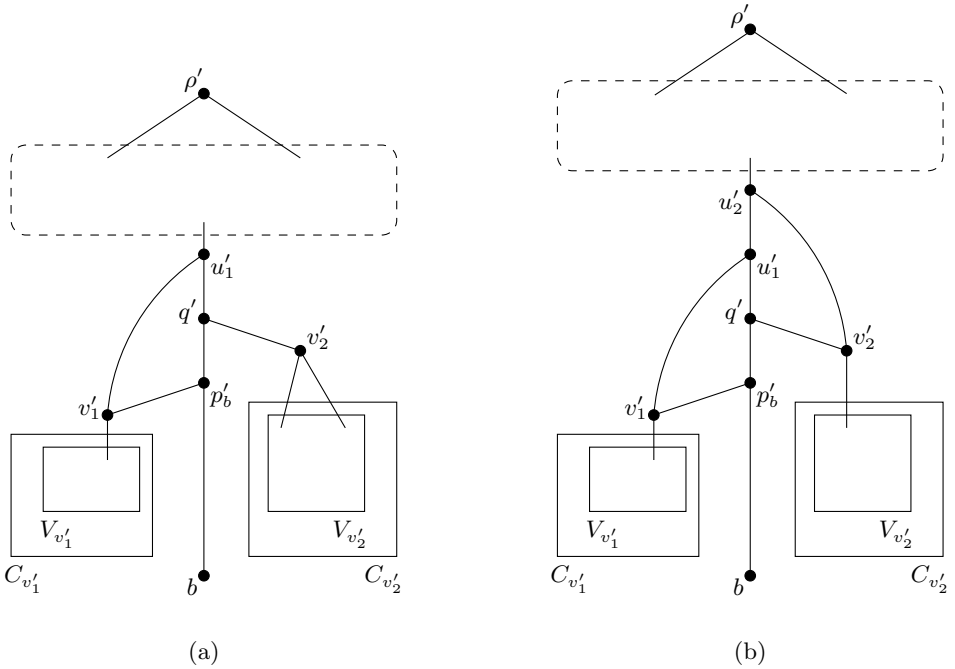


Figure 6: The two possible structures of $\mathcal{N}'$ local to the leaves a and b as shown in Proposition 11 when $\mathcal{N}$ is as shown in Fig. 3, the parent of b in $\mathcal{N}'$ is a tree vertex, and $T(\mathcal{N}) = T(\mathcal{N}')$. It is also shown that $\{V_{v'_1}, V_{v'_2}\} = \{\{a\}, V_q\}$ and $\{C_{v'_1}, C_{v'_2}\} = \{\{a\}, C_q - b\}$. Note that, if $C_{v'_i} \neq \{a\}$ for some $i \in \{1, 2\}$, then, for each leaf $\ell \in C_{v'_i} - V_{v'_i}$, there is a path from ρ' to ℓ avoiding v'_i. Furthermore, in (a), v'_2 could be a leaf.

Proposition 11. *Let $\mathcal{N}$ and $\mathcal{N}'$ be normal and tree-child networks on X, respectively, and suppose that $\mathcal{N}'$ has no trivial shortcuts and $\{a, b\}$ is a reticulated cherry of $\mathcal{N}$ as shown in Fig. 3. If the parent of b in $\mathcal{N}'$ is a tree vertex, then $T(\mathcal{N}) = T(\mathcal{N}')$ only if, up to isomorphism, in $\mathcal{N}$, leaves a and b are as shown in Fig. 5 and, in $\mathcal{N}'$, leaves a and b are as shown in either Fig. 6(a) or Fig. 6(b), where $\{V_{v'_1}, V_{v'_2}\} = \{\{a\}, V_q\}$ and $\{C_{v'_1}, C_{v'_2}\} = \{\{a\}, C_q - b\}$.*

Proof. Let $\{a, b\}$ be a reticulated cherry of $\mathcal{N}$ as shown in Fig. 3, and suppose that $T(\mathcal{N}) = T(\mathcal{N}')$. Let p'_b denote the parent of b in $\mathcal{N}'$, and suppose that p'_b is a tree vertex. Let v'_1 denote the child of p'_b in $\mathcal{N}'$ that is not b. If v'_1 is a tree vertex or a leaf, then either there is no phylogenetic X-tree displayed by $\mathcal{N}'$ in which $\{a, b\}$ is a cherry or there is no phylogenetic X-tree displayed by $\mathcal{N}'$ in which $\{b, V_q\}$ is a generalised cherry. This contradiction to Lemma 9 implies that we may assume v'_1 is a reticulation.

11.1. *Either $C_{v'_1} = \{a\}$ or $V_{v'_1} = V_q$.*

Proof. Using the arc (p'_b, v'_1), there are embeddings of phylogenetic X-trees in $\mathcal{N}'$ in which $\{b, C_{v'_1}\}$ and $\{b, V_{v'_1}\}$ are generalised cherries. Thus, as $T(\mathcal{N}) = T(\mathcal{N}')$, Lemma 9 implies that if $a \in C_{v'_1}$, then $C_{v'_1} = \{a\}$. Furthermore, by the same lemma, if $a \notin C_{v'_1}$, then $V_q \subseteq V_{v'_1}$. But, using (q, p_b) and not (p_a, p_b), there is an embedding of a phylogenetic X-tree in $\mathcal{N}$ in which $\{b, V_q\}$ is a generalised cherry and so, as $T(\mathcal{N}) = T(\mathcal{N}')$, we also have $V_{v'_1} \subseteq V_q$. Hence if $a \notin C_{v'_1}$, then $V_{v'_1} = V_q$. □

Since v'_1 is a reticulation, p'_b is not the root of $\mathcal{N}'$. Let q' denote the parent of p'_b in $\mathcal{N}'$.

11.2. *The vertex q' is either the root of $\mathcal{N}'$ or a tree vertex.*

Proof. Suppose that q' is a reticulation, and let u'_1 and u'_2 denote the parents of q'. First assume that neither (u'_1, q') nor (u'_2, q') is a shortcut. Let ℓ_1 and ℓ_2 be leaves at the end of tree-paths for u'_1 and u'_2, respectively. Note that $\ell_1 \neq \ell_2$ and $\ell_1, \ell_2 \notin C_{v'_1}$.For each $i \in \{1, 2\}$, let $\mathcal{T}_i$ be a phylogenetic X-tree displayed by $\mathcal{N}'$ for which an embedding uses (u'_i, q') and not (p'_b, v'_1). If $V_q = V_{v'_1}$, then by the first part of Lemma 9, either $\mathcal{T}_1$ or $\mathcal{T}_2$ is not displayed by $\mathcal{N}$. Hence, by (11.1), we may assume that $C_{v'_1} = \{a\}$. Since $\mathcal{T}_1$ is displayed by $\mathcal{N}$, we now deduce by the first part of Lemma 9 again that $\ell_1 \in V_q$. But then $\mathcal{T}_2$ has a generalised cherry $\{b, C'_{u'_2}\}$, where $V_q \not\subseteq C'_{u'_2} \subseteq C_{u'_2}$ as $\ell_1 \notin C_{u'_2}$, contradicting Lemma 9. Hence, without loss of generality, we may assume that (u'_2, q') is a shortcut.

Using the arc (u'_1, q') but not (p'_b, v'_1), there are embeddings of phylogenetic X-trees in $\mathcal{N}'$ in which $\{b, C_{u'_1} - b\}$ and $\{b, V_{u'_1}\}$ are generalised cherries. Therefore, as $T(\mathcal{N}) = T(\mathcal{N}')$, it follows by Lemma 9 that if $a \in C_{u'_1}$, then $C_{u'_1} - b = \{a\}$. Moreover, if $a \notin C_{u'_1}$, then, again by Lemma 9, $V_q \subseteq V_{u'_1}$. But $\mathcal{N}$ displays a phylogenetic X-tree in which $\{b, V_q\}$ is a generalised cherry and so, as $T(\mathcal{N}) = T(\mathcal{N}')$, we have $V_{u'_1} \subseteq V_q$. Hence if $a \notin C_{u'_1}$, then $V_{u'_1} = V_q$.

If (u'_2, u'_1) is an arc of $\mathcal{N}'$, then (u'_2, q') is a trivial shortcut. Therefore we may assume that (u'_2, u'_1) is not an arc. Let P' be a path in $\mathcal{N}'$ from u'_2 to u'_1. Since (u'_2, u'_1) is not an arc, P' contains at least one vertex, w' say, in addition to u'_2 and u'_1. Choose w' to be

the first such vertex on P' that has a child that does not lie on P'. As $\mathcal{N}'$ is tree-child, and so each vertex has a tree-path, it is easily checked that w' exists and $w' \neq u'_1$. Let x' denote the child of w' that does not lie on P'. Since x' has a tree-path, there is a leaf in $C_{u'_2}$ that is not in $C_{u'_1}$, that is, $C_{u'_1}$ is a proper subset of $C_{u'_2}$.

Using (u'_2, q') and not (p'_b, v'_1), it is easily seen that there is an embedding of a phylogenetic X-tree in $\mathcal{N}'$ in which $\{b, C_{u'_2} - b\}$ is a generalised cherry. If $C_{u'_1} = \{a\}$, then $a \in C_{u'_2}$ but $|C_{u'_2} - b| \geqslant 2$. Since $T(\mathcal{N}) = T(\mathcal{N}')$, this contradicts the first part of Lemma 9. Thus, $a \notin C_{u'_1}$, and so, by (11.1), $C_{v'_1} = \{a\}$ and $V_{u'_1} = V_q$, in which case, by the first part of Lemma 9, $C_{u'_2} - b \subseteq C_q - b$. On the other hand, $\mathcal{N}$ displays a phylogenetic X-tree $\mathcal{T}$ in which $\{b, C_q - b\}$ is a generalised cherry. Since $V_{u'_1} = V_q$, it follows that, for $\mathcal{N}'$ to display $\mathcal{T}$, we must have $C_q - b \subseteq C_{u'_2} - b$. Thus $C_q - b = C_{u'_2} - b$.

Now using (u'_1, q'), (w', x'), and the arcs on P', but not using (p'_b, v'_1), there is an embedding of a phylogenetic X-tree $\mathcal{T}'$ in $\mathcal{N}'$ that has two distinct clusters $b \cup C_{u'_1}$ and $C_{u'_2}$. But, by Lemma 5, if $\mathcal{S}'$ is an embedding of $\mathcal{T}'$ in $\mathcal{N}$, then the vertex of $\mathcal{S}'$ corresponding to $C_{u'_2}$ is q as $C_{u'_2} = C_q$, but then there is no distinct vertex in $\mathcal{N}$ that corresponds to $b \cup C_{u'_1}$. In particular, $\mathcal{N}$ does not display $\mathcal{T}'$. This completes the proof of (11.2). □

By (11.2), q' is either the root of $\mathcal{N}'$ or a tree vertex. Let v'_2 be the child of q' that is not p'_b. Note that $v'_1 \neq v'_2$; otherwise, (q', v'_2) is a trivial shortcut. Using the arc (q', v'_2) and not (p'_b, v'_1), there are embeddings of phylogenetic X-tree in $\mathcal{N}'$ in which $\{b, C_{v'_2}\}$ and $\{b, V_{v'_2}\}$ are generalised cherries. Therefore, by the first part of Lemma 9, if $a \in C_{v'_2}$, then $C_{v'_2} = \{a\}$. Furthermore, if $a \notin C_{v'_2}$, then, by the same Lemma 9, $V_q \subseteq V_{v'_2}$. But $\mathcal{N}$ displays a phylogenetic X-tree in which $\{b, V_q\}$ is a generalised cherry and so, by Lemma 9 again, we have $V_{v'_2} \subseteq V_q$. Thus if $a \notin C_{v'_2}$, then $V_{v'_2} = V_q$. In combination with (11.1), we now have

11.3. $\{V_{v'_1}, V_{v'_2}\} = \{\{a\}, V_q\}$. *Also, if* $V_{v'_i} = \{a\}$, *then* $C_{v'_i} = \{a\}$ *for each* $i \in \{1, 2\}$.

Using arcs (p'_b, v'_1) and (q', v'_2), there is an embedding of a phylogenetic X-tree $\mathcal{T}'$ in $\mathcal{N}'$ with generalised cherries $\{b, V_{v'_1}\}$ and $\{V_{v'_1} \cup b, V_{v'_2}\}$. Since $T(\mathcal{N}) = T(\mathcal{N}')$, it follows that $\mathcal{N}$ displays $\mathcal{T}'$ as well. But then, by considering an embedding of $\mathcal{T}'$ in $\mathcal{N}$ together with (11.3), it is easily seen that $\mathcal{N}$, and therefore $\mathcal{N}'$, displays a phylogenetic X-tree $\mathcal{T}$ with generalised cherries $\{b, V_{v'_2}\}$ and $\{V_{v'_2} \cup b, V_{v'_1}\}$. To see this, observe that an embedding of $\mathcal{T}$ in $\mathcal{N}$ can be obtained from an embedding of $\mathcal{T}'$ in $\mathcal{N}$ by either deleting (p_a, p_b) and adding (q, p_b), or deleting (q, p_b) and adding (p_a, p_b). It follows that q' is not the root of $\mathcal{N}'$. Let u'_1 be the parent of v'_1 that is not p'_b.

11.4. *The arc* (u'_1, v'_1) *is a shortcut in* $\mathcal{N}'$. *In particular,* (u'_1, q') *is an arc in* $\mathcal{N}'$.

Proof. Consider an embedding $\mathcal{S}'$ of $\mathcal{T}$ in $\mathcal{N}'$, where $\mathcal{T}$ is the phylogenetic X-tree with generalised cherries $\{b, V_{v'_2}\}$ and $\{V_{v'_2} \cup b, V_{v'_1}\}$. Clearly, $\mathcal{S}'$ uses (u'_1, v'_1) and not (p'_b, v'_1). If (u'_1, v'_1) is not a shortcut, then $\mathcal{N}'$ has a tree-path from u'_1 to a leaf that is not in $V_q \cup a$. But then $\mathcal{S}'$ is not an embedding of $\mathcal{T}$ in $\mathcal{N}'$. Thus (u'_1, v'_1) is a shortcut in $\mathcal{N}'$.

Now, in $\mathcal{N}'$, there is a tree-path from u'_1 to a leaf ℓ. Since $\mathcal{S}'$ is an embedding of $\mathcal{T}$ in $\mathcal{N}'$, it is easily checked that either $\ell = b$, or v'_2 is a tree vertex and ℓ is at the end of a tree-path for v'_2. Both possibilities imply that there is a tree-path P' in $\mathcal{N}'$ from u'_1 to

b. Let t' denote the parent of q' and observe that t' is on P'. We next show that $t' = u'_1$. Towards a contradiction, assume that $t' \neq u'_1$. Let w' be the child of t' that is not q'. If $w' = v'_2$, then $\mathcal{N}'$ has a trivial shortcut, so $w' \neq v'_2$. It follows by (11.3) that there is a tree-path from w' to a leaf ℓ' such that $\ell' \notin V_q \cup a$. Using (u'_1, v'_1), (q', v'_2), (t', w'), and the arcs on P', there is an embedding of a phylogenetic X-tree $\mathcal{T}'_1$ in $\mathcal{N}'$ with generalised cherries $\{b, V_{v'_2}\}$ and $\{V_{v'_2} \cup b, V_{w'}\}$. Note that $\ell' \in V_{w'}$. By considering an embedding of $\mathcal{T}'_1$ in $\mathcal{N}$, it is easily seen that $\mathcal{N}$, and therefore $\mathcal{N}'$ displays a phylogenetic X-tree $\mathcal{T}_1$ with generalised cherries $\{b, V_{v'_1}\}$ and $\{V_{v'_2}, V_{w'}\}$. If v'_2 is a tree vertex in $\mathcal{N}'$, then $\mathcal{N}'$ does not display $\mathcal{T}_1$. Therefore we may assume that v'_2 is a reticulation in $\mathcal{N}'$.

If w' is not reachable from v'_2, then $\ell' \notin C_q \cup a$, in which case, using (u'_1, v'_1), (t', w'), the arcs on P', but not (q', v'_2), we deduce that there is an embedding of a phylogenetic X-tree in $\mathcal{N}'$ with a generalised cherry $\{b, C_{w'}\}$, where $\ell' \in C_{w'}$. This contradiction to the first part of Lemma 9 implies w' is reachable from v'_2. But then using (u'_1, v'_1), (t', w'), the arcs on P', but not (q', v'_2), it follows that there is an embedding of a phylogenetic X-tree in $\mathcal{N}'$ such that neither $\{a, b\}$ nor $\{b, C'_q\}$, where $V_q \subseteq C'_q$, is a generalised cherry. But then, as $T(\mathcal{N}) = T(\mathcal{N}')$, we again obtain a contradiction to the first part of Lemma 9. Hence $t' = u'_1$, that is (u'_1, q') is an arc in $\mathcal{N}'$. □

We next establish the additional structure of $\mathcal{N}$ as shown in Fig. 5. Let $\mathcal{S}$ be an embedding of $\mathcal{T}$ in $\mathcal{N}$, where $\mathcal{T}$ is still the phylogenetic X-tree with generalised cherries $\{b, V_{v'_2}\}$ and $\{V_{v'_2} \cup b, V_{v'_1}\}$. Let t denote the tree vertex in $\mathcal{S}$ corresponding to the cluster $V_q \cup \{a, b\}$, and let P_a and P_q denote the paths in $\mathcal{S}$ from t to p_a and t to q, respectively.

11.5. *In $\mathcal{N}$, the paths P_a and P_q consist of the arcs (t, p_a) and (t, q), respectively.*

Proof. We begin by observing that, apart from p_a and q, there is no vertex on either P_a or P_q which is the start of a tree-path to a leaf avoiding p_a and q. Otherwise, $\mathcal{S}$ is not an embedding of $\mathcal{T}$ in $\mathcal{N}$. First consider P_a, and suppose that (t, u) is an arc on P_a, where $u \neq p_a$. Assume u is a tree vertex. Then u has a child vertex, w say, that is not on either P_a or P_q. To see this, if u has both of its child vertices on P_a, then one of its children is a reticulation, and so there is a tree-path from u to a leaf avoiding p_a and q, a contradiction. Furthermore, if u has a child vertex on P_q, then either $\mathcal{N}$ has a trivial shortcut or there is a tree-path from a vertex on P_q to a leaf avoiding p_a and q, another contradiction. Now, there is a tree-path from w to a leaf ℓ_w such that $\ell_w \notin \{a, b\} \cup V_q$. By (11.3), either $C_{v'_1} = \{a\}$ or $C_{v'_2} = \{a\}$. If $C_{v'_1} = \{a\}$, then, by using (q, p_b), the arcs on P_a and P_q, and (u, w), it is easily checked that there is an embedding of a phylogenetic X-tree in $\mathcal{N}$ that is not displayed by $\mathcal{N}'$. Moreover, if $C_{v'_2} = \{a\}$, then, by using (p_a, p_b), the arcs on P_a and P_q, and (u, w), it is again easily checked that there is an embedding of a phylogenetic X-tree in $\mathcal{N}$ that is not displayed by $\mathcal{N}'$. These contradictions imply that u is not a tree vertex.

Now assume that u is a reticulation. Let s denote the parent of u that is not t. Since $\mathcal{N}$ is acyclic, s is not on P_a. Also, s is not on P_q; otherwise, (t, u) is shortcut, contradicting that $\mathcal{N}$ is normal. As $\mathcal{N}$ is normal, (s, u) is not a shortcut and so there is a tree-path from s to a leaf ℓ_s, where $\ell_s \notin \{a, b\} \cup C_q$. Note that ℓ_s is not reachable from q; otherwise, s is reachable from q and so (t, u) is a shortcut in the normal network $\mathcal{N}$, contradiction.

Applying essentially the same argument to that when u is a tree vertex, we again obtain a contradiction to $T(\mathcal{N}) = T(\mathcal{N}')$ and conclude that P_a consists of the arc (t, p_a).

Now consider P_q and suppose that (t, u) is an arc on P_q. If u is a tree vertex, then there is a child vertex, w say, of u that is not on P_q, and so there is a tree-path from u to a leaf ℓ_w, where $\ell_w \notin V_q \cup \{a, b\}$. If $C_{v'_1} = \{a\}$, then, by using (q, p_b), the arcs on P_q, and (u, w), it is easily seen that there is an embedding of a phylogenetic X-tree in $\mathcal{N}$ that is not displayed by $\mathcal{N}'$. Moreover, if $C_{v'_2} = \{a\}$, then, by using (p_a, p_b), the arcs on P_q, and (u, w), it is again easily seen that there is an embedding of a phylogenetic X-tree in $\mathcal{N}$ that is not displayed by $\mathcal{N}'$. These contradictions imply that u is not a tree vertex, and so we may assume that u is a reticulation. Let s denote the parent of u that is not t. As $\mathcal{N}$ is normal, (s, u) is not a shortcut and there is a tree-path from s to a leaf ℓ_s, where $\ell_s \notin C_q \cup \{a, b\}$. Note that s is not reachable from q; otherwise, $\mathcal{N}$ has a directed cycle. Applying essentially the same argument to that when u is a tree vertex, we conclude that P_q consists of the arc (t, q). This completes the proof of (11.5). □

We complete the proof of Proposition 11 by considering v'_2 in $\mathcal{N}'$. First assume that v'_2 is a tree vertex or a leaf. Then, as $T(\mathcal{N}) = T(\mathcal{N}')$ and $\{V_{v'_1}, V_{v'_2}\} = \{\{a\}, V_q\}$, it follows that $\{C_{v'_1}, C_{v'_2}\} = \{\{a\}, C_q - b\}$. In particular, in combination with (11.3) we have the outcome shown in Fig. 6(a). Now assume that v'_2 is a reticulation. Let u'_2 denote the parent of v'_2 that is not q'. If (u'_2, v'_2) is not a shortcut, then there is a tree-path from u'_2 to a leaf not in $\{a, b\} \cup C_q$, in which case, by using (u'_2, v'_2) and not (q', v'_2), it follows from (11.5) that there is an embedding of a phylogenetic X-tree in $\mathcal{N}'$ not displayed by $\mathcal{N}$, a contradiction. So (u'_2, v'_2) is a shortcut. As $\mathcal{N}'$ is tree-child, u'_2 is a tree vertex and the child vertex of u'_2 that is not v'_2, say w', is also a tree vertex. If $w' \neq u'_1$, then there is a child vertex y' of w' that is the initial vertex of a tree-path to a leaf not in $\{a, b\} \cup C_q$. But then, by using (u'_2, v'_2) and (w', y'), it follows from (11.5) that there is an embedding of a phylogenetic X-tree in $\mathcal{N}'$ that is not displayed by $\mathcal{N}$, a contradiction. Thus $w' = u'_1$, and so (u'_2, u'_1) is an arc in $\mathcal{N}'$. Furthermore, as $T(\mathcal{N}) = T(\mathcal{N}')$, it follows that if $C_{v'_1} = V_{v'_1} = \{a\}$, then $C_{v'_2} = C_q$, while if $C_{v'_2} = V_{v'_2} = \{a\}$, then $C_{v'_1} = C_q$. Thus we have the outcome shown in Fig. 6(b), thereby completing the proof of the proposition. □

4 Recursion Lemmas

With the structural outcomes of Propositions 8, 10, and 11 in hand, we next establish the three lemmas that will allow the algorithm to recurse correctly. The proof of the first lemma is straightforward and omitted.

Lemma 12. *Let $\mathcal{N}$ and $\mathcal{N}'$ be normal and tree-child networks on X, respectively, and suppose that $\{a, b\}$ is a cherry of $\mathcal{N}$ and $\mathcal{N}'$. Then $T(\mathcal{N}) = T(\mathcal{N}')$ if and only if $T(\mathcal{N}\backslash b) = T(\mathcal{N}'\backslash b)$.*

Lemma 13. *Let $\mathcal{N}$ and $\mathcal{N}'$ be normal and tree-child networks on X, and suppose that $\{a, b\}$ is a reticulated cherry of $\mathcal{N}$ and $\mathcal{N}'$ as shown in Figs. 3 and 4, respectively. Then $T(\mathcal{N}) = T(\mathcal{N}')$ if and only if $T(\mathcal{N}\backslash(p_a, p_b)) = T(\mathcal{N}'\backslash(p'_a, p'_b))$.*

Proof. First observe that $T(\mathcal{N}) - T(\mathcal{N}\backslash(p_a, p_b))$ (resp. $T(\mathcal{N}') - T(\mathcal{N}'\backslash(p'_a, p'_b))$) consists of precisely the phylogenetic X-trees displayed by $\mathcal{N}$ (resp. $\mathcal{N}'$) in which $\{a, b\}$ is a cherry. Thus if $T(\mathcal{N}) = T(\mathcal{N}')$, then $T(\mathcal{N}\backslash(p_a, p_b)) = T(\mathcal{N}'\backslash(p'_a, p'_b))$. Suppose $T(\mathcal{N}\backslash(p_a, p_b)) = T(\mathcal{N}'\backslash(p'_a, p'_b))$, and let $\mathcal{T}$ be a phylogenetic X-tree displayed by $\mathcal{N}$. If $\{a, b\}$ is not a cherry in $\mathcal{T}$, then, by the observation, $\mathcal{N}\backslash(p_a, p_b)$, and therefore $\mathcal{N}'\backslash(p'_a, p'_b)$, displays $\mathcal{T}$. This implies that $\mathcal{N}'$ displays $\mathcal{T}$. So assume $\{a, b\}$ is a cherry in $\mathcal{T}$. Let $\mathcal{S}$ be an embedding of $\mathcal{T}$ in $\mathcal{N}$. Note that $\mathcal{S}$ must use the arc (p_a, p_b). Let $\mathcal{S}_1$ be the embedding in $\mathcal{N}$ of a phylogenetic X-tree $\mathcal{T}_1$ obtained from $\mathcal{S}$ by deleting (p_a, p_b) and adding (q, p_b). Since $\{a, b\}$ is not a cherry of $\mathcal{T}_1$, it follows that $\mathcal{N}'$ displays $\mathcal{T}_1$, that is, $\mathcal{N}'$ has an embedding $\mathcal{S}'_1$ of $\mathcal{T}_1$. Now, by replacing (q'_2, p'_b) with (p'_a, p'_b) in $\mathcal{S}'_1$, we have an embedding of $\mathcal{T}$ in $\mathcal{N}'$. Hence $\mathcal{N}'$ displays $\mathcal{T}$, and so $T(\mathcal{N}) \subseteq T(\mathcal{N}')$. Similarly, $T(\mathcal{N}') \subseteq T(\mathcal{N})$. Thus $T(\mathcal{N}) = T(\mathcal{N}')$. □

Lemma 14. *Let $\mathcal{N}$ and $\mathcal{N}'$ be normal and tree-child networks on X, respectively. Suppose that $\{a, b\}$ is a reticulated cherry of $\mathcal{N}$ as shown in Fig. 5, while $\mathcal{N}'$ has the structure local to leaves a and b as shown in either Fig. 6(a) or Fig. 6(b).*

(i) *If $C_{v'_1} = \{a\}$, then $T(\mathcal{N}) = T(\mathcal{N}')$ if and only if*

$$T(\mathcal{N}\backslash(p_a, p_b)) = \begin{cases} T(\mathcal{N}'\backslash(p'_b, v'_1)), & v'_2 \text{ a tree vertex or a leaf;} \\ T(\mathcal{N}'\backslash\{(p'_b, v'_1), (u'_2, v'_2)\}), & \text{otherwise.} \end{cases}$$

(ii) *If $C_{v'_2} = \{a\}$, then $T(\mathcal{N}) = T(\mathcal{N}')$ if and only if*

$$T(\mathcal{N}\backslash(p_a, p_b)) = \begin{cases} T(\mathcal{N}'\backslash(u'_1, v'_1)), & v'_2 \text{ a tree vertex or a leaf;} \\ T(\mathcal{N}'\backslash\{(u'_1, v'_1), (u'_2, v'_2)\}), & \text{otherwise.} \end{cases}$$

Proof. We shall prove (i). The proof of (ii) is similar and omitted. Suppose $C_{v'_1} = \{a\}$. For convenience, let $\mathcal{N}_1$ denote $\mathcal{N}\backslash(p_a, p_b)$. Furthermore, let $\mathcal{N}'_1$ denote $\mathcal{N}'\backslash(p'_b, v'_1)$ if v'_2 is a tree vertex or a leaf; otherwise, let $\mathcal{N}'_1$ denote $\mathcal{N}'\backslash\{(p'_b, v'_1), (u'_2, v'_2)\}$. We begin by observing that $T(\mathcal{N}) - T(\mathcal{N}_1)$ (resp. $T(\mathcal{N}') - T(\mathcal{N}'_1)$) consists of precisely the phylogenetic X-trees displayed by $\mathcal{N}$ (resp. $\mathcal{N}'$) in which $\{a, b\}$ is a cherry. Therefore if $T(\mathcal{N}) = T(\mathcal{N}')$, then $T(\mathcal{N}_1) = T(\mathcal{N}'_1)$.

For the converse, suppose that $T(\mathcal{N}_1) = T(\mathcal{N}'_1)$. Let $\mathcal{T}$ be a phylogenetic X-tree displayed by $\mathcal{N}$. If $\{a, b\}$ is not a cherry in $\mathcal{T}$, then, by the observation, $\mathcal{N}_1$, and therefore $\mathcal{N}'_1$, displays $\mathcal{T}$. It follows that $\mathcal{N}'$ displays $\mathcal{T}$. So assume $\{a, b\}$ is a cherry in $\mathcal{T}$. Let $\mathcal{S}$ be an embedding of $\mathcal{T}$ in $\mathcal{N}$. Since $\{a, b\}$ is a cherry in $\mathcal{T}$, the embedding $\mathcal{S}$ uses (p_a, p_b). Let $\mathcal{S}_1$ denote the embedding in $\mathcal{N}$ of a phylogenetic X-tree $\mathcal{T}_1$ obtained from $\mathcal{S}$ by deleting (p_a, p_b) and adding (q, p_b). Since $\{a, b\}$ is not a cherry in $\mathcal{T}_1$, it follows that $\mathcal{N}'$ has an embedding $\mathcal{S}'_1$ of $\mathcal{T}_1$. This embedding $\mathcal{S}'_1$ must use (u'_1, v'_1). By replacing (u'_1, v'_1) with (p'_b, v'_1) in $\mathcal{S}'_1$, it is easily seen that we have an embedding of $\mathcal{T}$ in $\mathcal{N}'$. Hence $\mathcal{N}'$ displays $\mathcal{T}$ and so $T(\mathcal{N}) \subseteq T(\mathcal{N}')$.

Now let $\mathcal{T}'$ be a phylogenetic X-tree displayed by $\mathcal{N}'$. If $\{a, b\}$ is not a cherry, then, by the observation, $\mathcal{N}'_1$, and therefore $\mathcal{N}_1$, displays $\mathcal{T}'$. So $\mathcal{N}$ displays $\mathcal{T}'$. Assume $\{a, b\}$

is a cherry in $\mathcal{T}'$. Let $\mathcal{S}'$ be an embedding of $\mathcal{T}'$ in $\mathcal{N}'$. As $\{a, b\}$ is a cherry in $\mathcal{T}'$ and as any embedding of $\mathcal{T}'$ in $\mathcal{N}'$ must use (u'_1, q'), (q', p'_b), (p'_b, b) and, if it exists, (u'_2, u'_1), it is easily seen that we may choose $\mathcal{S}'$ so that it uses (p'_b, v'_1) and (q', v'_2). Let $\mathcal{S}'_1$ be the embedding in $\mathcal{N}'$ of a phylogenetic X-tree $\mathcal{T}'_1$ obtained from $\mathcal{S}'$ by deleting (p'_b, v'_1) and adding (u'_1, v'_1). Since $\{a, b\}$ is not a cherry in $\mathcal{T}'_1$, it follows that $\mathcal{N}$ has an embedding $\mathcal{S}_1$ of $\mathcal{T}'_1$. This embedding $\mathcal{S}_1$ must use (q, p_b). By replacing (q, p_b) with (p_a, p_b) in $\mathcal{S}_1$, it is easily checked that we obtain an embedding of $\mathcal{T}'$ in $\mathcal{N}$. Thus $\mathcal{N}$ displays $\mathcal{T}'$, and so $T(\mathcal{N}') \subseteq T(\mathcal{N})$. We conclude that $T(\mathcal{N}) = T(\mathcal{N}')$. □

5 The Algorithm

We now give a formal description of the algorithm SameDisplaySet for deciding if $T(\mathcal{N}) = T(\mathcal{N}')$, where $\mathcal{N}$ and $\mathcal{N}'$ are normal and tree-child networks on X, respectively. Immediately after the description of the algorithm, we show that SameDisplaySet works correctly and analyse its running time. We end the section by briefly describing how to construct a tree displayed by exactly one of $\mathcal{N}$ and $\mathcal{N}'$ if $T(\mathcal{N}) \neq T(\mathcal{N}')$.

SameDisplaySet
Input: Normal and tree-child networks $\mathcal{N}$ and $\mathcal{N}'$ on X, respectively.
Output: *No* if $T(N) \neq T(\mathcal{N}')$, and *Yes* if $T(\mathcal{N}) = T(\mathcal{N}')$.

1. Delete all trivial shortcuts in $\mathcal{N}'$ and suppress all resulting vertices of in-degree one and out-degree one, and denote the resulting normal and tree-child networks on X as $\mathcal{N}_0$ and $\mathcal{N}'_0$, respectively.

2. Set $i = 0$.

3. If the leaf set of $\mathcal{N}_i$ has size two, return *yes*. Else, find a cherry or a reticulated cherry, say $\{a, b\}$, of $\mathcal{N}_i$.

4. If $\{a, b\}$ is a cherry, then determine if $\{a, b\}$ is a cherry of $\mathcal{N}'_i$.

   (a) If no, then return *No*.

   (b) Else, set $\mathcal{N}_{i+1} = \mathcal{N}_i \backslash b$ and set $\mathcal{N}'_{i+1} = \mathcal{N}'_i \backslash b$. Go to Step 6.

5. Else, $\{a, b\}$ is a reticulated cherry of $\mathcal{N}_i$, where the parent of b is a reticulation.

   (a) If the parent of b in $\mathcal{N}'_i$ is a reticulation, then determine if, up to isomorphism, the structure in $\mathcal{N}'_i$ local to a and b is as shown in Fig. 4.

      (i) If no, then return *No*.

      (ii) Else, set $\mathcal{N}_{i+1} = \mathcal{N}_i \backslash (p_a, p_b)$ and set $\mathcal{N}'_{i+1} = \mathcal{N}'_i \backslash (p'_a, p'_b)$. Go to Step 6.

   (b) If the parent of b in $\mathcal{N}'_i$ is the root, then return *No*.

   (c) Else, the parent of b in $\mathcal{N}'_i$ is a tree vertex. Determine if, up to isomorphism, the structures in $\mathcal{N}_i$ and $\mathcal{N}'_i$ local to a and b are as shown in Fig. 5 and Fig. 6(a) or Fig. 6(b), respectively.

(i) If no, then return *No*.

(ii) Else, set $\mathcal{N}_{i+1}$ to be the normal network $\mathcal{N}_i \backslash (p_a, p_b)$. Further, if $C_{v'_1} = \{a\}$, set $\mathcal{N}'_{i+1}$ to be the tree-child network $\mathcal{N}'_i \backslash \{(p'_b, v'_1), (u'_2, v'_2)\}$. Otherwise, if $C_{v'_2} = \{a\}$, set $\mathcal{N}'_{i+1}$ to be the tree-child network $\mathcal{N}'_i \backslash \{(u'_1, v'_1), (u'_2, v'_2)\}$. Go to Step 6.

6. Increase i by 1 and go back to Step 3.

Theorem 1 immediately follows from the next theorem.

Theorem 15. *Let $\mathcal{N}$ and $\mathcal{N}'$ be normal and tree-child networks on X, respectively. Then* SameDisplaySet *applied to $\mathcal{N}$ and $\mathcal{N}'$ correctly determines if $T(\mathcal{N}) = T(\mathcal{N}')$. Furthermore,* SameDisplaySet *runs in time quadratic in the size of X.*

Proof. Ignoring the running time, by Lemma 7, we may assume that $\mathcal{N}'$ has no trivial shortcuts. Therefore, as there is exactly one phylogenetic tree for when $|X| = 2$, the fact that SameDisplaySet correctly determines whether or not $T(\mathcal{N}) = T(\mathcal{N}')$ follows by combining Propositions 8, 10, and 11 and Lemmas 12, 13, and 14. Thus to complete the proof of the theorem, it suffices to show that the running time of the algorithm is quadratic in the size of X.

Let $n = |X|$ and note that the total number of vertices in a tree-child network is linear in the size of X (see [13]). Thus both $\mathcal{N}$ and $\mathcal{N}'$ have at most $O(n)$ vertices in total. Now consider SameDisplaySet applied to $\mathcal{N}$ and $\mathcal{N}'$. Step 1 is a preprocessing step that considers, for each reticulation v in $\mathcal{N}'$, whether there is an arc joining the parents of v. Since this takes constant time for each reticulation, this step takes $O(n)$ time to complete. For iteration i, Step 3 finds a cherry or a reticulated cherry in $\mathcal{N}_i$. Since $\mathcal{N}_i$ is normal, one way to do this is to construct a maximal path that starts at the root of $\mathcal{N}_i$ and ends at a tree vertex. The two leaves below this tree vertex, say a and b, either form a cherry or a reticulated cherry in $\mathcal{N}_i$. As the total number of vertices in $\mathcal{N}_i$ is $O(n)$, this takes time $O(n)$. If $\{a, b\}$ is a cherry in $\mathcal{N}_i$, then Step 4 determines whether or not $\{a, b\}$ is a cherry in $\mathcal{N}'_i$ and, if so, deletes b in both $\mathcal{N}_i$ and $\mathcal{N}'_i$ and suppresses any resulting vertex of in-degree one and out-degree one. Therefore Step 4 takes constant time. On the other hand, if $\{a, b\}$ is a reticulated cherry in $\mathcal{N}_i$, then Step 5 is called. Similar to Step 4, this step considers the structure in $\mathcal{N}_i$ and $\mathcal{N}'_i$ local to a and b, but is less straightforward. In terms of running time, the longest part of the step to complete is in determining the cluster and visibility sets of certain vertices. A single postorder transversal of each of $\mathcal{N}_i$ and $\mathcal{N}'_i$ can be used to determine all cluster sets of $\mathcal{N}_i$ and $\mathcal{N}'_i$. Since $\mathcal{N}$ and $\mathcal{N}'$ are both binary, the number of arcs in each is $O(n)$, so this takes time $O(n)$. Furthermore, to determine the visibility set of a vertex u of $\mathcal{N}_i$, we delete u and its incident arcs, and check, for each leaf ℓ, whether the resulting rooted acyclic directed graph, D_i say, has a path from the root to ℓ. That is, loosely speaking, we want to find the 'cluster set', X' say, of the root in D_i. It then follows that the visibility set of u is $X_i - X'$, where X_i is the leaf set of $\mathcal{N}_i$. A single postorder transversal of D_i is sufficient to determine X', so this takes time $O(n)$. Similarly, the visibility set of a vertex in $\mathcal{N}'_i$ can be found in this

way. As we only need to find the visibility sets of three vertices in $\mathcal{N}_i$ and $\mathcal{N}_i'$, the total time to determine the necessary visibility sets is $O(n)$. Thus the time to complete Step 5, including the deletion of certain arcs, is $O(n)$. Hence, each iteration of SameDisplaySet takes $O(n)$. Since each iteration deletes at least one vertex or arc in each of $\mathcal{N}$ and $\mathcal{N}'$, it follows that there are $O(n)$ iterations, and so the entire algorithm runs in time $O(n^2)$. □

Algorithm returns *No*

Suppose that SameDisplaySet is applied to normal and tree-child networks $\mathcal{N}$ and $\mathcal{N}'$ on X, respectively, and returns *No*. In this case, it is natural to ask for a phylogenetic X-tree displayed by exactly one of $\mathcal{N}$ and $\mathcal{N}'$. Without going into detail, it is straightforward to amend the algorithm so that such a tree is constructed. To illustrate, assume that SameDisplaySet returns *No* at Step 5(a)(i). Then, at some iteration i, the normal network $\mathcal{N}_i$ has a reticulated cherry $\{a, b\}$, in which the parent of b is a reticulation, and the parent of b in the tree-child network $\mathcal{N}_i'$ is a reticulation, but the structure of $\mathcal{N}_i'$ local to a and b is not as that shown in Fig. 4. Comparing this figure with Fig. 3, this implies that, while the display set of $\mathcal{N}_i$ contains a tree with cherry $\{a, b\}$, a tree with generalised cherry $\{b, V_q\}$, and a tree with generalised cherry $\{b, C_q - b\}$, the display set of $\mathcal{N}_i'$ does not contain a tree of one of these three types. By choosing an embedding in $\mathcal{N}_i$ of such a tree in the display set of $\mathcal{N}_i$ and then reversing the steps in the algorithm that have been performed up to Step 5(a)(i) in iteration i, we can construct a subdivision of a tree displayed by $\mathcal{N}$ but not displayed by $\mathcal{N}'$.

If there exists a phylogenetic tree $\mathcal{T}$ that is displayed by $\mathcal{N}$ and not displayed by $\mathcal{N}'$, then it may be possible for practitioners, who compare $\mathcal{N}$ and $\mathcal{N}'$ with a biological question in mind, to interpret the presence and absence of $\mathcal{T}$ in the display set of $\mathcal{N}$ and $\mathcal{N}'$, respectively, in a biologically meaningful way. For example, if $\mathcal{T}$ is know to be a gene tree that is associated with a DNA segment used to reconstruct $\mathcal{N}$ and $\mathcal{N}'$, then the fact that $\mathcal{T}$ is not displayed by $\mathcal{N}'$ may indicate that $\mathcal{N}$ more faithfully represents the evolutionary history of the species under consideration than $\mathcal{N}'$.

Acknowledgments

We thank the two anonymous referees for their constructive comments.

References

[1] M. Bordewich and C. Semple. Reticulation-visible networks. *Adv. Appl. Math.*, 76:114–141, 2016.

[2] M. Bordewich and C. Semple. Determining phylogenetic networks from inter-taxa distances. *J. Math. Bio.*, 73:283–303, 2016.

[3] G. Cardona, F. Rosselló, and G. Valiente. Comparison of tree-child phylogenetic networks. *IEEE/ACM Trans. Comput. Biol. Bioinf.*, 6:552–569, 2009.

[4] P. Cordue, S. Linz, and C. Semple. Phylogenetic networks that display a tree twice. *B. Math. Biol.*, 76:2664–2679, 2014.

[5] J. Döcker, S. Linz, and C. Semple. Displaying trees across two phylogenetic networks. *Theor. Comput. Sci.*, 796:129–146, 2019.

[6] P. Gambette and K. T. Huber. On encodings of phylogenetic networks of bounded level. *J. Math. Bio.*, 65:157–180, 2012.

[7] A. D. M. Gunawan. Solving the Tree Containment problem for reticulation-visible networks in linear time. In *Algorithms for Computational Biology*, pages 24–36. Springer, 2018.

[8] A. D. M. Gunawan, B. DasGupta, and L. Zhang. A decomposition theorem and two algorithms for reticulation-visible networks. *Inform. Comput.*, 252:161–175, 2017.

[9] L. van Iersel and V. Moulton. Trinets encode tree-child and level-2 phylogenetic networks. *J. Math. Bio.*, 68:1707–1729, 2014.

[10] L. van Iersel, C. Semple, and M. Steel. Locating a tree in a phylogenetic network. *Inform. Process. Lett.*, 110:1037–1043, 2010.

[11] I. A. Kanj, L. Nakhleh, C. Than, and G. Xia. Seeing the trees and their branches in the network is hard. *Theor. Comput. Sci.*, 401:153–164, 2008.

[12] S. Linz, K. St John, and C. Semple. Counting trees in a phylogenetic network is #P-complete. *SIAM J. Comput.*, 42:1768–1776, 2013.

[13] C. McDiarmid, C. Semple, and D. Welsh. Counting phylogenetic networks. *Ann. Comb.*, 19:205–224, 2015.

[14] N. F. Müller, U. Stolz, G. Dudas, T. Stadler, and T. G. Vaughan. Bayesian inference of reassortment networks reveals fitness benefits of reassortment in human influenza viruses. *P. Natl. Acad. Sci. USA*, 117:17104–17111, 2020.

[15] L. Nakhleh, G. Jin, F. Zhao, and J. Mellon-Crummey. Reconstructing phylogenetic networks using maximum parsimony. In *Proceedings of the 2005 IEEE Computational Systems Bioinformatics Conference*, pages 93–102. IEEE, 2005.

[16] C. Semple. Phylogenetic networks with every embedded phylogenetic tree a base tree. *B. Math. Biol.*, 78:32–137, 2016.

[17] C. Solís-Lemus, P. Bastide, and C. Ané. PhyloNetworks: a package for phylogenetic networks. *Mol. Biol. Evol.*, 34:3292–3298, 2017.

[18] L. J. Stockmeyer. The polynomial-time hierarchy. *Theor. Comput. Sci.*, 3:1–22, 1978.

[19] S. J. Willson. Properties of normal phylogenetic networks. *B. Math. Biol.*, 72:340-358, 2010.

[20] S. J. Willson. Regular networks can be uniquely constructed from their trees. *IEEE/ACM Trans. Comput. Biol. Bioinf.*, 8:785–796, 2011.

[21] S. J. Willson. Tree-average distances on certain phylogenetic networks have their weights uniquely determined. *Algorithm. Mol. Biol.*, 7:13, 2012.

[22] J. Zhu, D. Wen, Y. Yu, H. M. Meudt, and L. Nakhleh. Bayesian inference of phylogenetic networks from bi-allelic genetic markers. *PLoS Comput. Biol.*, 14(1):e1005932, 2018.

1.5 The monotone satisfiability problem with bounded variable appearances

The published version [DDD18a] of the following paper is available online at the following URL: `https://doi.org/10.1142/S0129054118500168`.

International Journal of Foundations of Computer Science

The Monotone Satisfiability Problem with Bounded Variable Appearances

Andreas Darmann

Institute of Public Economics, University of Graz,
Universitaetsstr. 15, 8010 Graz, Austria
andreas.darmann@uni-graz.at

Janosch Döcker

Department of Computer Science, University of Tuebingen
Sand 12, 72076 Tuebingen, Germany
janosch.doecker@uni-tuebingen.de

Britta Dorn

Department of Computer Science, University of Tuebingen
Sand 12, 72076 Tuebingen, Germany
britta.dorn@uni-tuebingen.de

Received (Day Month Year)
Accepted (Day Month Year)
Communicated by (xxxxxxxxxx)

The prominent Boolean formula satisfiability problem SAT is known to be $\mathcal{NP}$-complete even for very restricted variants such as 3-SAT, Monotone 3-SAT, or Planar 3-SAT, or instances with bounded variable appearance. We settle the computational complexity status for two variants with bounded variable appearance: We show that Planar Monotone Sat—the variant of Monotone Sat in which the incidence graph is required to be planar—is $\mathcal{NP}$-complete even if each clause consists of at most three distinct literals and each variable appears exactly three times, and that Monotone Sat is $\mathcal{NP}$-complete even if each clause consists of three distinct literals and each variable appears exactly four times in the formula. The latter confirms a conjecture stated in scribe notes [DKY14] of an MIT lecture by Eric Demaine. In addition, we provide hardness results with respect to bounded variable appearances for two variants of Planar Monotone Sat.

Keywords: (Planar) Monotone Satisfiability; bounded variable appearance; computational complexity; 3-Satisfiability.

1. Introduction

The satisfiability problem (Sat) for Boolean formulae in conjunctive normal form—and especially 3-Satisfiability (3-Sat), where each clause contains exactly three distinct literals—is frequently used to prove $\mathcal{NP}$-hardness of decision problems[a].

[a]For an introduction to the theory of $\mathcal{NP}$-completeness we refer to Garey and Johnson [GJ79].

However, the complexity status of the satisfiability problem and its many variants is not only relevant in order to provide a base problem for deriving $\mathcal{NP}$-hardness, but obviously interesting in its own right. In his seminal paper, Tovey [Tov84] proved that 3-Sat is $\mathcal{NP}$-complete even if every variable appears in at most four clauses; in contrast, he showed that any instance of 3-Sat in which each variable appears at most three times is always satisfiable.

Here, we consider the question whether similar results with respect to bounded variable appearances also hold for more restricted variants of Sat. In particular, we consider the monotone satisfiability problem (Monotone Sat), where each clause is *monotone*, i.e., the literals of the clause are either all positive or all negative, and *planar* variants of this problem in the case of bounded variable appearances. In an instance of the planar satisfiability problem (Planar Sat), the incidence graph of the formula—i.e., the undirected bipartite graph whose vertices correspond to the variables and clauses of the instance and whose edges connect a variable vertex with a clause vertex whenever the variable appears (positively or negatively) in the respective clause—is required to be planar. Lichtenstein [Lic82] proved Planar Sat to be $\mathcal{NP}$-complete even in the case that each clause consists of at most three literals. Dahlhaus et al. [DJP$^+$94] showed that Planar Sat remains $\mathcal{NP}$-complete even when restricted to instances in which each clause consists of two or three distinct literals and every variable appears exactly three times (see also Section 3).

Planar Monotone Sat, the restriction of Planar Sat to monotone instances, remains $\mathcal{NP}$-complete even if each clause contains at most three literals and a rectilinear representation[b] is given (de Berg and Khosravi [dBK12]). We show that Planar Monotone Sat is $\mathcal{NP}$-complete even if each clause contains two or three literals and each variable appears at most (or exactly) three times; this hardness result also holds if every 3-clause contains only positive literals and each variable appears negated exactly once. Since Sat can be solved in polynomial time when restricted to instances with at most two appearances per variable [Tov84], this settles the computational complexity status for Planar Monotone Sat with at most three literals per clause with respect to the number of variable appearances.

Concerning the general variant Monotone Sat, we answer a question stated as a conjecture attributed to S. Eisenstat in scribe notes [DKY14] of an MIT lecture[c]: there, it is conjectured that Monotone 3-Sat is $\mathcal{NP}$-hard even when restricted to instances in which each variable appears at most five times.

Monotone 3-Sat is known to be $\mathcal{NP}$-complete (see the works by Gold [Gol78]

[b] A rectilinear representation of an instance of Planar Sat is a crossing-free drawing of the incidence graph as follows (see de Berg and Khosravi [dBK12] and Knuth and Raghunatan [KR92]): variables and clauses are represented by rectangles such that all variable-rectangles are drawn along a horizontal line; the edges linking the variables with the clauses are vertical segments; moreover, in a monotone rectilinear representation all positive clauses are drawn above the variables while all negative clauses are drawn below the variables.

[c] Algorithmic Lower Bounds: Fun with Hardness Proofs (Fall '14), Prof. E. Demaine, Teaching assistants: S. Eisenstat, J. Lynch.

and Li [Li97]). However, the complexity status of Monotone 3-Sat in the case of bounded variable appearances has been open. We show that Monotone 3-Sat remains $\mathcal{NP}$-complete even if the number of appearances of each variable is bounded by 4 (this result also holds if each variable appears exactly four times). Therewith, this result not only confirms the above conjecture, but also settles the computational complexity status of Monotone 3-Sat for a bounded number of variable appearances, since, as each instance of 3-Sat is satisfiable if the number of appearances of each variable is bounded by 3 [Tov84], so is any instance of Monotone 3-Sat.

Moreover, we consider the variant of Planar Monotone Sat in which each clause consists of three not necessarily distinct literals (duplicates of literals are allowed within a clause); for that variant, we provide hardness results for each of the special cases that each variable appears exactly four times, or each variable appears five times and the incidence graph is biconnected.

The paper is structured as follows. In Section 2 we introduce the formal framework of this paper. Sections 3, 4, and 5 contain computational complexity results for variants of the monotone satisfiability problem with bounded variable appearances: In Section 3, Planar Monotone Sat is considered, while in Section 4, the focus is laid on the general (non-planar) problem Monotone Sat; Section 5 deals with the variation of the planar monotone satisfiability problem where duplicates of literals are allowed within a clause. Section 6 concludes the paper.
Finally, we refer to Darmann and Döcker [DD16] and Darmann et al. [DDD16] for preliminary versions of this paper.

2. Preliminaries

Throughout this work, we are concerned with the satisfiability problem (Sat) for Boolean formulae in conjunctive normal form. An instance $\mathcal{I} = (\mathcal{V}, \mathcal{C})$ of Sat consists of a finite set $\mathcal{V}$ of *variables* and a set $\mathcal{C}$ of *clauses* over $\mathcal{V}$. A clause is a disjunction of *literals*: Each variable x induces a positive literal x and a negative literal $\bar{x}$ (the negation of variable x); the set of literals is given by $\mathcal{L} = \{x, \bar{x} \mid x \in \mathcal{V}\}$. Unless explicitly stated otherwise, a clause consists of distinct literals, i.e., it can be written as a subset of $\mathcal{L}$; only in one variation of the problems considered we use multisets in order to allow for duplicates of literals within a clause. An *assignment* of the variables $\mathcal{V}$ maps each variable to either true or false. A positive (negative) literal is true under an assignment if and only if the corresponding variable is assigned the value true (false). A clause is satisfied under an assignment β if and only if at least one contained literal is set true by β.
Now, the satisfiability problem (Sat) is as follows: Given an instance $\mathcal{I} = (\mathcal{V}, \mathcal{C})$ made up of a set of variables $\mathcal{V}$ and a set $\mathcal{C}$ of clauses over $\mathcal{V}$, does there exist an assignment of the variables satisfying all clauses?
In the positive case, the corresponding formula, i. e., the set of clauses, is satisfiable; otherwise the formula is unsatisfiable.

We will consider different variants of SAT restricting the structure of the considered instances. For instance, a clause is called *positive (negative)* if it contains only positive (negative) literals. A clause is called *monotone* if it is either positive or negative. A *mixed clause* is a clause which is not monotone, i.e., it contains at least one positive and at least one negative literal. MONOTONE SAT refers to the restriction of SAT to instances with monotone clauses only.

PLANAR SAT refers to instances of SAT with a planar incidence graph. Formally, given a set $\mathcal{V}$ of variables and a formula, i.e., a set of clauses $\mathcal{C}$ over $\mathcal{V}$, the *incidence graph* $G_{\mathcal{V},\mathcal{C}}$ of the formula is defined by $G_{\mathcal{V},\mathcal{C}} := (V, E)$ with $V := \mathcal{V} \cup \mathcal{C}$ and $E := \{\{v, C\} \mid v \in C \vee \bar{v} \in C$, where $v \in \mathcal{V}$ and $C \in \mathcal{C}\}$.

In what follows, by SAT-s (respectively SAT-Es) we denote the restriction of SAT to instances in which each variable appears at most s times (respectively exactly s times), with $s \in \mathbb{N}$.
A *k-clause* consists of exactly k distinct literals, $k \in \mathbb{N}$. By $(k, k+1)$-SAT we denote the restriction of SAT to instances in which each clause is either a k-clause or a $(k+1)$-clause. Finally, k-SAT* denotes the variant of SAT in which each clause consists of k (not necessarily distinct) literals, i.e., a clause is a multiset of literals and hence may contain the same literal more than once (duplicates of literals are allowed within a clause). The variant $(k, k+1)$-SAT* is defined in an analogous manner.

With the above definitions, by means of prefixes and suffixes, the variants of SAT considered in this work can be described as follows. As a prefix of the problem name SAT, we use an element of

$$\{\text{PLANAR}, \lambda\} \times \{\text{MONOTONE}, \lambda\} \times \{(2,3), 3\},$$

and as a suffix, we use an element of

$$\{^*, \lambda\} \times \{\text{E}, \lambda\} \times \{k \mid k \geq 1\},$$

where λ is the empty word, and where we omit commas in the notation. E.g., PLANAR MONOTONE $(2,3)$-SAT-E3 is the restriction of SAT to instances with planar incidence graph and monotone clauses, where each clause consists of exactly 2 or exactly 3 distinct literals and each variable appears exactly three times.

3. Hardness of PLANAR MONOTONE SAT with bounded variable appearances

We start with investigating the planar version of the monotone satisfiability problem. In the planar variants, the incidence graph of the formula—i.e., the undirected bipartite graph consisting of variable-vertices, clause-vertices and edges that connect a variable-vertex with a clause-vertex if and only if the variable appears (positively or negatively) in the clause—is required to be planar. For an introduction to the theory of planar graphs we refer to, e.g., Bondy and Murty [BM08, Chapter 10].

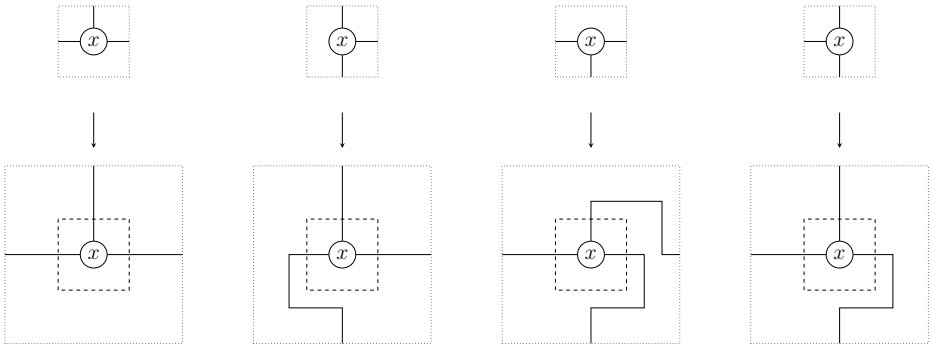


Figure 1. Local replacement of a variable vertex x. After application of this rule the drawing of the vertex locally looks as shown in the dashed square.

Our main result is hardness of a restricted variant of Planar Monotone (2, 3)-Sat-3, both for at most and exactly three appearances of each variable.

Theorem 1. Planar Monotone (2, 3)-Sat-3 *is $\mathcal{NP}$-complete even if every variable appears at least twice, every 3-clause contains only positive literals and each variable appears negated exactly once.*

Proof. We show $\mathcal{NP}$-hardness by a reduction from Planar $(2,3)$-Sat-E3: each clause consists of two or three distinct literals, each variable appears in exactly three clauses. Dahlhaus et al. [DJP$^+$94] have shown that this restricted version of Planar Sat remains $\mathcal{NP}$-complete even if for each variable, one of its literals appears in two clauses and the other literal in one clause. Let $\mathcal{I}$ be such an instance of Planar $(2,3)$-Sat-E3. A planar and orthogonal drawing of the incidence graph on a grid of size $n \times n$ can be computed in linear time using the algorithm of Biedl and Kant [BK98]. To this drawing we apply the local replacements shown in Figure 1, so that for each variable, the outgoing edges of the corresponding vertex are locally drawn identically (see the dashed squares in Figure 1).

For every variable, we replace the dashed square with the construction shown in Figure 2 (this gadget is inspired by the one given by Dahlhaus et al. [DJP$^+$94, p. 18] and the one by de Berg and Khosravi [dBK12, p. 6]). For the ith appearance of a variable x, we create two new variables x_i and a_i. Observe that the ring structure forces us to assign the same truth value to all x_i, and consequently the opposite truth value to all a_i.

If, in its ith appearance, x appears non-negated in the corresponding clause C_{x_i} of the original instance, we replace x with x_i and just use the outgoing dashed line of x_i. Otherwise, we replace $\bar{x}$ with a_i in C_{x_i}, and replace the edge $\{x, C_{x_i}\}$ with $\{a_i, C_{x_i}\}$. The graph remains planar since we only need to reroute the dashed line to a_i by using the corresponding dotted line instead. Now, it is not hard to see that

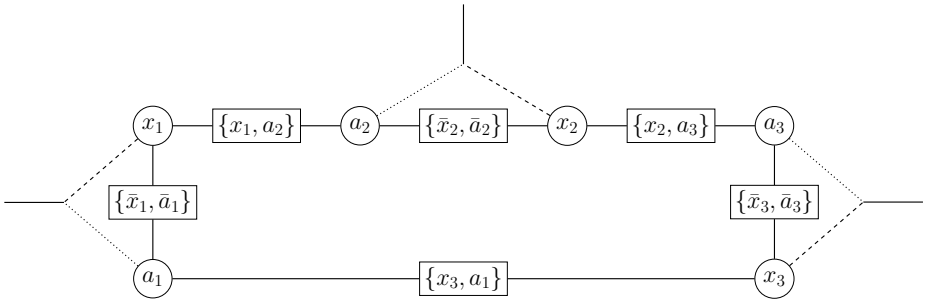


Figure 2. Local replacement of a variable vertex x with three appearances. The gadget introduces two variables for each appearance, x_i and a_i, and enforces that either all x_i have to be set true and all a_i false or the other way round. Thus, we can replace all negated appearances of x by the corresponding unnegated a_i's while preserving equisatisfiability. In the visual representation that means: The dotted line is used if the corresponding appearance of the variable is negated; otherwise the dashed line is used.

the resulting instance $\mathcal{I}'$ is equisatisfiable: Let ϕ be a satisfying truth assignment in instance $\mathcal{I}$, then define the truth assignment ϕ' for instance $\mathcal{I}'$ by setting, for all i, $\phi'(x_i)$ to true and $\phi'(a_i)$ to false if x is set true under ϕ, and $\phi'(x_i)$ to false and $\phi'(a_i)$ to true otherwise. As observed above, by the ring structure of the gadget ϕ' is a feasible truth assignment. Now, for any clause in $\mathcal{I}$, some literal x (or $\bar{x}$) in the clause is set true by ϕ; by construction of ϕ', a literal x_i (or a_i) in the corresponding clause in $\mathcal{I}'$ is set true by ϕ' as well. Also, since x_i and a_i are assigned opposite truth values, all clauses in the ring structure (Fig. 2) are satisfied.
On the other hand, for any satisfying truth assignment ϕ' in $\mathcal{I}'$ we know that for any choice of x, by the ring structure of the gadget the same truth value is assigned to all x_i, $i \in \{1, 2, 3\}$; thus, analogously to above it follows that the assignment ϕ for instance $\mathcal{I}$ that sets literal x true if and only if, for all i, x_i is set true by ϕ' is satisfying for instance $\mathcal{I}$.
Note that in $\mathcal{I}'$ every clause contains two or three distinct literals since we introduced only monotone 2-clauses. All clauses of the original instance are replaced with positive clauses containing the same number of literals. Hence, every clause is monotone and all 3-clauses are positive. We replace every variable with the construction described above, so every variable appears either two or three times; either a_i appears two times and x_i three times or the other way round. Moreover, recalling that only the clauses introduced in the gadget contain negative literals in the final instance, we can conclude that every variable appears negated exactly once. □

We remark that $\mathcal{NP}$-hardness of the general Planar Monotone (2, 3)-Sat-3 problem can also be shown directly by applying Gold's replacement rule [Gol78, p. 314f] to the restricted variant of Planar (2, 3)-Sat-E3 used in the above reduction. Gold's replacement rule replaces a mixed clause with two monotone clauses as

follows. Consider any mixed clause $C = C^+ \cup C^-$, where C^+ contains the positive literals of C and C^- the negative literals, respectively. Then, according to Gold's rule for any such mixed clause C we create a new variable u and replace C with the two clauses $C^+ \cup \{u\}$ and $C^- \cup \{\bar{u}\}$, which yields an equisatisfiable instance with one mixed clause less. Note that either both C^+ and C^- are made up of exactly one literal or one of them is made up of one literal and the other one of two literals. Thus, the introduced clauses are either of size 2 each or one is of size two and the other one of size 3.

However, note that for any instance $\mathcal{I} = (\mathcal{V}, \mathcal{C})$ of Planar Monotone (2, 3)-Sat-3 and any variable $x \in \mathcal{V}$ which appears at most twice, we can construct an equisatisfiable instance $\mathcal{I}' = (\mathcal{V}', \mathcal{C}')$ of Planar Monotone (2, 3)-Sat-3 in which the number of appearances of x is increased by exactly one by adding to $\mathcal{C}$ the clauses $\{x, u, v\}$, $\{u, v\}$, $\{\bar{u}, \bar{v}\}$, where u, v are newly introduced variables. As a consequence, from Theorem 1 we obtain the following corollary.

Corollary 2. Planar Monotone (2, 3)-Sat-E3 *is $\mathcal{NP}$-complete even if every 3-clause contains only positive literals and each variable appears negated exactly once.*

4. Hardness of Monotone Sat with bounded variable appearances

We now turn to investigating the computational complexity of the (non-planar) problem Monotone 3-Sat-s, $s \geq 4$.

Theorem 3. Monotone 3-Sat-4 *is $\mathcal{NP}$-complete.*

Proof. We give a reduction from Planar Monotone (2, 3)-Sat-3. Starting with an instance of Planar Monotone (2, 3)-Sat-3, the goal is to get rid of the clauses of size 2 while preserving equisatisfiability. We show a way to replace a negative 2-clause by a finite number of monotone 3-clauses, and at the end of the proof we remark how the case of a positive 2-clause can be dealt with analogously.
In particular, we present a finite set of monotone 3-clauses $\mathcal{C}_z$ such that no variable appears more than four times and a designated variable z appears exactly three times, and show that this set of clauses is satisfiable if and only if z is set to true. If there is a negative 2-clause, i.e., a clause of the form $\{\bar{x}, \bar{y}\}$ in the instance, we replace this clause with $\{\bar{x}, \bar{y}, \bar{z}\}$ and add $\mathcal{C}_z$ to the instance. The result is an equisatisfiable Monotone (2, 3)-Sat-4 instance with one negative 2-clause less than in the original instance. Of course, all variables appearing in $\mathcal{C}_z$ are newly created. The set $\mathcal{C}_z$ is given by the following 25 clauses.

(1) $\{u, w, z\}$
(2) $\{u, v, z\}$
(3) $\{\bar{w}, \bar{v}, \bar{g}\}$
(4) $\{\bar{w}, \bar{v}, \bar{h}\}$
(5) $\{\bar{w}, \bar{v}, \bar{i}\}$
(6) $\{g, h, i\}$
(7) $\{\bar{m}, \bar{n}, \bar{g}\}$
(8) $\{\bar{m}, \bar{n}, \bar{h}\}$
(9) $\{\bar{m}, \bar{n}, \bar{i}\}$
(10) $\{m, a, b\}$
(11) $\{n, a, b\}$
(12) $\{\bar{u}, \bar{a}, \bar{r}\}$
(13) $\{\bar{u}, \bar{b}, \bar{r}\}$
(14) $\{r, z, f\}$
(15) $\{\bar{d}, \bar{e}, \bar{a}\}$
(16) $\{\bar{d}, \bar{e}, \bar{b}\}$
(17) $\{p, q, d\}$
(18) $\{p, q, e\}$
(19) $\{\bar{f}, \bar{p}, \bar{c}\}$
(20) $\{\bar{f}, \bar{q}, \bar{c}\}$
(21) $\{r, c, j\}$
(22) $\{\bar{j}, \bar{p}, \bar{k}\}$
(23) $\{\bar{j}, \bar{q}, \bar{k}\}$
(24) $\{k, c, \ell\}$
(25) $\{\bar{\ell}, \bar{j}, \bar{f}\}$

Assume that the above set of clauses is satisfiable by a truth assignment in which z is set false.
First, we show that this implies that u has to be set true. If u is set false, then the first two clauses imply that both w and v need to be set true. Clauses $3, 4, 5$ thus yield that all three of g, h, i have to be set false, in contradiction with clause 6. Thus, u has to be set true.
By clause 6 at least one of g, h, i has to be set true. Thus, clauses $7, 8, 9$ imply that at least one of m, n has to be set false. As a consequence, clauses $10, 11$ yield that at least one of a, b needs to be set true. In turn, by clauses $12, 13$ this means that r has to be set false (recall that u is set true). Since both r, z are set false, f must be set true due to clause 14. By the fact that at least one of a, b is true, clauses $15, 16$ imply that at least one of d, e is set false. In turn, by the next two clauses this means that at least one of p, q must be set true. In addition, recalling that f is set true, clauses $19, 20$ imply that c has to be set false. Also recalling that r is set false, this means that j has to be set true due to clause 21. Now, clauses $22, 23$ imply—since at least one of p, q is true—that k has to be set false. Hence, as a consequence of clause 24 and the fact that both k, c are set false, ℓ has to be set true. That is, all of ℓ, j, f are set true, in contradiction with clause 25. Therewith, there in no satisfying truth assignment for the above formula in which z is set false.

On the other hand, it is not hard to verify that the formula is satisfiable; e.g., setting all variables of the set $\{z, g, a, r, e, p, k\}$ true and the remaining ones false yields a satisfying truth assignment.

Observe that a positive 2-clause can be replaced by a finite set of monotone 3-clauses analogously: By negating every variable appearance in $\mathcal{C}_z$, we can force z to be set to false. Therefore, we can get rid of clauses of the form $\{x, y\}$ analogously. Finally, note that z appears exactly 3 times, while none of the other variables is contained in more than four clauses.
Therewith, starting with an instance of Planar Monotone (2, 3)-Sat-3, we can create an equisatisfiable instance of Monotone 3-Sat-4 in polynomial time by adding a polynomial number of clauses and variables. □

In addition, we can prove that Monotone 3-Sat remains $\mathcal{NP}$-complete even if each variable appears *exactly* four times.

Corollary 4. Monotone 3-Sat-E4 *is $\mathcal{NP}$-complete.*

Proof. By Theorem 3, Monotone 3-Sat-4 is $\mathcal{NP}$-complete. Given an instance $\mathcal{I} = (\mathcal{V}, \mathcal{C})$ of Monotone 3-Sat-4, let $x \in \mathcal{V}$ appear less than four times. By adding all clauses of $\mathcal{C}^*$, where

$$\mathcal{C}^* = \{\{x, u, v\}, \{u, a, b\}, \{u, a, c\}, \{u, b, c\}, \{v, a, b\}, \{v, a, c\}, \{v, b, c\}\},$$

we get an instance $\mathcal{I}' = (\mathcal{V} \cup \{u, v, a, b, c\}, \mathcal{C} \cup \mathcal{C}^*)$ in which the number of appearances of x is increased by one and the newly introduced variables u, v, a, b, c appear exactly four times. Clearly, all clauses in C^* can be satisfied by setting true the newly introduced variables. Thus, if there is a satisfying truth assignment in instance $\mathcal{I}$, there must be one in $\mathcal{I}'$ as well. On the other hand, if all clauses in instance $\mathcal{I}'$ are satisfiable, this must also hold for the strict subset $\mathcal{C}$ of these clauses that make up $\mathcal{I}$. Therewith, the instances $\mathcal{I}$ and $\mathcal{I}'$ are equisatisfiable. As a consequence, given an instance of Monotone 3-Sat-4, we can create an equisatisfiable instance of Monotone 3-Sat-E4 in polynomial time by adding a polynomial number of clauses and variables. □

Theorem 3 clearly implies $\mathcal{NP}$-hardness of Monotone 3-Sat-5. We remark that the latter can also be shown by a reduction from Monotone (2, 3)-Sat-4 by the use of suitable replacement rules to replace the monotone 2-clauses with monotone 3-clauses without increasing the number of appearances of the variables in the 2-clauses (see [DD16] for details).

5. Hardness of Planar Monotone Sat* with bounded variable appearances

Finally, we consider a variation of the Planar Monotone Sat problem. An interesting question is whether or not we can replace a monotone 2-clause in an instance of Planar Monotone (2, 3)-Sat with monotone 3-clauses such that the result is an equisatisfiable instance of this problem, i. e., the corresponding graph remains planar. Unfortunately, replacement rules in the spirit of the rule used by Li [Li97, p. 295] do not preserve planarity: Li's rule replaces a clause $\{x, y\}$ with clauses

$$\{x, y, u\}, \{x, y, v\}, \{x, y, w\}, \{\bar{u}, \bar{v}, \bar{w}\},$$

where u, v, w are new variables; a clause $\{\bar{x}, \bar{y}\}$ is handled analogously. The incidence graph of the replaced clause already contains $K_{3,3}$ as a minor, cf. Figure 3.

Of course, we can achieve the goal described above by allowing duplicates of a variable in a clause; the clauses are now multisets for this reason. As described in Section 2, we denote this variation of Sat in which clauses are multisets of variables by Sat*. With this relaxation we show that the problem is $\mathcal{NP}$-complete already for the case that each variable appears exactly 4 times.

Since we use multisets instead of sets for the clauses, we slightly modify the definition of the incidence graph. Intuitively, we want the vertex degree to reflect the number of times a variable appears in the formula and the number of literals

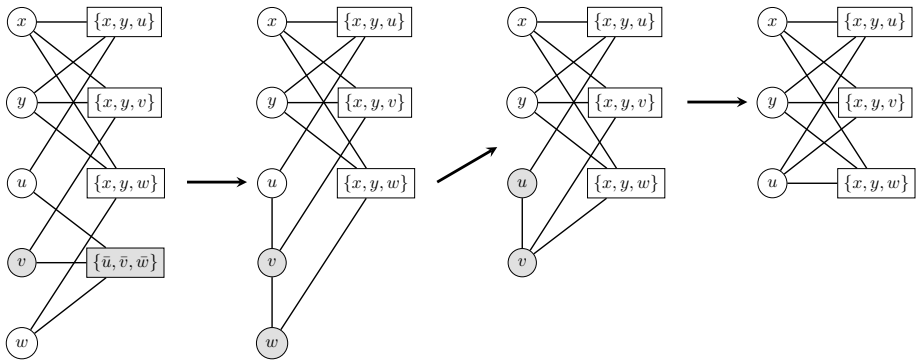


Figure 3. Replacement rules based on Li's rule do not preserve planarity, since the incidence graph of a replaced 2-clause $\{x, y\}$ has $K_{3,3}$ as a minor.

in a clause, respectively. Thus, if a variable v appears d_v^C times in a clause C, there are d_v^C edges between the corresponding vertices. Hence, the incidence graph is a multigraph without loops.

Theorem 5. Planar Monotone 3-Sat*-E4 *is $\mathcal{NP}$-complete.*

Proof. We give a reduction from Planar Monotone (2, 3)-Sat-E3 which is $\mathcal{NP}$-complete by Corollary 2. Given an instance $\mathcal{I} = (\mathcal{V}, \mathcal{C})$ of Planar Monotone (2, 3)-Sat-E3, construct instance $\mathcal{I}' = (\mathcal{V}', \mathcal{C}')$ of Planar Monotone 3-Sat*-E4 as follows. Replace each 2-clause $\{x, y\}$ of $\mathcal{C}$ by the two 3-clauses $\{x, y, z\}$, $\{\bar{z}, \bar{z}, \bar{z}\}$, where z is a new variable. Recall that each variable $x \in \mathcal{V}$ appears exactly three times in set $\mathcal{C}$. For each variable $x \in \mathcal{V}$, add the clauses $\{x, u, u\}$, $\{u, u, v\}$, $\{v, v, v\}$, where u, v are new variables.
Obviously, $\mathcal{I}$ is a "yes"-instance of Planar Monotone (2, 3)-Sat-E3 if and only if $\mathcal{I}'$ is a "yes"-instance of Planar Monotone 3-Sat*-E4. It is also easy to see that the graph $G_{\mathcal{V}',\mathcal{C}'}$ is planar. □

As noted in the work by Biedl and Kant [BK98, p. 172], their algorithm to produce orthogonal graph drawings also works for graphs with multiple edges. Let G be the incidence graph associated with an instance of Planar Monotone 3-Sat*-E4. Since all vertices of G have degree at most 4 and G has no loops, we can use the algorithm of Biedl and Kant to compute an orthogonal drawing of G.

Theorem 6. Planar Monotone 3-Sat*-E5 *is $\mathcal{NP}$-complete even if restricted to instances in which the graph $G_{\mathcal{V},\mathcal{C}}$ is biconnected.*

Proof. This time we use a reduction from 4-Bounded Planar 3-Connected 3-Sat [Kra94] in order to show $\mathcal{NP}$-hardness. In an instance $\mathcal{I} := (\mathcal{V}, \mathcal{C})$ of this

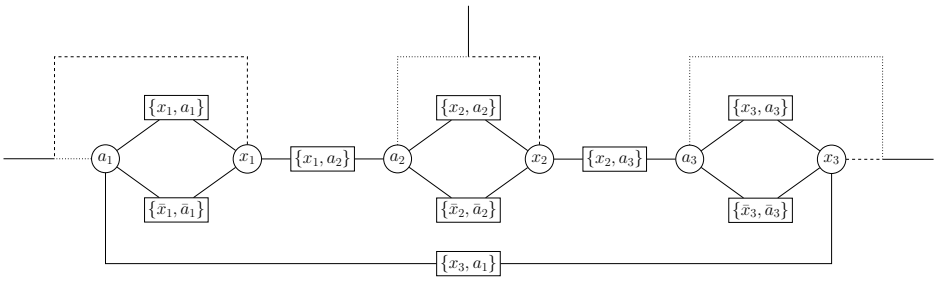


Figure 4. Gadget for vertices of degree 3.

variant, every variable appears at least three and at most four times and every clause contains exactly three distinct literals. Further, the incidence graph is planar and vertex 3-connected[d]. As in Theorem 1, we may assume an orthogonal drawing to be given—otherwise we can compute it in linear time—and replace every variable vertex locally. In Figure 4 the construction for a vertex of degree 3 is given; this gadget works in the same way as the one shown in Figure 2 (the additional 2-clause between a_i and x_i enforces opposite truth values for a_i and x_i also locally). The adaptation of the gadget for a vertex of degree 4 is straightforward (see Figure 5). Every clause of the original instance $\mathcal{I}$ contains exactly three literals, and is monotone after the local replacements. Further, every variable vertex of $\mathcal{I}$ is replaced. Thus, by showing that after using a gadget $\mathcal{G}$ for a local replacement, we can duplicate literals in the clauses of $\mathcal{G}$ so that every clause of $\mathcal{G}$ contains exactly three literals and every variable of $\mathcal{G}$ appears exactly 5 times in the final instance, these properties follow globally for the final instance. Recall that duplication of a literal within a clause is now possible. Now, the reasoning is identical for both gadgets: We consider the ring structure of a gadget in the clockwise direction. For a_i and x_i recall that depending on the instance either a_i or x_i has degree 3 and the other one has degree 4. We always duplicate x_i in the clause on the right of the corresponding variable vertex (on the bottom of Figure 5 this clause is drawn on the left). For the two clauses drawn between a_i and x_i we have two cases: If the degree of the vertex x_i is 5 after the just described duplication, we duplicate a_i in the upper clause and $\bar{a}_i$ in the lower clause. Since in this case a_i must have had degree 3 before the duplication, it now has degree 5 as well. Otherwise both x_i and a_i have degree 4 and we duplicate, e. g., x_i in the upper clause and $\bar{a}_i$ in the lower clause. Again both variable vertices have now degree 5. Doing this for every pair (a_i, x_i) in the ring structure yields an equisatisfiable construction with the desired properties.

The resulting graph is biconnected since the graph in the instance $\mathcal{I}$ is vertex 3-connected and the gadget used for the local replacement in the construction of

[d] A connected graph $G = (V, E)$ is called vertex k-connected, if both $|V| > k$ and the graph remains connected whenever at most $(k-1)$ vertices are removed.

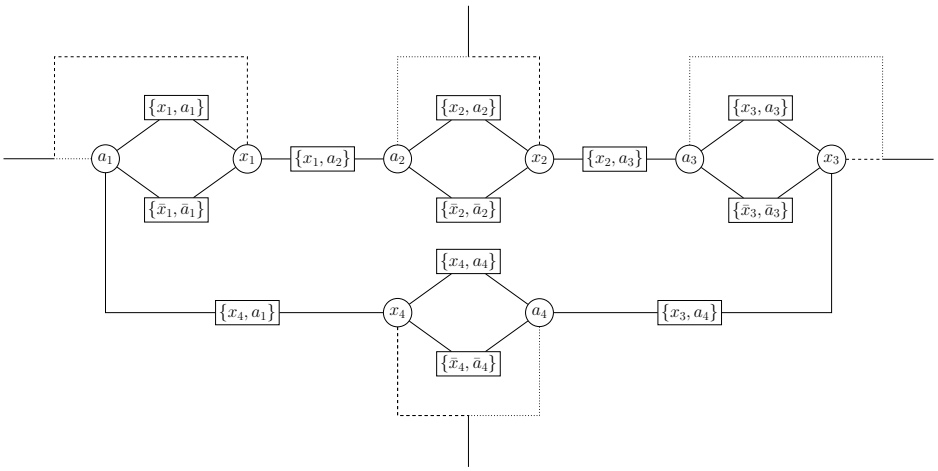


Figure 5. Gadget for vertices of degree 4.

the instance of Planar Monotone 3-Sat*-E5 is (obviously) biconnected. □

Biconnectedness in Planar Monotone Sat

In our $\mathcal{NP}$-completeness results for Planar Monotone Sat given in Section 3, we may assume the corresponding bipartite graph to be biconnected: Consider an instance of 4-Bounded Planar 3-Connected 3-Sat [Kra94] as defined in the proof of Theorem 6. By replacing every variable vertex with the gadget described by Dahlhaus et al. [DJP+94, p. 18] to reduce variable appearances (the gadget for the special cases for vertices of degree 3 and 4 is shown in Figure 6), we get a biconnected version of Planar $(2,3)$-Sat-E3, where for each variable, one of its literals appears in two clauses and the other literal appears in one clause.

Now, we can replace every variable vertex as shown in Figure 7. Note that the clause depicted in the center is always satisfied, since every variable appears as a positive and as a negative literal in the original instance. This approach yields a biconnected version of Corollary 2 and also Theorem 1 (recall that Theorem 1 (and Corollary 2) are proven by a reduction from Planar $(2,3)$-Sat-E3).

6. Conclusion

With respect to bounds on the number of times a variable may appear in the formula, we have provided $\mathcal{NP}$-completeness results for the monotone satisfiability problem with clauses containing (at most) three literals and some of its planar variants.

In particular, we have shown that Planar Monotone Sat is $\mathcal{NP}$-complete even if restricted to instances in which each clause consists of two or three dis-

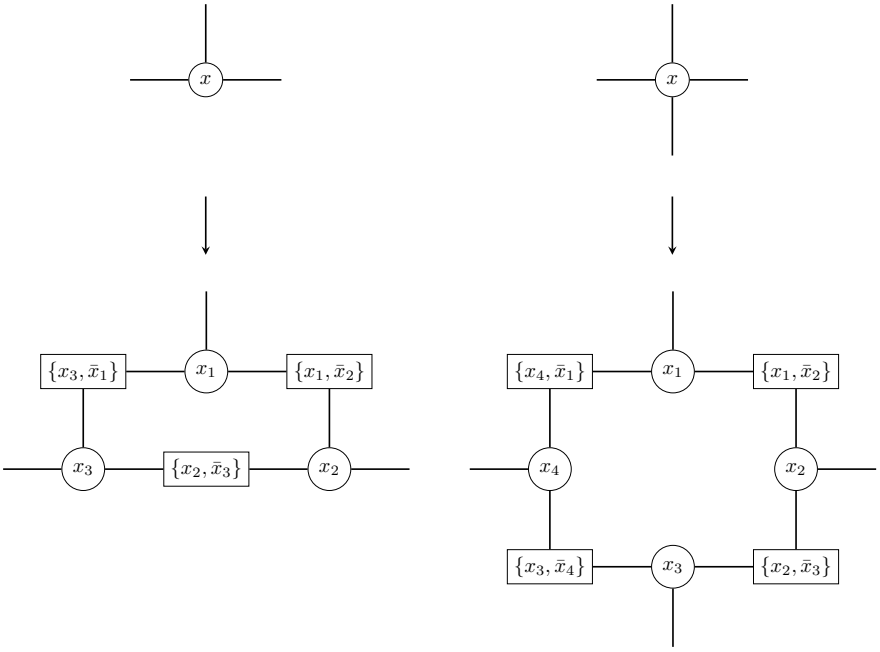


Figure 6. Reducing variable appearances with the gadget of Dahlhaus et al. [DJP+94, p. 18] restricted to vertices of degree 3 (left) and 4 (right).

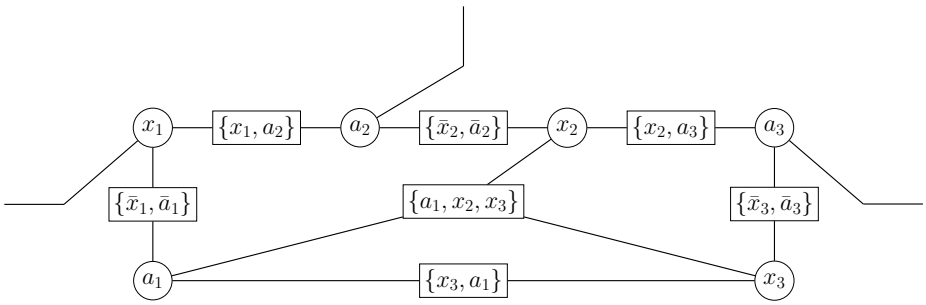


Figure 7. Local replacement of a variable vertex x with exactly three appearances. Here, the first appearance of x (corresponding to x_1) in the original instance is positive and the other appearances are negative, respectively. The variable vertices not directly connected to the original instance appear in an additional clause shown in the center.

tinct literals and each variable appears exactly three times. For Monotone 3-Sat, we could show that the problem is $\mathcal{NP}$-complete even if restricted to instances in

which each variable appears exactly four times, therewith confirming a conjecture made in scribe notes [DKY14] of an MIT lecture. Therewith, we have settled the computational complexity status of Planar Monotone Sat with at most three distinct literals per clause and of Monotone Sat with exactly three distinct literals per clause with respect to the number of variable appearances, since each of the problems is solvable in polynomial time when the number of variable appearances is reduced (see [Tov84]).

Concerning the two further variants of Planar Monotone Sat considered in Section 5, note that in the hardness proof of Theorem 1, the instance has an incidence graph with vertex degree bounded by four. Further, we have shown that these variants remain hard even if restricted to instances in which every variable appears exactly as often in the set of clauses as the upper bound for the variables allows for. Thus, planar and orthogonal drawings of these graphs exist and can be computed efficiently (see the work by Biedl and Kant [BK98]), which may be useful when using these variants as a starting point for a reduction in an $\mathcal{NP}$-hardness proof. Moreover, the variant proven to be $\mathcal{NP}$-complete in Theorem 5 has the property that each clause contains exactly three, not necessarily distinct, literals and every variable appears exactly four times in the set of clauses. Finally, the variant shown to be $\mathcal{NP}$-complete in Theorem 6 differs from the one considered in Theorem 5 in the way that every variable is required to appear exactly five times and the corresponding planar graph is biconnected.

An interesting open question remains: If we require every clause to contain exactly three *distinct* literals, is there a number $s \in \mathbb{N}$ such that Planar Monotone 3-Sat-s is $\mathcal{NP}$-hard and if so, what is the smallest number with this property (clearly, $s \geq 4$)?

Acknowledgement. The authors are grateful for the reviewers' valuable comments that improved the paper.

References

[BK98] T. Biedl and G. Kant. A better heuristic for orthogonal graph drawings. *Computational Geometry*, 9(3):159–180, 1998.

[BM08] J.A. Bondy and U.S.R. Murty. *Graph Theory*. Springer, 2008.

[dBK12] M. de Berg and A. Khosravi. Optimal binary space partitions for segments in the plane. *International Journal of Computational Geometry and Applications*, 22(3):187–206, 2012.

[DD16] A. Darmann and J. Döcker. Monotone 3-Sat-4 is NP-complete. *CoRR*, abs/1603.07881, 2016.

[DDD16] A. Darmann, J. Döcker, and B. Dorn. On planar variants of the monotone satisfiability problem with bounded variable appearances. *CoRR*, abs/1604.05588, 2016.

[DJP+94] E. Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour, and

M. Yannakakis. The complexity of multiterminal cuts. *SIAM Journal on Computing*, 23:864–894, 1994.

[DKY14] E. Demaine, J. Ku, and Y.W. Yu. Class 4 scribe notes. http://courses.csail.mit.edu/6.890/fall14/scribe/lec4.pdf; Instructor: E. Demaine; Notetakers: J. Ku, Y.W. Yu, 2014.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[Gol78] E.M. Gold. Complexity of automaton identification from given data. *Information and Control*, 37(3):302–320, 1978.

[KR92] D.E. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM Journal on Discrete Mathematics*, 5(3):422–427, 1992.

[Kra94] J. Kratochvíl. A special planar satisfiability problem and a consequence of its NP-completeness. *Discrete Applied Mathematics*, 52(3):233–252, 1994.

[Li97] W.N. Li. Two-segmented channel routing is strong NP-complete. *Discrete Applied Mathematics*, 78(1), 1997.

[Lic82] D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11:329–343, 1982.

[Tov84] C.A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85–89, 1984.

1.6 On simplified NP-complete variants of Monotone 3-SAT

The following paper [DD21] is also available online at the following URL: `https://doi.org/10.1016/j.dam.2020.12.010`.

Discrete Applied Mathematics 292 (2021) 45–58

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

On simplified NP-complete variants of Monotone 3-Sat

Andreas Darmann [a], Janosch Döcker [b,*]

[a] Department of Operations and Information Systems, University of Graz, Austria
[b] Department of Computer Science, University of Tübingen, Germany

ARTICLE INFO

Article history:
Received 14 August 2019
Received in revised form 15 October 2020
Accepted 4 December 2020
Available online xxxx

ABSTRACT

We consider simplified versions of 3-Sat, the variant of the famous Satisfiability Problem where each clause is made up of exactly three distinct literals formed over pairwise distinct variables. More precisely, the focus of this work is laid on Monotone 3-Sat, the restriction of 3-Sat to formulas with monotone clauses, where a clause is monotone if it contains only unnegated variables or only negated variables. We prove several hardness results for Monotone 3-Sat with respect to a variety of restrictions imposed on the variable appearances.

In particular, we show that for any $k \geq 2$, Monotone 3-Sat turns out to be NP-complete even if each variable appears exactly k times unnegated and exactly k times negated. Therewith, for Monotone 3-Sat with balanced variable appearances we establish a sharp boundary between NP-complete and polynomial time solvable cases.

In addition, we prove that for any $k \geq 5$, Monotone 3-Sat is NP-complete even if each variable appears exactly k times unnegated and exactly once negated. Further, we prove that the problem remains NP-complete when restricted to instances in which each variable appears either exactly once unnegated and three times negated or the other way around. Thereby, we improve on a result by Darmann et al. (2018) showing NP-completeness for four appearances per variable. Our stronger result also implies that 3-Sat remains NP-complete even if each variable appears exactly three times unnegated and once negated, therewith complementing a result by Berman et al. (2003).

1. Introduction

The famous Boolean satisfiability problem, and in particular 3-Satisfiability, can be considered *the* classical decision problem in computer science. 3-Satisfiability has been the first problem shown to be NP-complete decades ago (Cook [3]) and is of undisputed theoretical and practical importance[1]; it both appears in practical applications of routing, scheduling and artificial intelligence (see, e.g., Devlin and O'Sullivan [7], Nam et al. [17], Horbach et al. [12], and Kautz and Selman [13]), and is the most prominent problem, and probably the most frequently used one, for complexity analysis of decision problems. Therefore, it has continuously attracted researchers through decades focusing on the computational complexity of variants of the satisfiability problem (for recent work see, e.g., Pilz [19] or Paulusma and Szeider [18]).

* Corresponding author.
E-mail addresses: andreas.darmann@uni-graz.at (A. Darmann), janosch.doecker@uni-tuebingen.de (J. Döcker).

[1] This is also witnessed by the dedicated series of International Conferences on Theory and Applications of Satisfiability Testing (http://www.satisfiability.org/), focusing on theoretical and practical research in connection with the satisfiability problem.

https://doi.org/10.1016/j.dam.2020.12.010
0166-218X/

In this paper, we add to that branch of literature and investigate the computational complexity[2] of restricted variants of 3-Satisfiability.

In 3-Satisfiability, we are given a set of propositional variables and a collection of clauses, where each clause contains three literals. The question is whether there is a satisfying truth assignment, i.e., whether we can satisfy all clauses by assigning truth values to the variables.

In what follows, we will refer to 3-Sat as the variant of 3-Satisfiability in which each clause contains exactly three *distinct* literals formed over pairwise distinct variables—which is the setting we focus on in this paper.

The focus of this paper is laid on Monotone 3-Sat, the restriction of 3-Sat to formulas in which each clause is monotone, i.e., contains only unnegated or only negated variables. It is known that Monotone 3-Sat is NP-complete [11,15], and that intractability holds even if (1) each variable appears exactly four times [6, Corollary 4]. We show that this problem remains NP-complete even if condition (1) is replaced by either one of the following three conditions:

(1a) each variable appears exactly p times unnegated and q times negated, for every fixed pair (p, q) with $p \geq 2$ and $q \geq 2$,
(1b) each variable appears exactly k times unnegated and once negated, for every fixed integer $k \geq 5$, or
(1c) each variable appears exactly three times unnegated and once negated or three times negated and once unnegated.

We point out that (1a) includes the case of balanced variable appearances, i.e, the case of each variable appearing exactly k times unnegated and k times negated, for each $k \geq 2$. Further, we remark that the hardness results for conditions (1a) and (1c) improve upon the result for condition (1) by Darmann et al. [6, Corollary 4]. Also, as a by-product, we derive the result that the classical 3-Sat problem remains NP-complete even if each variable appears exactly three times unnegated and once negated (observe that this implies hardness also for the vice versa case where each variable appears exactly once unnegated and three times negated). Therewith, we complement results of Tovey [21] and Berman et al. [2]: The former showed that 3-Sat remains NP-complete even if each variable appears in at most four clauses and it is trivial if the number of variable appearances is bounded by 3 [21, Theorem 2.3 and Theorem 2.4]; Berman et al. [2, Theorem 1] added to that result by showing that NP-completeness holds even if each variable appears exactly twice unnegated and twice negated.

Further related literature is concerned with the planar[3] variants of (Monotone) 3-Satisfiability. Both Planar 3-Satisfiability and Planar Monotone 3-Satisfiability are known to be NP-complete even in restricted settings (e.g., see [14,16] respectively [1,6]), while Pilz [19, Theorem 11] shows that all instances of Planar Monotone 3-Sat, i.e., where each clause contains three distinct literals formed over pairwise distinct variables, are satisfiable.

The paper is structured as follows. In Section 2 we introduce basic notation and formally state the considered decision problems. In Section 3 we present a tool for increasing the number of literal appearances in an instance of Monotone 3-Sat without affecting satisfiability. In Section 4 we provide hardness results for Monotone 3-Sat in the restricted setting of balanced variable appearances, where each variable appears unnegated and negated equally often. In fact, we show that Monotone 3-Sat is NP-complete if each variable appears exactly p times unnegated and q times negated, for every fixed pair (p, q) with $p \geq 2$ and $q \geq 2$. In Section 5 Monotone 3-Sat is analyzed restricted to instances in which each variable appears exactly once negated. Then we consider Monotone 3-Sat restricted to instances in which each variable appears either three times unnegated and once negated or once unnegated and three times negated in Section 6. Finally, Section 7 concludes the paper with a concise summary of the results and a challenge for future research.

Preliminary versions of the results in this paper are contained in the online preprints [4,8] on arXiv.

2. Preliminaries

Let $V = \{x_1, \ldots, x_n\}$ be a set of propositional *variables*. For the remainder of the paper we simply say variable instead of propositional variable since all variables take on values in $\{T, F\}$, where T represents true and F represents false. A *literal* is a variable or its negation, i.e., an element of $L_V = \{x_i, \overline{x_i} \mid x_i \in V\}$. A *clause* is a subset of L_V, and a k-clause contains exactly k distinct literals. Further, a clause is *monotone* if either all contained variables are negated (*negative clause*) or none of them is (*positive clause*). A Boolean formula C in *conjunctive normal form* (CNF) is a collection of m clauses, i.e., $C = \bigcup_{j=1}^{m} \{c_j\}$. We say that a Boolean formula C is in k-CNF if each clause in C is a k-clause. It is also common to use logical connectives, e.g. $\vee$ and $\wedge$, to describe a Boolean formula. Then, C is a conjunction of clauses $\bigwedge_{j=1}^{m} c_j$, where $c_j = (\ell_{j,1} \vee \ell_{j,2} \vee \ldots \vee \ell_{j,i_j})$ is a disjunction of pairwise distinct literals; in what follows, we use the set notation $c_j = \{\ell_{j,1}, \ell_{j,2}, \ldots, \ell_{j,i_j}\}$ for describing clauses. A Boolean formula is *linear* if all pairs of distinct clauses share at most one variable. A *truth assignment* is a mapping $\beta: V \to \{T, F\}$ which extends to literals in the obvious way, i.e., for $\ell = x_i$ we have $\beta(\ell) = \beta(x_i)$ and for $\ell = \overline{x_i}$ we have $\beta(\ell) \in \{T, F\} \setminus \beta(x_i)$, $i \in \{1, 2, \ldots, n\}$. A clause c_j is *satisfied* under β if $\beta(\ell) = T$ for at least one $\ell \in c_j$. A Boolean formula $C = \bigcup_{j=1}^{m} \{c_j\}$ in CNF is *satisfiable* if there exists a truth assignment $\beta: V \to \{T, F\}$ such that all clauses $c_1, \ldots, c_m$

[2] We assume the reader to be familiar with the basic concepts of the theory of NP-completeness and refer to Garey and Johnson [9] for an extensive introduction.

[3] In that respect, planarity refers to the corresponding graph property of the following associated bipartite graph: there is a vertex for each variable v and for each clause c, and an edge connects a variable vertex v with a clause vertex c if and only if variable v appears in clause c.

are satisfied. We say that a truth assignment β' for V' *extends* a truth assignment β for V if $V \subseteq V'$ and $\beta'(v) = \beta(v)$ for all $v \in V$. Finally, a Boolean formula C in CNF is *nae-satisfiable* if there exists a truth assignment for the variables such that each clause has at least one true and at least one false literal.

2.1. Problem statements

The decision problems considered in this work are variants of the classical decision problem 3-SAT restricted to subsets of instances, where clauses, variables or both are required to have certain properties (e.g. that each clause is monotone).

3-SAT
Instance. A set V of variables, and a collection C of clauses over V such that each clause $c \in C$ contains $|c| = 3$ distinct literals formed over pairwise distinct variables.
Question. Is there a truth assignment for V such that each clause in C has at least one true literal?

Now, we present the variants of 3-SAT that we focus on in this article.

3-SAT-E4 is the restriction of 3-SAT to instances in which each variable appears in exactly four clauses.

MONOTONE 3-SAT-E4 is the restriction of 3-SAT to instances in which each clause is monotone, and every variable appears in exactly four clauses.

MONOTONE 3-SAT-(p, q) is the restriction of 3-SAT to instances in which each clause is monotone, and every variable appears unnegated in exactly p clauses and negated in exactly q clauses.

The following relaxed definition of MONOTONE 3-SAT-(p, q) is particularly useful in establishing hardness results, since it is equivalent in terms of NP-completeness (see also Section 3) and allows us to focus on the crucial parts of the constructions (i.e., we do not have to blow up the formula by adding clauses in order to get the required number of literal appearances).

MONOTONE 3-SAT-$(\leq p, \leq q)$ is the restriction of 3-SAT to instances in which each clause is monotone, and every variable appears unnegated in at most p clauses and negated in at most q clauses.

Note that all of the above decision problems belong to the class NP. Hence, the NP-completeness proofs in this paper reduce to showing NP-hardness of the respective problem.

3. Increasing the number of literal appearances in Monotone 3-SAT

We thank an anonymous referee for providing the following two lemmata (along with the below proof and a description) which are very helpful for improving the presentation of the paper. The lemmata establish a tool for increasing the number of literal appearances in any instance of MONOTONE 3-SAT without affecting satisfiability, therefore, e.g., implying that MONOTONE 3-SAT-(p, q) and MONOTONE 3-SAT-$(\leq p, \leq q)$ are equivalent in terms of NP-completeness.

Lemma 1. *For every $k \geq 1$, $\ell \geq 1$, and $r \geq 3$, such that $k \leq r^2$ and $\ell \leq r^2$, there is a 3-CNF formula $C_{k,\ell,r}$ with variables $\{y_1, y_2, y_3\} \cup \mathcal{U} \cup \mathcal{V} \cup \mathcal{W}$ where $\mathcal{U} = \{u_0, \ldots, u_{r-1}\}$, $\mathcal{V} = \{v_0, \ldots, v_{r-1}\}$, and $\mathcal{W} = \{w_0, \ldots, w_{r-1}\}$, such that*

- *each of the clauses is monotone,*
- *each of the variables y_1, y_2, y_3 has one unnegated and no negated appearance,*
- *each of the variables in $\mathcal{U} \cup \mathcal{V} \cup \mathcal{W}$ has k unnegated and ℓ negated appearances,*
- *$C_{k,\ell,r}$ is satisfied by setting the variables in $\mathcal{U}$ to false and the variables in $\mathcal{V}$ and $\mathcal{W}$ to true independently of the truth values of the variables y_1, y_2, y_3.*

Proof. Let $\mathbb{Z}_r := \{0, 1, \ldots, r-1\}$. For every pair $(s, t) \in \mathbb{Z}_r \times \mathbb{Z}_r$, let

$$\theta_{s,t}(\mathcal{U}, \mathcal{V}, \mathcal{W}) = \{\{u_i, v_{i+s}, w_{i+t}\} \mid i \in \mathbb{Z}_r\}$$

and let

$$\theta_{s,t}(\overline{\mathcal{U}}, \overline{\mathcal{V}}, \overline{\mathcal{W}}) = \{\{\overline{u_i}, \overline{v_{i+s}}, \overline{w_{i+t}}\} \mid i \in \mathbb{Z}_r\},$$

where indices are added modulo r, i.e., for $r \leq j \leq 2r-2$ variables v_j and w_j correspond to v_{j-r} and w_{j-r} respectively. Moreover, let $\theta'_{0,0}$ be the formula obtained from $\theta_{0,0}(\mathcal{U}, \mathcal{V}, \mathcal{W})$ by replacing the three clauses

$$\{u_0, v_0, w_0\}, \{u_1, v_1, w_1\}, \{u_2, v_2, w_2\}$$

with the four clauses

$$\{u_0, v_0, y_1\}, \{u_1, v_1, y_2\}, \{u_2, v_2, y_3\}, \{w_0, w_1, w_2\}.$$

Let I be an arbitrary subset of $\mathbb{Z}_r \times \mathbb{Z}_r$ of size $k-1$ not containing $(0,0)$ and let J be an arbitrary subset of $\mathbb{Z}_r \times \mathbb{Z}_r$ of size ℓ. Then, the formula

$$C_{k,\ell,r} = \theta'_{0,0} \cup \bigcup_{(s,t)\in I} \theta_{s,t}(\mathcal{U}, \mathcal{V}, \mathcal{W}) \cup \bigcup_{(s,t)\in J} \theta_{s,t}(\overline{\mathcal{U}}, \overline{\mathcal{V}}, \overline{\mathcal{W}})$$

satisfies the requirements of the statement. □

By negating all the variables in $C_{\ell,k,r}$, one can obtain also the following.

Lemma 2. *There is a formula $\overline{C_{k,\ell,r}}$ with the same properties as $C_{k,\ell,r}$ except that the variables y_1, y_2, y_3 have negated appearances and the guaranteed satisfying assignment sets all variables in $\mathcal{U}$ to true and all variables in $\mathcal{V}$ and $\mathcal{W}$ to false.*

In order to increase the number of literal appearances in an instance of Monotone 3-Sat without affecting satisfiability the above formulas $C_{k,\ell,r}$ and $\overline{C_{k,\ell,r}}$ can be used as follows. Consider an instance of Monotone 3-Sat-($\leq p, \leq q$) consisting of a formula C over variables $V_C = \{x_1, \ldots, x_n\}$ which we would like to translate into an instance of Monotone 3-Sat-(p, q). For each variable $x_i \in V_C$, let p_i (resp. q_i) denote the number of unnegated (resp. negated) appearances of x_i. The goal is, for each variable x_i, to increase the number of unnegated (resp. negated) appearances by $\Delta_i^+ := p - p_i$ (resp. $\Delta_i^- := q - q_i$). To this end, we construct three copies C_j of C, where variable x_i is replaced with $x_{i,j}$, $j = 1, 2, 3$. Further, let $r \geq 3$, such that $p \leq r^2$ and $q \leq r^2$. Now, for each $i = 1, \ldots, n$, we introduce Δ_i^+ copies of $C_{p,q,r}$, where in the latter y_j is replaced with $x_{i,j}$, $j = 1, 2, 3$, and in each copy of $C_{p,q,r}$ the sets $\mathcal{U}, \mathcal{V}, \mathcal{W}$ consist of newly introduced variables. Analogously, for each $i = 1, \ldots, n$, we introduce Δ_i^- copies of $\overline{C_{p,q,r}}$, where $\overline{y_j}$ is replaced with $\overline{x_{i,j}}$, $j = 1, 2, 3$. We denote the resulting formula $C_1 \cup C_2 \cup C_3$ together with the introduced copies of $C_{p,q,r}$ and $\overline{C_{p,q,r}}$ by $\tilde{C}$. In the formula $\tilde{C}$, each variable $x_{i,j}$ – as well as each of the newly introduced auxiliary variables – appears exactly p times unnegated and q times negated. By Lemmas 1 and 2, the formula $\tilde{C}$ is satisfiable if and only if the original formula C is satisfiable. Finally, we have that $\tilde{C}$ contains $3m + \Delta(1 + pr + qr)$ clauses over $3n + \Delta \cdot 3r$ variables, where m is the number of clauses contained in C and $\Delta = \sum_{i=1}^{n}(\Delta_i^+ + \Delta_i^-)$ is the total number of introduced copies of $C_{p,q,r}$ and $\overline{C_{p,q,r}}$. Since the total number of introduced copies of $C_{p,q,r}$ and $\overline{C_{p,q,r}}$ is at most $p + q$ per variable, it follows that the number of clauses in $\tilde{C}$ is $3m + \mathcal{O}(n)$, and the number of variables is $\mathcal{O}(n)$. Hence, the size of the constructed instance is polynomial in the size of the original instance and, thus, the transformation is polynomial.

Hence, we can polynomially transform an instance of Monotone 3-Sat-($\leq p, \leq q$) into an instance of Monotone 3-Sat-(p, q) without affecting satisfiability. Therefore, we can state the following theorem.

Theorem 1. *Let p, q be fixed integers with $p \geq 1$ and $q \geq 1$. Then,* Monotone 3-Sat-($\leq p, \leq q$) *is NP-complete if and only if* Monotone 3-Sat-(p, q) *is NP-complete.*

Observe that Theorem 1 implies the following corollary.

Corollary 1. *Let r, s be fixed positive integers such that* Monotone 3-Sat-($\leq r, \leq s$) *is NP-complete. Then,* Monotone 3-Sat-(p, q) *is NP-complete for all pairs of fixed integers p, q with $p \geq r$ and $q \geq s$.*

4. Monotone 3-Sat with balanced variable appearances

In this section we consider the case of *balanced* variable appearances, where each variable appears unnegated and negated equally often; the section is structured as follows.

Section 4.1 is dedicated to Monotone 3-Sat-(k, k) for $k \geq 3$. There, we begin with a simple corollary stating NP-completeness of Monotone 3-Sat-(4, 4), even in a restricted setting. Then we turn to Monotone 3-Sat-(3, 3) and, by the use of several lemmata, show its NP-completeness. Along the way, we in fact show that Monotone 3-Sat-($\leq 3, \leq 2$) is NP-complete.

Finally, in Section 4.2 we turn our attention to instances in which each variable appears exactly twice unnegated and exactly twice negated. There, we first construct an unsatisfiable instance of Monotone 3-Sat-(2, 2) and, with the help of the latter, prove that the problem is NP-complete. Thereby, we in fact establish the more general result that Monotone 3-Sat-(p, q) is NP-complete for each fixed pair $(p, q) \in \{(r, s), (s, r) \mid r \geq 2, s \geq 2\}$.

4.1. Monotone 3-Sat-(k, k), for $k \geq 3$

Corollary 2. Monotone 3-Sat-(4, 4) *is NP-complete, even if no pair of clauses has exactly two variables and more than one literal in common.*

Proof. The proof proceeds by a reduction from Monotone Not-All-Equal-3-Sat-E4. In Not-All-Equal 3-Sat we are given a 3-Sat formula and ask whether the formula is nae-satisfiable. An instance of Not-All-Equal 3-Sat is *monotone*[4] if and

[4] We point out that *monotonicity* has different meanings for 3-Satisfiability and Not-All-Equal 3-Satisfiability, which is certainly not ideal but appears to be the established notation.

only if negations are completely absent, i.e., there are no negated variables in the formula. Darmann and Döcker [5] have shown that Monotone Not-All-Equal 3-Sat-E4 is NP-complete even for linear instances, i.e., instances where each pair of distinct clauses shares at most one variable.

Now, NP-hardness follows from the simple standard transformation from Not-All-Equal 3-Sat to 3-Sat: Given an instance of Monotone Not-All-Equal-3-Sat-E4 where the formula is linear, introduce for each clause $\{\ell_1, \ell_2, \ell_3\}$ a second clause $\{\overline{\ell_1}, \overline{\ell_2}, \overline{\ell_3}\}$. Note that the resulting formula has the desired properties. □

In the next step, we consider Monotone 3-Sat-(3, 3). First, we show that not all instances of this problem are satisfiable. In particular, we present an unsatisfiable instance of Monotone 3-Sat-(3, 3) with 7 variables in the following proposition. Since the first 7 clauses, which are adapted from [8], are not nae-satisfiable by a result from Porschen et al. [20, Corollary 4], unsatisfiability follows by construction (note that clauses 8 to 14 are obtained from clauses 1 to 7 by replacing each variable with its negation).

Proposition 1. *The following instance of* Monotone 3-Sat-(3, 3) *with 7 variables and 14 clauses is unsatisfiable.*

1. $\{a, b, g\}$
2. $\{a, c, f\}$
3. $\{a, d, e\}$
4. $\{b, c, d\}$
5. $\{b, e, f\}$
6. $\{c, e, g\}$
7. $\{d, f, g\}$
8. $\{\bar{a}, \bar{b}, \bar{g}\}$
9. $\{\bar{a}, \bar{c}, \bar{f}\}$
10. $\{\bar{a}, \bar{d}, \bar{e}\}$
11. $\{\bar{b}, \bar{c}, \bar{d}\}$
12. $\{\bar{b}, \bar{e}, \bar{f}\}$
13. $\{\bar{c}, \bar{e}, \bar{g}\}$
14. $\{\bar{d}, \bar{f}, \bar{g}\}$

In fact, it turns out that Monotone 3-Sat-(3, 3) is NP-complete. This is a consequence of the stronger statement proven below that Monotone 3-Sat-($\leq 3, \leq 2$) is NP-complete (stated in Theorem 2), which implies NP-completeness of Monotone 3-Sat-(3, 3) by Corollary 1.

To this end, we make use of a construction inspired by the idea of an *enforcer* for a clause described by Berman et al. [2, p. 3]. Intuitively speaking, an enforcer – we say *gadget* instead of enforcer in the following – is a set of clauses, usually having some desired properties, that imposes certain restrictions on satisfying truth assignments (e.g. simulate the behavior of a clause). For instance, the gadget introduced in the following lemma allows us to simulate a disjunction of three appearances of the same variable in the proof of Theorem 2, which is useful since we do not allow duplicates of a literal in a clause.

Lemma 3. *Let $\mathcal{G}(x, y, z)$ be the following set of clauses, where $V_{aux} = \{a, b, \ldots, f\}$ are new variables.*

1. $\{\bar{a}, \bar{b}, \bar{f}\}$
2. $\{\bar{a}, \bar{c}, \bar{d}\}$
3. $\{\bar{b}, \bar{c}, \bar{e}\}$
4. $\{\bar{d}, \bar{e}, \bar{f}\}$
5. $\{a, b, f\}$
6. $\{a, c, d\}$
7. $\{b, c, e\}$
8. $\{d, e, f\}$
9. $\{a, e, x\}$
10. $\{b, d, y\}$
11. $\{c, f, z\}$

Then, a truth assignment β for $\{x, y, z\}$ can be extended to a truth assignment β' for $\{x, y, z\} \cup V_{aux}$ that satisfies $\mathcal{G}(x, y, z)$ if and only if $\beta(v) = T$ for at least one $v \in \{x, y, z\}$.

Proof. First, let $\beta(v) = F$ for each $v \in \{x, y, z\}$. Then, we can simplify $\mathcal{G}(x, y, z)$ by removing the literals which evaluate to false from clauses 9–11. Assume towards a contradiction that β can be extended to a truth assignment β' that satisfies $\mathcal{G}(x, y, z)$. Note that β' cannot set both a and e to true since, in this case, clauses 1–4 are equivalent to the following implications.

1. $b \Rightarrow \bar{f}$
2. $d \Rightarrow \bar{c}$
3. $b \Rightarrow \bar{c}$
4. $d \Rightarrow \bar{f}$

Hence, clauses 1–4 in conjunction with clauses 10 and 11 is a contradiction. Analogously, setting both variables in clause 10 (resp. clause 11) to true yields a contradiction. Hence, we may assume that β' sets the variables in $\{a, e\}$ (resp. $\{b, d\}$ and $\{c, f\}$) to different truth values. It follows that the following set of clauses obtained from clauses 1–8 in $\mathcal{G}(x, y, z)$ by replacing e with $\bar{a}$, d with $\bar{b}$ and f with $\bar{c}$ is satisfiable.

(i) $\{\bar{a}, \bar{b}, c\}$
(ii) $\{\bar{a}, b, \bar{c}\}$
(iii) $\{a, \bar{b}, \bar{c}\}$
(iv) $\{a, b, c\}$
(v) $\{a, \underline{b}, \bar{c}\}$
(vi) $\{a, \bar{b}, c\}$
(vii) $\{\bar{a}, \underline{b}, c\}$
(viii) $\{\bar{a}, \bar{b}, \bar{c}\}$

The above eight clauses are clearly not satisfiable, which is in contradiction with our assumption that β' satisfies $\mathcal{G}(x, y, z)$. Thus, β cannot be extended to a truth assignment that satisfies $\mathcal{G}(x, y, z)$ if $\beta(v) = F$ for each $v \in \{x, y, z\}$.

Second, let β be a truth assignment for $\{x, y, z\}$ with $\beta(x) = T$, $\beta(y) = b_y$ and $\beta(z) = b_z$ where $b_y, b_z \in \{T, F\}$. Then, we extend β to a truth assignment β' that satisfies $\mathcal{G}(x, y, z)$ by setting $\beta'(a) = \beta'(e) = F$ and $\beta'(v) = T$ for all $v \in V_{\text{aux}} \setminus \{a, e\}$. It is easy to verify that $\mathcal{G}(x, y, z)$ is satisfied for this assignment even if $b_y = b_z = F$. By using the same approach, we can show that if $\beta(y) = T$ or $\beta(z) = T$, we can assign truth values to the remaining variables such that $\mathcal{G}(x, y, z)$ is satisfied. □

With the help of the gadget introduced in the lemma above we are able to define a gadget that, by Lemma 3, forces some variable y to true:

$$\mathcal{H}(y) = \mathcal{G}(y, y, y).$$

Now, we can also force some variable z to false by using the following gadget:

$$\mathcal{H}'(\overline{z}) = \{\{\overline{z}, \overline{u}, \overline{v}\}\} \cup \mathcal{G}(u, u, u) \cup \mathcal{G}(v, v, v).$$

Remark. Each instance of a gadget $\mathcal{H}(y)$ (resp. $\mathcal{H}'(\overline{z})$) has its own new auxiliary variables (i.e., the variables that are not in $\{y, z\}$). Further, each auxiliary variable introduced by $\mathcal{H}(y)$ (resp. $\mathcal{H}'(\overline{z})$) appears exactly three times unnegated and at most twice negated (u, v appear only once negated in $\mathcal{H}'(\overline{z})$). The variable y (resp. z) has exactly three unnegated appearances and no negated appearance (resp. no unnegated appearance and exactly one negated appearance).

We are now in a position to prove NP-completeness of Monotone 3-Sat-$(\leq 3, \leq 2)$.

Theorem 2. Monotone 3-Sat-$(\leq 3, \leq 2)$ *is NP-complete.*

Proof. By reduction from 3-Sat-(2, 2), for which NP-hardness was established by Berman et al. [2, Theorem 1]. Given an instance of the latter with a set V of variables and a set C of clauses over V, let $n := |V|$. For each variable $x_i \in V$, we introduce two new variables $x_{i,1}$, $x_{i,2}$ and replace the two negated appearances with $x_{i,1}$ and the two unnegated appearances with $x_{i,2}$, respectively. Then, we remove all negations and introduce the following clauses for $i \in \{1, 2, \ldots, n\}$:

$$\{\{\overline{x_{i,1}}, \overline{x_{i,2}}, \overline{y_i}\}, \{x_{i,1}, x_{i,2}, z_i\}\} \cup \mathcal{H}(y_i) \cup \mathcal{H}'(\overline{z_i}),$$

where y_i and z_i are new variables. Since these clauses can be satisfied if and only if we assign different truth values to $x_{i,1}$ and $x_{i,2}$, the resulting formula is satisfiable if and only if the original formula is satisfiable. By construction, all variables appear at most three times unnegated and twice negated. Hence, the constructed formula is an instance of Monotone 3-Sat-$(\leq 3, \leq 2)$. We conclude by remarking that the transformation is polynomial. □

Now, by Theorem 1 and Corollary 1, respectively, we get the following two corollaries, the latter of which will be used for our hardness result in the next section.

Corollary 3. Monotone 3-Sat-(3, 2) *is NP-complete.*

Corollary 4. Monotone 3-Sat-(3, 3) *is NP-complete.*

4.2. Monotone 3-Sat-(2, 2)

We now turn our attention to Monotone 3-Sat-(2, 2) and prove its NP-completeness. Therewith, we settle a sharp boundary in terms of the number of variable appearances between NP-complete and polynomial time solvable cases for balanced variable appearances, since Monotone 3-Sat-(1, 1) is trivial due to the classic result of Tovey [21] stating that 3-Sat is trivial if the number of variable appearances is bounded by 3.

We prove NP-completeness of Monotone 3-Sat-(2, 2) in two steps. In the first step (Section 4.2.1), we construct an unsatisfiable instance of Monotone 3-Sat-(2, 2). In the second step (Section 4.2.2), we make use of that instance and of Corollary 4 to formally prove our hardness result.

4.2.1. Construction of an unsatisfiable instance of Monotone 3-Sat-(2, 2)

In this section, we construct an unsatisfiable instance of Monotone 3-Sat-(2, 2). We start with the Boolean formula

$$C = \{\{\overline{a}, \overline{d}, \overline{f}\}, \{b, d, e\}, \{e, \overline{b}\}, \{d, \overline{f}, \overline{c}\}, \{a, \overline{c}, \overline{e}\}, \{\overline{e}, c\}, \{\overline{d}, a, b\}, \{\overline{a}, f\}\}$$

for which unsatisfiability is easy to verify. Note that the formula contains non-monotone clauses consisting of two or three distinct literals. In the following, we transform C into an unsatisfiable instance of Monotone 3-Sat-(2, 2) with the help of two families of gadgets (resp. enforcers[5]).

The first family $\mathcal{M}^{(i)}(u_1, \overline{u_2}, \overline{u_3})$ deals with clauses of the form $\{u_1, \overline{u_2}, \overline{u_3}\}$, i.e., clauses consisting of one unnegated variable and two negated variables. Moreover, this family can also be used to transform non-monotone 2-clauses (e.g., set $u_1 = e$ and $u_2 = u_3 = \overline{b}$ for the third clause in C), but this comes at the price of introducing an additional appearance of the negated variable, so it is not a viable option for the clauses $\{\overline{e}, c\}$, $\{\overline{a}, f\}$ as the literals $\overline{e}$ and $\overline{a}$ already appear twice in C (these clauses will be transformed with the help of the second family of gadgets). Both families are based on an unsatisfiable Boolean formula $\mathcal{M}$—which we construct in the following—containing one positive 2-clause, two negative 2-clauses and 39 monotone 3-clauses. Intuitively, the formula $\mathcal{M}$ consists of three collections of clauses $\mathcal{Q}$, $\mathcal{R}$ and $\mathcal{S}$ (the

[5] Recall that we use gadget and enforcer interchangeably to describe the same concept (see Section 4.1).

two latter were found via computer search[6]), where $\mathcal{Q}$ is only satisfiable by truth assignments, say β, for the corresponding variables that can be placed in one of two categories—a truth assignment of the first (resp. second) category cannot be extended to satisfy $\mathcal{Q} \cup \mathcal{R}$ (resp. $\mathcal{Q} \cup \mathcal{S}$). We start the construction of $\mathcal{M}$ with $\mathcal{Q} = \mathcal{Q}_2 \cup \mathcal{Q}_3$. First, let $\mathcal{Q}_2$ denote the set consisting of the following 2-clauses:

1. $\{x_1, x_2\}$
2. $\{\overline{x_2}, \overline{x_3}\}$
3. $\{\overline{x_2}, \overline{x_4}\}$

Second, let $\mathcal{Q}_3$ denote the set consisting of the following 3-clauses:

4. $\{\overline{x_3}, \overline{x_5}, \overline{x_6}\}$
5. $\{\overline{x_4}, \overline{x_5}, \overline{x_6}\}$
6. $\{x_5, x_7, x_8\}$
7. $\{x_6, x_7, x_8\}$
8. $\{\overline{x_7}, \overline{z_1}, \overline{z_2}\}$
9. $\{\overline{x_7}, \overline{z_3}, \overline{z_4}\}$
10. $\{\overline{x_8}, \overline{z_1}, \overline{z_2}\}$
11. $\{\overline{x_8}, \overline{z_3}, \overline{z_4}\}$

Note that the 2-clauses in $\mathcal{Q}_2$ are equivalent to the implications

$$\overline{x_2} \Rightarrow x_1, \quad x_2 \Rightarrow \overline{x_3}, \quad x_2 \Rightarrow \overline{x_4}.$$

Next, let $\mathcal{R}$ be the set consisting of the following 3-clauses:

12. $\{x_3, y_1, y_2\}$
13. $\{x_3, y_3, y_4\}$
14. $\{x_4, y_5, y_6\}$
15. $\{x_4, y_7, y_8\}$
16. $\{y_1, y_4, y_7\}$
17. $\{y_2, y_5, y_9\}$
18. $\{y_3, y_8, y_9\}$
19. $\{\overline{y_1}, \overline{y_5}, \overline{y_8}\}$
20. $\{\overline{y_1}, \overline{y_6}, \overline{y_9}\}$
21. $\{\overline{y_2}, \overline{y_3}, \overline{y_6}\}$
22. $\{\overline{y_2}, \overline{y_4}, \overline{y_8}\}$
23. $\{\overline{y_3}, \overline{y_5}, \overline{y_7}\}$
24. $\{\overline{y_4}, \overline{y_7}, \overline{y_9}\}$

For a truth assignment with $\beta(x_3) = \beta(x_4) = F$, omitting the appearances of x_3 and x_4 in $\mathcal{R}$ has no effect on the satisfiability and the resulting instance is unsatisfiable (this can be checked by means of a SAT-solver, see also Lemma 1 in [8]). Now, let us consider both possibilities to assign a truth value to the variable x_2:

First, if $\beta(x_2) = T$, we can infer $\beta(x_3) = \beta(x_4) = F$ by clauses 2 and 3 in $\mathcal{Q}_2$ and, thus, no truth assignment that satisfies $\mathcal{Q} \cup \mathcal{R}$ sets x_2 to T.

Second, let $\beta(x_2) = F$. By the first clause in $\mathcal{Q}_2$, we have $\beta(x_1) = T$. Further, we may assume that $\beta(x_i) = T$ for at least one $x_i \in \{x_3, x_4\}$ (otherwise, β does not satisfy $\mathcal{R}$ as alluded to above). Then, by clauses 4 and 5 we have $\beta(x_j) = F$ for at least one $x_j \in \{x_5, x_6\}$. Next, clauses 6 and 7 imply $\beta(x_k) = T$ for at least one $x_k \in \{x_7, x_8\}$. Hence, we can replace clauses 8, 9, 10 and 11 by the 2-clauses $\{\overline{z_1}, \overline{z_2}\}$ and $\{\overline{z_3}, \overline{z_4}\}$ without affecting satisfiability. Recalling that $\beta(x_1) = T$ and $\beta(x_2) = F$, the first three clauses in the following set $\mathcal{S}$ of 3-clauses evaluate to $\{\overline{z_5}, \overline{z_6}\}$, $\{\overline{z_7}, \overline{z_8}\}$ and $\{z_7, z_{15}\}$, respectively (i.e., for the considered truth assignment β, we may omit the unsatisfied literals).

25. $\{\overline{x_1}, \overline{z_5}, \overline{z_6}\}$
26. $\{\overline{x_1}, \overline{z_7}, \overline{z_8}\}$
27. $\{x_2, z_7, z_{15}\}$
28. $\{z_1, z_6, z_8\}$
29. $\{z_1, z_{11}, z_{12}\}$
30. $\{z_2, z_6, z_8\}$
31. $\{z_2, z_{11}, z_{12}\}$
32. $\{z_3, z_5, z_9\}$
33. $\{z_3, z_{13}, z_{14}\}$
34. $\{z_4, z_5, z_{14}\}$
35. $\{z_4, z_9, z_{10}\}$
36. $\{z_7, z_{10}, z_{13}\}$
37. $\{\overline{z_5}, \overline{z_8}, \overline{z_{15}}\}$
38. $\{\overline{z_6}, \overline{z_7}, \overline{z_9}\}$
39. $\{\overline{z_9}, \overline{z_{11}}, \overline{z_{13}}\}$
40. $\{\overline{z_{10}}, \overline{z_{11}}, \overline{z_{14}}\}$
41. $\{\overline{z_{10}}, \overline{z_{12}}, \overline{z_{14}}\}$
42. $\{\overline{z_{12}}, \overline{z_{13}}, \overline{z_{15}}\}$

Now, the inferred 2-clauses

$$\{\overline{z_1}, \overline{z_2}\},\ \{\overline{z_3}, \overline{z_4}\},\ \{\overline{z_5}, \overline{z_6}\},\ \{\overline{z_7}, \overline{z_8}\} \text{ and } \{z_7, z_{15}\}$$

in conjunction with the clauses $\mathcal{S} \setminus \{\{\overline{x_1}, \overline{z_5}, \overline{z_6}\}, \{\overline{x_1}, \overline{z_7}, \overline{z_8}\}, \{x_2, z_7, z_{15}\}\}$ are unsatisfiable (again this can be checked by means of a SAT-solver, see also Lemma 2 in [8]).

Hence, the constructed set of 42 clauses

$$\mathcal{M} = \{\{x_1, x_2\}, \{\overline{x_2}, \overline{x_3}\}, \{\overline{x_2}, \overline{x_4}\}\} \cup \mathcal{Q}_3 \cup \mathcal{R} \cup \mathcal{S}$$

over the set of variables

$$V = \{x_1, \ldots, x_8\} \cup \{y_1, \ldots, y_9\} \cup \{z_1, \ldots, z_{15}\}$$

is unsatisfiable. We note that each literal appears at most twice in $\mathcal{M}$. The only variables that appear less than 4 times are x_1, x_5, x_6, y_6 and z_{15}, each of which appears once unnegated and twice negated. Now, let

$$\mathcal{M}^{(i)}(u_1, \overline{u_2}, \overline{u_3}) = \{\{x_1^i, x_2^i, u_1\}, \{\overline{x_2^i}, \overline{x_3^i}, \overline{u_2}\}, \{\overline{x_2^i}, \overline{x_4^i}, \overline{u_3}\}\} \cup \mathcal{Q}_3^i \cup \mathcal{R}^i \cup \mathcal{S}^i,$$

[6] See Döcker [8] for more details.

where $\mathcal{Q}_3^i, \mathcal{R}^i, \mathcal{S}^i$ are obtained from $\mathcal{Q}_3, \mathcal{R}, \mathcal{S}$ by replacing each variable, say v, with v^i (e.g. z_1 is replaced with z_1^i). Gadgets in the family $\mathcal{M}^{(i)}(u_1, \overline{u_2}, \overline{u_3})$ can now be used to transform the third, fourth and fifth clause in the unsatisfiable formula C. For instance, the clause $\{e, \bar{b}\}$ can be replaced with the clauses in $\mathcal{M}^{(1)}(e, \bar{b}, \bar{b})$. Observe that each clause in $\mathcal{M}^{(1)}(e, \bar{b}, \bar{b})$ is a monotone 3-clause. To deal with the clauses in C consisting of two unnegated variables and one negated variable, we introduce a second gadget $\overline{\mathcal{M}}^{(i)}(\overline{u_1}, u_2, u_3)$ which is obtained from $\mathcal{M}^{(i)}(u_1, \overline{u_2}, \overline{u_3})$ by negating each literal.

Finally, we are in a position to construct an unsatisfiable instance $\mathcal{U}$ of Monotone 3-Sat-(2, 2):

$$\begin{aligned}\mathcal{U} = \{\{\bar{a}, \bar{d}, \bar{f}\}, \{b, d, e\}\} &\cup \mathcal{M}^{(1)}(e, \bar{b}, \bar{b}) \cup \mathcal{M}^{(2)}(d, \bar{f}, \bar{c}) \cup \mathcal{M}^{(3)}(a, \bar{c}, \bar{e}) \\ &\cup \overline{\mathcal{M}}^{(4)}(\bar{e}, c, c) \cup \overline{\mathcal{M}}^{(5)}(\bar{d}, a, b) \cup \overline{\mathcal{M}}^{(6)}(\bar{a}, f, f) \\ &\cup \bigcup_{i \in \{1,5,6\}} \{\{x_i^1, x_i^2, x_i^3\}, \{\overline{x_i^4}, \overline{x_i^5}, \overline{x_i^6}\}\} \\ &\cup \{\{y_6^1, y_6^2, y_6^3\}, \{\overline{y_6^4}, \overline{y_6^5}, \overline{y_6^6}\}, \{z_{15}^1, z_{15}^2, z_{15}^3\}, \{\overline{z_{15}^4}, \overline{z_{15}^5}, \overline{z_{15}^6}\}\}\end{aligned}$$

We note that the clauses in

$$\begin{aligned}\mathcal{U}' = \{\{\bar{a}, \bar{d}, \bar{f}\}, \{b, d, e\}\} \cup \mathcal{M}^{(1)}(e, \bar{b}, \bar{b}) \cup \mathcal{M}^{(2)}(d, \bar{f}, \bar{c}) \cup \mathcal{M}^{(3)}(a, \bar{c}, \bar{e}) \cup \\ \overline{\mathcal{M}}^{(4)}(\bar{e}, c, c) \cup \overline{\mathcal{M}}^{(5)}(\bar{d}, a, b) \cup \overline{\mathcal{M}}^{(6)}(\bar{a}, f, f)\end{aligned}$$

form an unsatisfiable instance of Monotone 3-Sat-($\leq 2, \leq 2$); the purpose of the other clauses in $\mathcal{U}$ is to obtain the precise variable bounds on unnegated and negated appearances. Note that it is also possible to obtain an unsatisfiable instance of Monotone 3-Sat-(2, 2) by increasing the variable appearances in $\mathcal{U}'$ using the construction in Section 3, but the resulting instance contains more variables (resp. clauses) than $\mathcal{U}$. Now, since the formula C is unsatisfiable, there is at least one gadget containing only literals of variables in $\{a, b, c, d, e, f\}$ that evaluate to F. Hence, this gadget is equivalent to $\mathcal{M}$ or $\overline{\mathcal{M}}$ (the latter is obtained from the former by negating each literal) depending on whether it belongs to the family $\mathcal{M}^{(i)}(u_1, \overline{u_2}, \overline{u_3})$ or to the family $\overline{\mathcal{M}}^{(i)}(\overline{u_1}, u_2, u_3)$.

By counting the variables in $\mathcal{U}$, we obtain the following proposition.

Proposition 2. *There is an unsatisfiable instance of* Monotone 3-Sat-(2, 2) *with 198 variables and 264 clauses.*

4.2.2. NP-completeness of Monotone 3-Sat-(2, 2)

The existence of an unsatisfiable instance of Monotone 3-Sat-(2, 2) (see Proposition 2) allows us to derive NP-completeness of the problem. For that purpose, we will make use of the following lemma. We note that the union operator for multisets used in the following lemma refers to what is also called the sum of multisets, where the multiplicity of an element in the union is obtained by taking the sum of the corresponding multiplicities in the involved sets.

Lemma 4. *Given an unsatisfiable instance of* Monotone 3-Sat-(2, 2)*, we can construct a gadget $M_{C_{Sat}, \mathcal{L}}$ where*

- *C_{Sat} is a set of monotone 3-clauses over a set of variables V,*
- *$\mathcal{L}$ is a multiset of the literals $L_V = \{x_i, \overline{x_i} \mid x_i \in V\}$,*

such that the following three conditions are met:

(M1) C_{Sat} is satisfiable. Moreover, each truth assignment $\beta: V \to \{T, F\}$ that satisfies C_{Sat} does not satisfy any of the literals contained in $\mathcal{L}$.
(M2) Let $\mathcal{L} = \mathcal{L}_+ \cup \mathcal{L}_-$ be the partition of $\mathcal{L}$ where $\mathcal{L}_+$ contains the positive literals, and $\mathcal{L}_-$ contains the negative literals. Then, we have $|\mathcal{L}_+| = |\mathcal{L}_-| = 3q$ for some fixed integer $q \geq 1$.
(M3) Let $\mathcal{L}_{C_{Sat}}$ denote the multiset of literals that appear in C_{Sat}. Then, for each variable $x \in V$, $\mathcal{L}_{C_{Sat}} \cup \mathcal{L}$ contains x exactly twice as a positive literal and exactly twice as a negative literal.

Proof. Given an unsatisfiable instance of Monotone 3-Sat-(2, 2), let C' denote the corresponding set of clauses over variables $V' = \{x'_1, \ldots, x'_n\}$. Then, there is a strict subset $C'_{\text{Sat}} \subsetneq C'$ such that C'_{Sat} is satisfiable and $C'_{\text{Sat}} \cup \{c\}$ is unsatisfiable for all $c \in C' \setminus C'_{\text{Sat}}$. Now, each variable that appears in $C' \setminus C'_{\text{Sat}}$ has a forced truth value, i.e., if x'_i appears negated (unnegated) in $C' \setminus C'_{\text{Sat}}$, then any satisfying truth assignments for C'_{Sat} sets x'_i true (false). Otherwise, there a satisfying assignment for C'_{Sat} such that a clause in $C' \setminus C'_{\text{Sat}}$ is satisfied which is a contradiction since, by construction, such a clause does not exist. Also observe that no variable appears both negated and unnegated in $C' \setminus C'_{\text{Sat}}$. Let $\mathcal{L}'_+$ denote the multiset containing the positive literals appearing in $C' \setminus C'_{\text{Sat}}$ and $\mathcal{L}'_-$ the multiset containing the negative literals (e.g., if a negative literal ℓ appears twice in $C' \setminus C'_{\text{Sat}}$, then $\mathcal{L}'_-$ contains two copies of ℓ). Since all clauses contain exactly three distinct literals, the number of literals in $\mathcal{L}'_- \cup \mathcal{L}'_+$ is divisible by 3. Observe that we can only guarantee that $\mathcal{L}'_+ \neq \emptyset$ or $\mathcal{L}'_- \neq \emptyset$. Therefore, we introduce a copy of C' denoted by C'' (where the copy of C'_{Sat} is denoted by C''_{Sat}) over new variables $V'' = \{x''_1, \ldots, x''_n\}$,

where we negate each literal. Observe that C'' is an instance of Monotone 3-Sat-(2, 2). With $\mathcal{L}''_+$ and $\mathcal{L}''_-$ defined as above, the clauses

$$C_{\text{Sat}} = C'_{\text{Sat}} \cup C''_{\text{Sat}} \tag{1}$$

force all literals in

$$\mathcal{L} = \mathcal{L}'_+ \cup \mathcal{L}''_+ \cup \mathcal{L}'_- \cup \mathcal{L}''_- \tag{2}$$

to be set to false. By construction, we have

$$|\mathcal{L}'_+ \cup \mathcal{L}''_+| = |\mathcal{L}'_- \cup \mathcal{L}''_-| = 3q$$

with $q \geq 1$. It is now straightforward to verify that the gadget $M_{C,\mathcal{L}}$ with C and $\mathcal{L}$ as defined in Eqs. (1) and (2), respectively, has properties (M1), (M2) and (M3). □

Theorem 3. Monotone 3-Sat-(2, 2) *is NP-complete.*

Proof. We sketch a polynomial reduction from Monotone 3-Sat-(3, 3), for which NP-hardness is stated in Corollary 4, with clauses over a set of variables $V = \{x_1, x_2, \ldots, x_n\}$. For each variable $x_i \in V$, we replace each appearance with a separate new variable $x_{i,s}$, $1 \leq s \leq 6$, such that the positive literal x_i is replaced with $x_{i,1}$, $x_{i,3}$ and $x_{i,5}$, respectively, and the negative literal $\overline{x_i}$ is replaced with $x_{i,2}$, $x_{i,4}$ and $x_{i,6}$, respectively. We denote the resulting set of clauses by C. Next, for each $i \in \{1, \ldots, n\}$, we introduce the following clauses

$$C_i = \{\{x_{i,1}, x_{i,2}\},\ \{\overline{x_{i,2}}, \overline{x_{i,3}}\},\ \{x_{i,3}, x_{i,4}\},\ \{\overline{x_{i,4}}, \overline{x_{i,5}}\},\ \{x_{i,5}, x_{i,6}\},\ \{\overline{x_{i,6}}, \overline{x_{i,1}}\}\},$$

which are equivalent to the following cyclic chain of implications

$$\overline{x_{i,1}} \Rightarrow x_{i,2} \Rightarrow \overline{x_{i,3}} \Rightarrow x_{i,4} \Rightarrow \overline{x_{i,5}} \Rightarrow x_{i,6} \Rightarrow \overline{x_{i,1}}.$$

Hence, a truth assignment β satisfies these clauses if and only if

$$\beta(x_{i,1}) = \beta(x_{i,3}) = \beta(x_{i,5}) \neq \beta(x_{i,2}) = \beta(x_{i,4}) = \beta(x_{i,6})$$

for all $i \in \{1, 2, \ldots, n\}$. Observe that each variable $x_{i,s}$ appears exactly once unnegated and exactly once negated in C_i and exactly once unnegated in the remaining clauses. In order to increase the number of negated appearances of each variable by one, we introduce

$$C'_i = \{\{\overline{x_{i,1}}, \overline{x_{i,2}}, \overline{x_{i,6}}\}, \{\overline{x_{i,3}}, \overline{x_{i,4}}, \overline{x_{i,5}}\}\}$$

for each $i \in \{1, 2, \ldots, n\}$. Recall that a truth assignment that satisfies C_i assigns different truth values to $x_{i,s}$ and $x_{i,t}$, where $s \in \{1, 3, 5\}$ and $t \in \{2, 4, 6\}$. Hence, a truth assignment that satisfies C_i also satisfies C'_i.

Finally, we deal with the 2-clauses introduced above. Consider an unsatisfiable instance of Monotone 3-Sat-(2, 2), and apply Lemma 4 and the gadget $M_{C_{\text{Sat}},\mathcal{L}}$ used in that lemma. Recall that the corresponding set of clauses C_{Sat} can be satisfied only by assignments that do not satisfy any of the literals contained in the multiset $\mathcal{L}$. Further, the multiset $\mathcal{L}$ contains exactly $3q$ positive and exactly $3q$ negative literals for some fixed integer $q \geq 1$. Note that if we knew that $q = 1$, then we could simply use n instances of this gadget to pad all 2-clauses, i.e., $\bigcup_{i=1}^{n} C_i$, since each C_i contains exactly 3 positive and exactly 3 negative 2-clauses. As we cannot make this assumption, we solve the parity problem as follows. First, we replace the clauses

$$\mathcal{C} = C \cup \bigcup_{i=1}^{n} \left(C_i \cup C'_i\right)$$

with q copies $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_q$ such that the variables of the kth copy are

$$V_k = \{x^k_{i,s} \mid 1 \leq i \leq n \text{ and } 1 \leq s \leq 6\}.$$

Now, the set of clauses $\bigcup_{i=1}^{q} \mathcal{C}_i$ contains exactly $q \cdot 3n$ negative 2-clauses and exactly $q \cdot 3n$ positive 2-clauses. Then, we use n instances of the gadget $M_{C_{\text{Sat}},\mathcal{L}}$, where each instance has their own new variables, to pad these 2-clauses. To be precise, we introduce the set of clauses $\bigcup_{i=1}^{n} C^i_{\text{Sat}}$, where C^i_{Sat} is the set of clauses corresponding to the ith instance of the gadget. The corresponding multiset of literals is $\bigcup_{i=1}^{n} \mathcal{L}^i$ and, by Property (M2), contains exactly $n \cdot 3q$ positive literals and exactly $n \cdot 3q$ negative literals. Hence, we can pair each positive (resp. negative) 2-clause with exactly one positive (resp. negative) literal that evaluates to false by Property (M1). Note that this is a one-to-one correspondence. Finally, replace each 2-clause with this union of the 2-clause with the paired literal. By construction and Property (M3) in Lemma 4, the resulting instance is indeed an instance of Monotone 3-Sat-(2, 2). It is straightforward to verify that the instance of Monotone 3-Sat-(2, 2) is satisfiable if and only if the given instance of Monotone 3-Sat-(3, 3) is satisfiable. □

Hence, with Theorem 1 and Corollary 1, respectively, we can derive the following two corollaries.

Corollary 5. Monotone 3-Sat-(p, q) *is NP-complete for each fixed pair*

$$(p, q) \in \{(r, s), (s, r) \mid r \geq 2, s \geq 2\}.$$

Corollary 6. Monotone 3-Sat-(k, k) *is NP-complete for all* $k \geq 2$.

5. Monotone 3-Sat with exactly one negated appearance per variable

In this section Monotone 3-Sat is analyzed restricted to instances in which each variable appears exactly once negated. We settle the computational complexity status of Monotone 3-Sat-$(k, 1)$ for each fixed $k \geq 5$. We do not answer the question of its computational complexity for $k \in \{3, 4\}$, which, to the best of our knowledge, is still open. However, for $k \in \{3, 4\}$ we can show that when restricted to a "small" number of unnegated appearances each instance of Monotone 3-Sat-$(k, 1)$ is satisfiable.

5.1. On Monotone 3-Sat-$(k, 1)$ for $k \geq 5$

It will be useful to introduce some additional notation. Let V be a set of variables and $C, C' \subseteq \mathcal{P}(V)$ non-empty sets of clauses, where $\mathcal{P}(V)$ denotes the power set of V. We say that C *subsumes* C' if for each clause $c' \in C'$ there is a clause $c \in C$ such that $c \subseteq c'$. Consequently, if C is satisfiable and subsumes C', then C' is satisfiable. On the other hand, if C' is unsatisfiable, then so is C.

We begin with Lemma 5 which will be used for proving the computational complexity result for Monotone 3-Sat-$(k, 1)$ for $k \geq 5$.

Lemma 5. *Let* $\mathcal{D}(X)$ *with* $X = (x_1, x_2, \ldots, x_6)$ *be the following set of clauses, where* $V_{aux} = \{a, b, \ldots, i\}$ *are new variables.*

1. $\{\bar{a}, \bar{b}, \bar{c}\}$
2. $\{\bar{d}, \bar{e}, \bar{f}\}$
3. $\{\bar{g}, \bar{h}, \bar{i}\}$
4. $\{a, f, g\}$
5. $\{a, f, h\}$
6. $\{a, f, i\}$
7. $\{b, d, g\}$
8. $\{b, d, h\}$
9. $\{b, d, i\}$
10. $\{c, e, g\}$
11. $\{c, e, h\}$
12. $\{c, e, i\}$
13. $\{a, d, x_1\}$
14. $\{a, e, x_2\}$
15. $\{b, e, x_3\}$
16. $\{b, f, x_4\}$
17. $\{c, d, x_5\}$
18. $\{c, f, x_6\}$

Then, a truth assignment β *for* X *can be extended to a truth assignment* β' *for* $X \cup V_{aux}$ *that satisfies* $\mathcal{D}(X)$ *if and only if* $\beta(x_i) = T$ *for at least one* $x_i \in X$.

Proof. First, let $\beta(x_i) = F$ for each $x_i \in X$. Then, $\mathcal{D}(X)$ can be simplified by removing the variables $x_1, \ldots, x_6$ from clauses 13–18 (consequently, these clauses become 2-clauses). Let $\mathcal{D}(\emptyset)$ denote the resulting set of clauses. Since each variable in V_{aux} appears only once negated, we can assume that a truth assignment that satisfies $\mathcal{D}(X)$ assigns the truth value false to exactly one variable of each negative clause (the corresponding literal evaluates to true). Hence, clauses 1, 2 and 3 in conjunction with the following set of $3^3 = 27$ clauses

$$\mathcal{U} = \{\{u, v, w\} \mid (u, v, w) \in \{a, b, c\} \times \{d, e, f\} \times \{g, h, i\}\}$$

are unsatisfiable. Now, we show that $\mathcal{D}(\emptyset)$ subsumes $\mathcal{U}$. Observe that the clauses 4–18 in $\mathcal{D}(\emptyset)$ subsume pairwise disjoint subsets of $\mathcal{U}$. Further, each of the clauses 4–12, say c_j, subsumes the subset $\{c_j\}$ of $\mathcal{U}$ and each of the clauses 13–18 subsumes a subset of size exactly 3 (e.g. clause 13 subsumes $\{\{a, d, g\}, \{a, d, h\}, \{a, d, i\}\}$). It follows that clauses 4–18 subsume a subset $\mathcal{U}' \subseteq \mathcal{U}$ with 27 distinct clauses. Thus, we have $\mathcal{U}' = \mathcal{U}$, which means that $\mathcal{D}(\emptyset)$ subsumes $\mathcal{U}$. Hence, $\mathcal{D}(\emptyset)$ is unsatisfiable and β cannot be extended to a truth assignment β' for $X \cup V_{\text{aux}}$ that satisfies $\mathcal{D}(X)$.

Second, let $\beta(x_i) = T$ for some $x_i \in X$. Then, the set $\mathcal{U}_{\text{sat}}$ of clauses of $\mathcal{U}$ that are not subsumed by $\mathcal{D}(\emptyset)$ with the satisfied clause (resp. clauses) removed is non-empty. More precisely, for each variable in X that is true, we have a subset of size 3 that is not subsumed (recall that clauses 4–18 in $\mathcal{D}(\emptyset)$ subsume pairwise disjoint subsets of $\mathcal{U}$). Now, each clause $\{u, v, w\} \in \mathcal{U}_{\text{sat}}$ induces a truth assignment β' for $X \cup V_{\text{aux}}$ as follows: Set u, v, w to false and all other variables in V_{aux} to true. By construction, each negative clause in $\mathcal{D}(X)$ is satisfied by setting the variables contained in an element of $\mathcal{U}$ to false. Further, each remaining positive clause (i.e., after removing clauses satisfied by β) contains at least one variable $z \notin \{u, v, w\}$, and thus is satisfied. Hence, β' satisfies $\mathcal{D}(X)$. □

Let y be a new variable. By construction, the set of clauses

$$\mathcal{F}(y) = \mathcal{D}(X_1) \cup \mathcal{D}(X_2) \cup \mathcal{D}(X_3) \cup \{\{\overline{u_1}, \overline{u_2}, \overline{u_3}\}\},$$

with new variables u_1, u_2, u_3 and $X_i = (y, u_i, u_i, u_i, u_i, u_i)$ for $i \in \{1, 2, 3\}$ forces y to true, where $\mathcal{D}(X_i)$ refers to the set of clauses in Lemma 5. Note that each new variable except y appears at most five times unnegated and once negated and

y appears exactly three times unnegated. We can use the gadget $\mathcal{F}(y)$ to obtain a second gadget that forces some new variable z to false:

$$\mathcal{F}'(\bar{z}) = \{\{\bar{z}, \overline{v}, \overline{w}\}\} \cup \mathcal{F}(v) \cup \mathcal{F}(w).$$

Observe that each variable of $\mathcal{F}'(\bar{z})$ appears at most five times unnegated and once negated; and z has no unnegated appearance.

We point out that with the help of the above gadget $\mathcal{F}(y)$ we can construct an unsatisfiable instance of Monotone 3-Sat-($\leq 5, 1$) as follows:

$$\mathcal{F}(y_1) \cup \mathcal{F}(y_2) \cup \mathcal{F}(y_3) \cup \{\{\overline{y_1}, \overline{y_2}, \overline{y_3}\}\}.$$

Hence, we get the following proposition.

Proposition 3. *There exists an unsatisfiable instance of* Monotone 3-Sat-($\leq 5, 1$) *with 166 clauses and 93 variables.*

In fact, we are ready to prove hardness of Monotone 3-Sat-($\leq 5, 1$). Consider the reduction from 3-Sat-(2, 2) in the proof of Theorem 2 and replace the enforcers $\mathcal{H}$ and $\mathcal{H}'$ by the enforcers $\mathcal{F}$ and $\mathcal{F}'$ defined above. Then we obtain the following theorem.

Theorem 4. Monotone 3-Sat-($\leq 5, 1$) *is NP-complete.*

Observe that, for any k, Monotone 3-Sat-($\leq k, 1$) and Monotone 3-Sat-($\leq k, \leq 1$) are polynomially equivalent, since any clause containing a variable that appears only unnegated can be removed from the formula without affecting satisfiability. Hence, with that observation and the above theorem Corollary 1 implies hardness of Monotone 3-Sat-($k, 1$), for any choice of $k \geq 5$.

Corollary 7. Monotone 3-Sat-($k, 1$) *is NP-complete for all* $k \geq 5$.

5.2. *On* Monotone 3-Sat-(3, 1) *and* Monotone 3-Sat-(4, 1)

We now discuss some properties of Monotone 3-Sat-($k, 1$), and conclude the section with two corollaries stating that for certain "small" numbers of variable appearances each instance of Monotone 3-Sat-(3, 1) and Monotone 3-Sat-(4, 1) is satisfiable.

Let $V = \{x_1, x_2, \ldots, x_n\}$ be a set of variables. First, the number of variables $n = |V|$ is divisible by 3 since otherwise there is a negative clause containing less than three variables. Second, we can restrict our attention to truth assignments that set exactly one literal in each negative clause to true (i.e., the corresponding variable to false), since we can simply modify any satisfying truth assignment to meet that requirement. Moreover, we can assume that (after relabeling) the negative clauses are

$$\{\overline{x_1}, \overline{x_2}, \overline{x_3}\}, \{\overline{x_4}, \overline{x_5}, \overline{x_6}\} \ldots, \{\overline{x_{n-2}}, \overline{x_{n-1}}, \overline{x_n}\}.$$

Hence, we can represent any truth assignment of interest by a tuple

$$(x_{i_1}, \ldots, x_{i_{\frac{n}{3}}}) \in \{x_1, x_2, x_3\} \times \{x_4, x_5, x_6\} \times \cdots \times \{x_{n-2}, x_{n-1}, x_n\},$$

such that the corresponding truth assignment $\beta: V \to \{T, F\}$ is defined as $\beta(x_j) = F$ if and only if $j \in \{i_1, i_2, \ldots, i_{\frac{n}{3}}\}$. It is convenient to define

$$\mathcal{M}_n = \{\{x_{i_1}, \ldots, x_{i_{\frac{n}{3}}}\} \mid (x_{i_1}, \ldots, x_{i_{\frac{n}{3}}}) \in \{x_1, x_2, x_3\} \times \cdots \times \{x_{n-2}, x_{n-1}, x_n\}\}$$

which represents the truth assignments that set exactly one literal in each negative clause to true (in an instance of Monotone 3-Sat-($k, 1$) with n variables).

We are now in a position to establish the following theorem. The corresponding proof using a probabilistic argument (cf. Gebauer et al. [10]) has been brought to our attention by an anonymous referee and provides a more concise way to obtain this result compared to our original proof (cf. [4]). We remark that the given bound is the best possible since there is an unsatisfiable instance with 27 positive 3-clauses (presented in the proof of Lemma 5).

Theorem 5. *Let* $V = \{x_1, x_2, \ldots, x_n\}$ *be a set of variables. An instance of* Monotone 3-Sat *with a collection of clauses*

$$C = \{\{\overline{x_1}, \overline{x_2}, \overline{x_3}\}, \{\overline{x_4}, \overline{x_5}, \overline{x_6}\} \ldots, \{\overline{x_{n-2}}, \overline{x_{n-1}}, \overline{x_n}\}\} \cup C^+,$$

where C^+ *is a collection of positive 3-clauses and* $|C^+| < 27$ *is satisfiable.*

Proof. As alluded to above, it is sufficient to consider truth assignments represented by

$$\mathcal{M}_n = \{\{x_{i_1}, \ldots, x_{i_{\frac{n}{3}}}\} \mid (x_{i_1}, \ldots, x_{i_{\frac{n}{3}}}) \in \{x_1, x_2, x_3\} \times \cdots \times \{x_{n-2}, x_{n-1}, x_n\}\}$$

that set exactly one variable in each negative clause to false. Now, consider a truth assignment $\beta_r : V \to \{T, F\}$ sampled from the uniform distribution on $\mathcal{M}_n$. Without loss of generality, we can assume that no clause in C^+ contains more than one variable from each group $\{x_{3i-2}, x_{3i-1}, x_{3i}\}$, $1 \leq i \leq \frac{n}{3}$ (each truth assignment in $\mathcal{M}_n$ satisfies any clause with at least two variables from the same group). Noting that $\beta_r(v) = F$ with probability $\frac{1}{3}$ for each $v \in V$, the probability that β_r does not satisfy a clause is then given by $\left(\frac{1}{3}\right)^3 = \frac{1}{27}$. By linearity of expectation, the formula C is not satisfied by β_r with probability

$$\sum_{c \in C^+} \frac{1}{27} = \frac{|C^+|}{27} < 1.$$

Hence, there is a positive probability that the random truth assignment β_r satisfies the formula C. Thus, C is satisfiable. □

Corollary 8. *Each instance of* Monotone 3-Sat-(4, 1) *with less than 21 variables is satisfiable.*

Corollary 9. *Each instance of* Monotone 3-Sat-(3, 1) *with less than 27 variables is satisfiable.*

6. On a restricted variant of Monotone 3-Sat-E4

Finally, we consider Monotone 3-Sat-E4, i.e., with exactly four appearances of each variable. We begin this short section with the following lemma.

Lemma 6. *Consider the following collection $\mathcal{C}(x, y)$ of monotone clauses, where $V_{aux} = \{a, b, \ldots, h\}$ are new variables.*

1. $\{\bar{a}, \bar{c}, \bar{e}\}$
2. $\{\bar{a}, \bar{c}, \bar{f}\}$
3. $\{\bar{a}, \bar{d}, \bar{g}\}$
4. $\{\bar{b}, \bar{c}, \bar{h}\}$
5. $\{\bar{b}, \bar{e}, \bar{g}\}$
6. $\{\bar{b}, \bar{f}, \bar{g}\}$
7. $\{\bar{d}, \bar{e}, \bar{h}\}$
8. $\{\bar{d}, \bar{f}, \bar{h}\}$
9. $\{a, b, x\}$
10. $\{c, d, x\}$
11. $\{e, f, x\}$
12. $\{g, h, y\}$

Then, a truth assignment β for $\{x, y\}$ can be extended to a truth assignment β' for $\{x, y\} \cup V_{aux}$ that satisfies $\mathcal{C}(x, y)$ if and only if $\beta(v) = T$ for at least one $v \in \{x, y\}$.

Proof. We show that this collection of clauses is unsatisfiable if x and y are both set to false. By clause 11 at least one of e, f has to be set to true. As a consequence clauses 1, 2; 5, 6 and 7, 8 imply that the additional clauses *i.* $\{\bar{a}, \bar{c}\}$; *ii.* $\{\bar{b}, \bar{g}\}$ and *iii.* $\{\bar{d}, \bar{h}\}$ would have to be satisfied as well. First consider any variable assignment β with $\beta(g) = F$:

$$\beta(g) = F \overset{12.}{\Rightarrow} \beta(h) = T \overset{iii.}{\Rightarrow} \beta(d) = F \overset{10.}{\Rightarrow} \beta(c) = T \overset{i.}{\Rightarrow} \beta(a) = F \overset{9.}{\Rightarrow} \beta(b) = T.$$

Thus, clause 4 is not satisfied. Now we consider the other case $\beta(g) = T$:

$$\beta(g) = T \overset{ii.}{\Rightarrow} \beta(b) = F \overset{9.}{\Rightarrow} \beta(a) = T \overset{i.}{\Rightarrow} \beta(c) = F \overset{10.}{\Rightarrow} \beta(d) = T.$$

Thus, clause 3 is not satisfied. Consequently, the collection of clauses is unsatisfiable if $\beta(x) = \beta(y) = F$. Without clause 12, there is a satisfying truth assignment: Set all variables in $\{a, g, h\}$ to false and all variables in $\{b, c, d, e, f\}$ to true. Hence, the collection of clauses is satisfiable if $\beta(y) = T$. Finally, if $\beta(x) = T$, we can satisfy all clauses by setting h to true and all variables in $\{a, b, \ldots, g\}$ to false. □

Lemma 6 implies the following corollary, where $\mathcal{C}(\cdot, \cdot)$ refers to the respective set of clauses introduced in Lemma 6.

Corollary 10. *Consider the collection of clauses $\mathcal{B}(x, y, z) = \mathcal{C}(u, x) \cup \mathcal{C}(v, y) \cup \mathcal{C}(w, z) \cup \{\{\bar{u}, \bar{v}, \bar{w}\}\}$, and let V be its associated set of variables. Then, a truth assignment β for $\{x, y, z\}$ can be extended to a truth assignment β' for V that satisfies $\mathcal{B}(x, y, z)$ if and only if $\beta(v) = T$ for at least one $v \in \{x, y, z\}$.*

Corollary 11. *Consider the collection of clauses $\bar{\mathcal{B}}(\bar{x}, \bar{y}, \bar{z})$ obtained from $\mathcal{B}(x, y, z)$ by replacing each literal with its negation, and let V be its associated set of variables. Then, a truth assignment β for $\{x, y, z\}$ can be extended to a truth assignment β' for V that satisfies $\bar{\mathcal{B}}(\bar{x}, \bar{y}, \bar{z})$ if and only if $\beta(v) = F$ for at least one $v \in \{x, y, z\}$.*

Remark. Each collection $\mathcal{C}(\cdot, \cdot)$ of clauses has its own new auxiliary variables; also, each instance of a gadget $\mathcal{B}(x, y, z)$ (resp. $\bar{\mathcal{B}}(\bar{x}, \bar{y}, \bar{z})$) has its own new auxiliary variables (i.e., the variables that are not in $\{x, y, z\}$).

The above corollaries will be useful to prove that Monotone 3-Sat-E4 is NP-complete even when restricted to instances in which each variable appears either three times unnegated and once negated or three times negated and once unnegated. The following theorem is based on the same reduction as the proofs of Theorems 2 and 4, respectively. Since we consider a slightly different type of restriction here, and to make verification of the precise bounds on the variable appearances easier, we included the complete proof again.

Theorem 6. Monotone 3-Sat-E4 *is NP-complete even if each variable appears three times unnegated and once negated or three times negated and once unnegated.*

Proof. We show NP-hardness by reducing from 3-Sat-(2, 2), for which NP-hardness was established by Berman et al. [2, Theorem 1]. Given an instance $\mathcal{I}$ of the latter with a set V of variables and a set C of clauses over V, let $n := |V|$. For each variable $x_i \in V$, we introduce two new variables $x_{i,1}$, $x_{i,2}$ and replace the two negated appearances with $x_{i,1}$ and the two unnegated appearances with $x_{i,2}$, respectively. Then, we remove all negations and introduce the following clauses for $i \in \{1, 2, \ldots, n\}$, where z_i and y_i are new variables:

$$\{\{x_{i,1}, x_{i,2}, y_i\}, \{\overline{x_{i,1}}, \overline{x_{i,2}}, \overline{z_i}\}\} \cup \bar{\mathcal{B}}(\overline{y_i}, \overline{y_i}, \overline{y_i}) \cup \mathcal{B}(z_i, z_i, z_i).$$

Let V_i denote the variables appearing in the clauses introduced above. By construction and Corollaries 10 and 11, a truth assignment β_i for $\{x_{i,1}, x_{i,2}\}$ can be extended to a truth assignment β_i' for V_i that satisfies these clauses if and only if $\beta_i(x_{i,1}) \neq \beta_i(x_{i,2})$. Now it is straightforward to verify that the constructed set of clauses is satisfiable if and only if the given instance $\mathcal{I}$ is satisfiable.

By construction of $\bar{\mathcal{B}}(\overline{y_i}, \overline{y_i}, \overline{y_i})$ and $\mathcal{B}(z_i, z_i, z_i)$, each variable in $\bigcup_{i=1}^{n} V_i$ appears three times unnegated and once negated or three times negated and once unnegated. Moreover, each variable in $\bigcup_{i=1}^{n}\{x_{i,1}, x_{i,2}\}$ appears once unnegated and once negated in the introduced clauses, and twice unnegated in the original clause set. Also observe that, by construction, all clauses are monotone. Hence, we constructed an instance of Monotone 3-Sat-E4 where each variable appears three times negated and once unnegated or three times unnegated and once negated.

We conclude the proof by remarking that the transformation is polynomial. □

Finally, dropping the monotonicity condition we point out that Theorem 6 implies also an interesting hardness result for 3-Sat-E4. For instance, replacing each variable x that appears negated exactly three times and unnegated exactly once with a new variable z such that literal z replaces literal $\bar{x}$ and literal $\bar{z}$ replaces literal x, it follows that 3-Sat-E4 is NP-complete even if each variable appears exactly three times unnegated and exactly once negated. An analogous result follows for the case that each variable appears exactly three times negated and exactly once unnegated. Therewith, we complement a result by Berman et al. [2] stating that 3-Sat-E4 is NP-complete even if each variable appears exactly twice unnegated and exactly twice negated. We summarize these findings in terms of the corollary below (for the sake of completeness, we include also the result by Berman et al. [2, Theorem 1]).

Corollary 12. 3-Sat-E4 *is NP-complete even if either*

- *each variable appears exactly three times unnegated and once negated, or*
- *each variable appears exactly three times negated and once unnegated, or*
- *each variable appears exactly twice unnegated and twice negated [2].*

7. Conclusion

We have shown that a restricted variant of Monotone 3-Sat-E4 is NP-complete, and that Monotone 3-Sat-(2, 2) is NP-complete. In addition, our results show that in fact Monotone 3-Sat-(p, q) is NP-complete for each fixed pair $(p, q) \in \{(r, s), (s, r) \mid r \geq 2, s \geq 2\}$.

In particular, Monotone 3-Sat-(k, k) is NP-complete for all $k \geq 2$. By a result of Tovey [21, Theorem 2.4] the latter problem is trivial for $k = 1$, i.e., all such instances are satisfiable. Therewith, for Monotone 3-Sat with balanced variable appearances our results establish a sharp boundary between NP-complete and polynomial time solvable cases.

Another focus of the paper was laid on Monotone 3-Sat-$(k, 1)$, where each variable appears exactly k times unnegated and once negated. For this variant, we proved NP-completeness for all $k \geq 5$. Again, by Tovey [21, Theorem 2.4] the problem is trivial for $k \leq 2$. The cases $k = 3$ and $k = 4$ are, to the best of our knowledge, open; we hence state the following challenge for future research:

Challenge. Is Monotone 3-Sat-$(k, 1)$ NP-hard for $k \in \{3, 4\}$?

CRediT authorship contribution statement

Andreas Darmann: Conceptualization, Writing - review & editing, Writing - original draft. **Janosch Döcker:** Conceptualization, Writing - review & editing, Writing - original draft.

Acknowledgments

The authors are grateful for the diligent report and the useful hints – in particular for the findings presented in Section 3 – of an anonymous referee which improved (and shortened) the presentation of the paper significantly. In addition, the authors would like to thank Britta Dorn for insightful comments on a draft containing the construction of an unsatisfiable instance of Monotone 3-Sat-(2, 2).

References

[1] M. de Berg, A. Khosravi, Optimal binary space partitions for segments in the plane, Internat. J. Comput. Geom. Appl. 22 (03) (2012) 187–205.
[2] P. Berman, M. Karpinski, A.D. Scott, Approximation hardness of short symmetric instances of MAX-3SAT, in: Electronic Colloquium on Computational Complexity, Report No. 49, 2003.
[3] S.A. Cook, The complexity of theorem-proving procedures, in: Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, ACM, 1971, pp. 151–158.
[4] A. Darmann, J. Döcker, On simplified NP-complete variants of Not-All-Equal 3-SAT and 3-SAT, 2019, arXiv:1908.04198 [cs.CC].
[5] A. Darmann, J. Döcker, On a simple hard variant of Not-All-Equal 3-SAT, Theoret. Comput. Sci. 815 (2020) 147–152.
[6] A. Darmann, J. Döcker, B. Dorn, The monotone satisfiability problem with bounded variable appearances, Internat. J. Found Comput. Sci. 29 (06) (2018) 979–993.
[7] D. Devlin, B. O'Sullivan, Satisfiability as a classification problem, in: Proceedings of the 19th Irish Conference on Artificial Intelligence and Cognitive Science, 2008.
[8] J. Döcker, Monotone 3-SAT-(2, 2) is NP-complete, 2019, arXiv:1912.08032 [cs.CC].
[9] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman and Company, New York, 1979.
[10] H. Gebauer, R.A. Moser, D. Scheder, E. Welzl, The Lovász local lemma and satisfiability, in: S. Albers, H. Alt, S. Näher (Eds.), Efficient Algorithms: Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday, Springer, 2009, pp. 30–54.
[11] M.E. Gold, Complexity of automaton identification from given data, Inf. Control 37 (3) (1978) 302–320.
[12] A. Horbach, T. Bartsch, D. Briskorn, Using a SAT-solver to schedule sports leagues, J. Sched. 15 (1) (2012) 117–125.
[13] H. Kautz, B. Selman, Pushing the envelope: Planning, propositional logic, and stochastic search, in: Proceedings of the 13th National Conference on Artificial Intelligence (AAAI'96), AAAI Press, 1996, pp. 1194–1201.
[14] J. Kratochvíl, A special planar satisfiability problem and a consequence of its NP-completeness, Discrete Appl. Math. 52 (3) (1994) 233–252.
[15] W.N. Li, Two-segmented channel routing is strong NP-complete, Discrete Appl. Math. 78 (1–3) (1997) 291–298.
[16] D. Lichtenstein, Planar formulae and their uses, SIAM J. Comput. 11 (2) (1982) 329–343.
[17] G. Nam, K.A. Sakallah, R.A. Rutenbar, Satisfiability-based layout revisited: Detailed routing of complex FPGAs via search-based Boolean SAT, in: Proceedings of the ACM/SIGDA 7th International Symposium on Field Programmable Gate Arrays (FPGA '99), ACM, 1999, pp. 167–175.
[18] D. Paulusma, S. Szeider, On the parameterized complexity of (k,s)-SAT, Inform. Process. Lett. 143 (2019) 34–36.
[19] A. Pilz, Planar 3-SAT with a clause/variable cycle, Discrete Math. Theor. Comput. Sci. 21 (3) (2019).
[20] S. Porschen, B. Randerath, E. Speckenmeyer, Linear time algorithms for some Not-All-Equal satisfiability problems, in: E. Giunchiglia, A. Tacchella (Eds.), Theory and Applications of Satisfiability Testing (SAT 2003), Springer, 2004, pp. 172–187.
[21] C.A. Tovey, A simplified NP-complete satisfiability problem, Discrete Appl. Math. 8 (1) (1984) 85–89.

1.7 On a simple hard variant of Not-All-Equal 3-SAT

The following paper [DD20] is also available online at the following URL: `https://doi.org/10.1016/j.tcs.2020.02.010`.

Theoretical Computer Science 815 (2020) 147–152

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Note

On a simple hard variant of Not-All-Equal 3-Sat ☆

Andreas Darmann [a], Janosch Döcker [b,*]

[a] Institute of Public Economics, University of Graz, Austria
[b] Department of Computer Science, University of Tübingen, Germany

ARTICLE INFO

Article history:
Received 24 August 2019
Received in revised form 5 January 2020
Accepted 8 February 2020
Available online 11 February 2020
Communicated by L.M. Kirousis

Keywords:
Not-All-Equal 3-Satisfiability
Linear formulas
Bounded variable appearances
Computational complexity

ABSTRACT

We consider a simplified version of Not-All-Equal 3-Sat, a variation of the famous Satisfiability problem, where each clause is made up of exactly three distinct literals and the question is whether there exists a truth assignment such that for each clause at least one literal is set to true and at least one is set to false. We show that Not-All-Equal 3-Sat remains NP-complete if (1) each variable appears exactly four times, (2) there are no negations in the formula, and (3) the formula is linear, i.e., each pair of distinct clauses shares at most one variable. Therewith, we improve upon two results in the literature.

1. Introduction

We consider a simplified variant of the Satisfiability problem, a classical and fundamental problem in computer science of significant theoretical and practical relevance, with applications in various fields such as, e.g., artificial intelligence (Kautz and Selman [5]), routing (Nam et al. [6]), or scheduling (Horbach et al. [3]).

In this work we are concerned with the Not-All-Equal Satisfiability problem, which asks whether there is a truth assignment such that for each clause at least one literal evaluates to true and at least one to false, respectively. As a consequence of Schaefer's dichotomy theorem [9] Not-All-Equal Satisfiability is NP-complete even if each clause is made up of three literals. In what follows, we refer to Not-All-Equal 3-Sat as the restriction of Not-All-Equal Satisfiability to formulas in which each clause contains exactly three distinct literals, and to Not-All-Equal 3-Sat* as the restriction to formulas with three—not necessarily distinct—literals per clause.

The focus of this work is laid on the simplified variant Monotone Not-All-Equal 3-Sat, which restricts Not-All-Equal 3-Sat to *monotone*[1] instances in which negations are completely absent.

Dehghan et al. [2] show that Monotone Not-All-Equal Satisfiability remains NP-complete if each variable appears unnegated exactly three times, and each clause is a disjunction of either two or three distinct variables. In contrast, Monotone Not-All-Equal 3-Sat can be solved in polynomial time in case that each variable appears exactly three times (Porschen et al. [7, Theorem 4]) and the corresponding proof can be adapted to show that the result also holds if each variable appears *at most* three times [7, p. 186]. However, Porschen et al. [8] show that Monotone Not-All-Equal 3-Sat is NP-complete even for *linear* instances, where each pair of distinct clauses shares at most one variable. In this paper we improve upon

☆ A preliminary version of this paper is contained in the online preprint [1] on arXiv.

* Corresponding author.
E-mail addresses: andreas.darmann@uni-graz.at (A. Darmann), janosch.doecker@uni-tuebingen.de (J. Döcker).

[1] We point out that *monotonicity* has different meanings for 3-Satisfiability and Not-All-Equal 3-Satisfiability.

https://doi.org/10.1016/j.tcs.2020.02.010
0304-3975/

the result of Porschen et al. [8] by showing that Monotone Not-All-Equal 3-Sat remains NP-complete even if further simplified to linear instances in which each variable appears exactly four times. Therewith, we also improve upon a result of Karpinski and Piecuch [4], who show that Not-All-Equal 3-Sat* (possibly with duplicates of literals in the same clause) is NP-complete even if each variable appears at most 4 times in the formula. We point out that our main result can be used not only for simplifying NP-hardness proofs of new or well-known problems but also for showing NP-hardness in restricted settings such as, e.g., proving NP-hardness of Hypergraph 2-Colorability for linear, 3-uniform, 4-regular hypergraphs.[2]

2. Formal framework

Let $V = \{x_1, \dots, x_n\}$ denote a finite set of (propositional) *variables*. A *literal* is either a variable x or a negated variable $\overline{x}$; let $L_V = \{x, \overline{x} \mid x \in V\}$ denote the set of literals. A *clause* is a set of literals, i.e., a subset of L_V. A k-clause contains exactly k distinct literals. A clause is *monotone* if all of its contained variables are unnegated. A Boolean formula C in *conjunctive normal form* (CNF) is a finite set of m clauses, i.e., $C = \{c_1, \dots, c_m\}$. We denote the total number of appearances of a variable $x \in V$ in a formula C by $a(x)$. A Boolean formula is *linear* if all pairs of distinct clauses share at most one variable.

A *truth assignment* β for the set of variables V is a mapping $\beta\colon V \to \{T, F\}$, i.e., assigns to each variable the value "true" (T) or "false" (F). In order to extend β to a truth assignment on literals, we set $\beta(\overline{x_i}) = T$ if $\beta(x_i) = F$ and $\beta(\overline{x_i}) = F$ otherwise. A truth assignment *satisfies* a clause c_j if $\beta(x) = T$ for at least one $x \in c_j$. A truth assignment *nae-satisfies* a clause c_j if there are literals $x, x' \in c_j$ such that $\beta(x) \neq \beta(x')$. A truth assignment satisfies (nae-satisfies) a Boolean formula C in CNF if it satisfies (nae-satisfies) all clauses in C. We say that a truth assignment β' for V' *extends* a truth assignment β for V if $V \subseteq V'$ and $\beta'(x) = \beta(x)$ for all $x \in V$.

Finally, abbreviating Not-All-Equal 3-Sat with NAE-3-Sat, we state the considered decision problem Monotone NAE-3-Sat-E4: Given a set V of variables and a collection C of monotone 3-clauses over V such that every variable appears in exactly four clauses, is there a truth assignment for V that nae-satisfies C?

3. A simple hard variant of Not-All-Equal 3-Sat

We begin with proving NP-completeness of Monotone NAE-3-Sat-E4 in Section 3.1. This result, in turn, is then used in Section 3.2 to derive the even stronger result that Monotone NAE-3-Sat-E4 remains NP-complete even when restricted to linear formulas. Since membership in NP is obvious, the NP-completeness proofs in this paper reduce to showing NP-hardness.

3.1. Hardness of Monotone NAE-3-Sat-E4

For our first result, NP-completeness of Monotone NAE-3-Sat-E4, we give two different proofs. The reason for doing so is that the first proof has the advantage of being relatively simple, while featuring the drawback of using an auxiliary gadget to increase the number of variable appearances; the latter is avoided in the second proof.

Theorem 1. *Monotone NAE-3-Sat-E4 is NP-complete.*

Proof 1 of Theorem 1. We show NP-hardness of Monotone NAE-3-Sat-E4 by reduction from Monotone NAE-3-Sat (see, e.g., Porschen et al. [8, Theorem 3] for a proof that the latter problem is NP-complete). Let $\mathcal{I} = (V, C)$ be an instance of Monotone NAE-3-Sat. Let $n := |V|$ denote the number of variables, $m := |C|$ the number of clauses and recall that $a(x_i)$ denotes the number of appearances of a variable $x_i \in V$ in the formula C. Further, let the set of variables be given as $V := \{x_1, x_2, \dots, x_n\}$.

For each variable x_i, we replace the jth appearance with a new variable $x_{i,j}$ and introduce the clauses

$$\mathrm{EQ}(x_{i,a(x_i)}, x_{i,1}) \cup \bigcup_{j=1}^{a(x_i)-1} \mathrm{EQ}(x_{i,j}, x_{i,j+1}),$$

where $\mathrm{EQ}(x_{i,s}, x_{i,t})$ is an *equality gadget* (a set of clauses) enforcing that $x_{i,s}$ and $x_{i,t}$ are mapped to the same truth value by any satisfying assignment. More precisely, a truth assignment β for $\{x_s, x_t\}$ can be extended to a truth assignment β' for all variables appearing in $\mathrm{EQ}(x_{i,s}, x_{i,t})$ that nae-satisfies $\mathrm{EQ}(x_{i,s}, x_{i,t})$ if and only if $\beta(x_{i,s}) = \beta(x_{i,t})$. We construct this gadget in two steps. First, we define a *non-equality gadget* enforcing that two variables are set to different truth values in any nae-satisfying truth assignment.

Consider the set of clauses

$$\mathrm{NE}(x, y) := \{\{x, y, a\}, \{x, y, b\}, \{a, b, u\}, \{a, b, v\}, \{a, b, w\}, \{u, v, w\}\},$$

[2] See Porschen et al. [8] for more details on the connection between Monotone Not-All-Equal Satisfiability and Hypergraph 2-Colorability (resp. Set Splitting) and related hardness results in the linear setting.

where a, b, u, v, w are new variables not appearing anywhere else, e.g., the clause sets $\mathrm{NE}(x, y)$ and $\mathrm{NE}(y, z)$ do not have any common variables except of y. In order to nae-satisfy the last clause in $\mathrm{NE}(x, y)$, at least one of u, v, w is set to true and at least one of them is set to false. Hence, by construction of the three preceding clauses, a and b are set to different truth values. Then, due to the first two clauses x and y are set to different truth values in any truth assignment that nae-satisfies $\mathrm{NE}(x, y)$. Now, the equality gadget is defined as

$$\mathrm{EQ}(x, y) := \mathrm{NE}(p, q) \cup \mathrm{NE}(p, r) \cup \{\{x, q, r\}, \{y, q, r\}\},$$

where p, q and r are new variables not appearing anywhere else. Note that by construction of the two non-equality gadgets, q and r are set to the same truth value. Hence, due to the two last clauses, x and y are set to the same truth value. By symmetry of nae-satisfying truth assignments, we can, thus, extend any truth assignment β for $\{x, y\}$ with $\beta(x) = \beta(y)$ to a truth assignment that nae-satisfies $\mathrm{EQ}(x, y)$.

Note that each variable $x_{i,j}$ appears in two equality gadgets, once in each gadget, and in exactly one clause of the original instance. Moreover, each introduced variable appears in at most four clauses. With the following gadget, we can increase the appearances of a variable by one, while only introducing variables with exactly four appearances. Let

$$\mathrm{P1}(x) := \{\{x, a, b\}, \{a, c, d\}, \{a, b, e\}, \{a, d, e\}, \{b, c, d\}, \{b, c, e\}, \{c, d, e\}\},$$

where a, b, c, d, e are new variables not appearing anywhere else. Note that these clauses are satisfiable independently of the truth value of x by setting each variable in $\{a, c, e\}$ true and each variable in $\{b, d\}$ false. Now, we can use this gadget to increase the appearances of each variable until it appears exactly four times. The number of introduced variables and clauses is clearly polynomial and the verification of the reduction is straightforward. □

We now present a second proof for Theorem 1 which reduces from the more general NAE-3-SAT* problem and does not require a separate gadget to increase the number of variable appearances. The proof will make use of the two following lemmata.

Lemma 1. *Let* $\mathrm{NE}(x, y)$ *be the following set of clauses, where* $V_{aux} = \{a, b, \ldots, f\}$ *are new variables.*

1. $\{x, a, b\}$
2. $\{y, c, d\}$
3. $\{y, e, f\}$
4. $\{c, e, f\}$
5. $\{b, c, e\}$
6. $\{a, c, f\}$
7. $\{a, d, e\}$
8. $\{a, b, d\}$
9. $\{b, d, f\}$

Then, a truth assignment β *for* $\{x, y\}$ *can be extended to a truth assignment* β' *for* $\{x, y\} \cup V_{aux}$ *that nae-satisfies* $\mathrm{NE}(x, y)$ *if and only if* $\beta(x) \neq \beta(y)$.

Proof. First, we can nae-satisfy all clauses in $\mathrm{NE}(x, y)$ by setting all variables in $\{x, c, d, e\}$ true (resp. false) and all variables in $\{y, a, b, f\}$ false (resp. true). Second, assume towards a contradiction that there is an nae-satisfying assignment β with $\beta(x) = \beta(y) = T$. We consider all four possible assignments of the variables a and c to truth values.

Case $\beta(a) = F$, $\beta(c) = F$: By clause 6 we have $\beta(f) = T$. Then, by clause 3 we have $\beta(e) = F$. By clauses 5 and 7 we have $\beta(b) = T$ and $\beta(d) = T$, respectively. Hence, all literals in clause 9 evaluate to true, i.e., β does not nae-satisfy clause 9.

Case $\beta(a) = F$, $\beta(c) = T$: By clause 2 we have $\beta(d) = F$. Then, by clauses 7 and 8 we have $\beta(e) = T$ and $\beta(b) = T$, respectively. By clause 5 we have $\beta(b) = F$. Thus, we have $\beta(b) \neq \beta(b)$, a contradiction.

Case $\beta(a) = T$, $\beta(c) = F$: By clause 1 we have $\beta(b) = F$. Then, by clause 5 we have $\beta(e) = T$. By clause 7 we have $\beta(d) = F$. Then, by clause 9 we have $\beta(f) = T$. Hence, all literals in clause 3 evaluate to true, i.e., β does not nae-satisfy clause 3.

Case $\beta(a) = T$, $\beta(c) = T$: By clauses 1, 2 and 6 we have $\beta(b) = F$, $\beta(d) = F$ and $\beta(f) = F$, respectively. Hence, all literals in clause 9 evaluate to false, i.e., β does not nae-satisfy clause 9.

By symmetry of nae-satisfying truth assignments, there is no nae-satisfying assignment β with $\beta(x) = \beta(y) = F$. □

Lemma 2. *Let* $\mathrm{EQ}(x, y)$ *be the following set of clauses, where* $V_{aux} = \{a, b, \ldots, i\}$ *are new variables.*

1. $\{x, a, b\}$
2. $\{y, c, d\}$
3. $\{y, e, f\}$
4. $\{a, c, g\}$
5. $\{a, e, d\}$
6. $\{a, h, i\}$
7. $\{b, e, h\}$
8. $\{b, f, h\}$
9. $\{b, g, i\}$
10. $\{c, e, i\}$
11. $\{c, f, g\}$
12. $\{d, g, h\}$
13. $\{d, f, i\}$

Then, a truth assignment β *for* $\{x, y\}$ *can be extended to a truth assignment* β' *for* $\{x, y\} \cup V_{aux}$ *that nae-satisfies* $\mathrm{EQ}(x, y)$ *if and only if* $\beta(x) = \beta(y)$.

Proof. First, we can nae-satisfy all clauses in EQ(x, y) by setting all variables in $\{x, y, e, g, h, i\}$ true (resp. false) and all variables in $\{a, b, c, d, f\}$ false (resp. true). Hence, we can extend a truth assignment β for $\{x, y\}$ to a truth assignment β' for $\{x, y\} \cup V_{\text{aux}}$ that nae-satisfies EQ(x, y) if $\beta(x) = \beta(y)$.

Second, assume towards a contradiction that $\beta(x) = F$ and $\beta(y) = T$ for a truth assignment β that nae-satisfies EQ(x, y).

Case $\beta(a) = F,\ \beta(c) = F$: By clauses 1 and 4 we have $\beta(b) = T$ and $\beta(g) = T$, respectively. Then, by clause 9 we have $\beta(i) = F$. By clauses 6 and 10 we have $\beta(h) = T$ and $\beta(e) = T$, respectively. But then all literals in clause 7 evaluate to true, i.e., β does not nae-satisfy clause 7, a contradiction to our assumption.

Case $\beta(a) = F,\ \beta(c) = T$: By clauses 1 and 2 we have $\beta(b) = T$ and $\beta(d) = F$, respectively. Then, by clause 5 we have $\beta(e) = T$. By clauses 7 and 10 we have $\beta(h) = F$ and $\beta(i) = F$, respectively. Therewith all literals in clause 6 evaluate to false and hence β does not nae-satisfy clause 6, a contradiction.

Case $\beta(a) = T,\ \beta(c) = F$:

- Case $\beta(e) = F$: By clause 10 we have $\beta(i) = T$. Then, by clause 6 we have $\beta(h) = F$. By clause 7 we have $\beta(b) = T$. Then, by clause 9 we have $\beta(g) = F$. By clause 12 we have $\beta(d) = T$. Then, by clause 13 we have $\beta(f) = F$. This, however, implies that β does not nae-satisfy clause 11, a contradiction.
- Case $\beta(e) = T$: By clauses 3 and 5 we have $\beta(f) = F$ and $\beta(d) = F$, respectively. Then, by clauses 13 and 11 we have $\beta(i) = T$ and $\beta(g) = T$, respectively. By clauses 9 and 6 we have $\beta(b) = F$ and $\beta(h) = F$, respectively. Hence, β does not nae-satisfy clause 8, in contradiction with our assumption.

Case $\beta(a) = T,\ \beta(c) = T$: By clauses 2 and 4 we have $\beta(d) = F$ and $\beta(g) = F$, respectively. Then, by clause 12 we have $\beta(h) = T$. By clause 6 we have $\beta(i) = F$. Then, by clauses 13 and 9 we have $\beta(f) = T$ and $\beta(b) = T$, respectively. Thus, β does not nae-satisfy clause 8, a contradiction.

Hence, there is no truth assignment β with $\beta(x) = F$ and $\beta(y) = T$ that nae-satisfies EQ(x, y). By symmetry of nae-satisfying truth assignments, there is also no truth assignment β with $\beta(x) = T$ and $\beta(y) = F$ that can be extended to a truth assignment that nae-satisfies EQ(x, y). □

Now, we have the tools we need for our second proof of Theorem 1.

Proof 2 of Theorem 1. We show NP-hardness by reduction from NAE-3-SAT*. NP-completeness of NAE-3-SAT* was established by Schaefer [9]. Let $\mathcal{I} = (V, C)$ be an instance of NAE-3-SAT*. Let $n := |X|$ denote the number of variables, $m := |C|$ the number of clauses and recall that $a(x_i)$ denotes the number of appearances of a variable $x_i \in V$ in the formula C. Further, let the set of variables be given as $V := \{x_1, x_2, \ldots, x_n\}$.

For each variable $x_i \in V$, we replace the jth appearance with a new variable $x_{i,j}$, such that $x_{i,j}$ is unnegated for $j \leq u(x_i)$ and negated for $j > u(x_i)$, where $u(x_i) \in \{0, 1, \ldots, a(x_i)\}$ is the number of unnegated appearances of x_i in C. First, we make sure that, for each $x_i \in V$, all variables in $\{x_{i,j} \mid j \leq u(x_i)\}$ are mapped to the same truth value in any nae-satisfying assignment by introducing the clauses

$$\bigcup_{i=1}^{n} \bigcup_{j=1}^{u(x_i)-1} EQ(x_{i,j}, x_{i,j+1}),$$

where $EQ(x_{i,j}, x_{i,j+1})$ is the equality gadget defined in Lemma 2. Second, we do the same for the variables in $\{x_{i,j} \mid j > u(x_i)\}$, i.e., we introduce the clauses

$$\bigcup_{i=1}^{n} \bigcup_{j=u(x_i)+1}^{a(x_i)-1} EQ(x_{i,j}, x_{i,j+1}).$$

Now, we delete all negations and make sure that $x_{i,j}$ and $x_{i,j'}$ with $j \leq u(x_i)$ and $j' > u(x_i)$ are to be mapped to different truth values by introducing

$$\bigcup_{\substack{1 \leq i \leq n \\ u(x_i) \notin \{0, a(x_i)\}}} NE(x_{i,u(x_i)}, x_{i,u(x_i)+1}),$$

where $NE(x_{i,j}, x_{i,j'})$ is the non-equality gadget defined in Lemma 1. Next, in order to get the right number of variable appearances, we introduce for each x_i that appears only negated or only unnegated the clauses $EQ(x_{i,a(x_i)}, x_{i,1})$ and for each variable $x_{i'}$ that appears both negated and unnegated we introduce the clauses $NE(x_{i',a(x_{i'})}, x_{i',1})$. Thus, for each variable x_i we get the ring structure

$$EQ(x_{i,1}, x_{i,2}) \cup EQ(x_{i,2}, x_{i,3}) \cup \ldots \cup EQ(x_{i,a(x_i)-1}, x_{i,a(x_i)}) \cup EQ(x_{i,a(x_i)}, x_{i,1}),$$

if x_i appears only negated or only unnegated, and we get the ring structure

$$\bigcup_{j=1}^{u(x_i)-1} EQ(x_{i,j}, x_{i,j+1}) \cup NE(x_{i,u(x_i)}, x_{i,u(x_i)+1}) \cup \bigcup_{j=u(x_i)+1}^{a(x_i)-1} EQ(x_{i,j}, x_{i,j+1}) \cup NE(x_{i,a(x_i)}, x_{i,1}),$$

otherwise. It is straightforward to verify that the resulting instance is nae-satisfiable if and only if $\mathcal{I}$ is nae-satisfiable.

Note that for $a(x_i) > 1$ each variable $x_{i,j}$ appears exactly once as the first argument and exactly once as the second argument of a gadget (it is not important of which gadget) yielding three appearances of $x_{i,j}$. Observe that in the case $a(x_i) = 1$ we introduce $EQ(x_{i,a(x_i)}, x_{i,1}) = EQ(x_{i,1}, x_{i,1})$ only, hence yielding three appearances of $x_{i,1}$ by means of that gadget. Since each $x_{i,j}$ also replaces exactly one appearance of x_i in the clause set C, we get exactly four appearances of $x_{i,j}$ in the constructed instance. All other variables introduced by the gadgets (variables of the gadgets that are not arguments are always newly created, i.e., these variables are *not* shared between gadgets) appear exactly four times by construction. Hence, the resulting instance is indeed an instance of Monotone NAE-3-Sat-E4. □

3.2. Hardness of Monotone NAE-3-Sat-E4 for linear formulas

In this section, we strengthen our result from the previous section by showing that Monotone NAE-3-Sat-E4 remains NP-complete even when restricted to linear formulas. We begin by stating the following lemma.

Lemma 3. *Let* EQ(x, y, z, u) *be the following set of clauses, where* $V_{aux} = \{a, b, \ldots, f\}$ *are new variables.*

1. $\{x, a, e\}$
2. $\{x, b, d\}$
3. $\{x, c, f\}$
4. $\{y, a, b\}$
5. $\{y, c, e\}$
6. $\{y, d, f\}$
7. $\{z, a, f\}$
8. $\{z, c, d\}$
9. $\{z, u, b\}$
10. $\{u, a, c\}$
11. $\{u, d, e\}$
12. $\{b, e, f\}$

Then, a truth assignment β *for* $\{x, y, z, u\}$ *can be extended to a truth assignment* β' *for* $\{x, y, z, u\} \cup V_{aux}$ *that nae-satisfies* EQ(x, y, z, u) *if and only if* $\beta(x) = \beta(y) = \beta(z) = \beta(u)$. *In addition, the above set of clauses is linear if the variables* x, y, z, u *are pairwise distinct.*

Proof. First, by setting all variables in $\{x, y, z, u, e\}$ true and all variables in $\{a, b, c, d, f\}$ false we can nae-satisfy all clauses in EQ(x, y, z, u). Further, by flipping the truth values for these sets, we obtain a nae-satisfying truth assignment where x, y, z and u are all set false. Second, we show that $\beta(x) = \beta(y) = \beta(z) = \beta(u)$ for each assignment β that nae-satisfies EQ(x, y, z, u). Let β be a nae-satisfying assignment. Assume towards a contradiction that $\beta(x) \neq \beta(y)$. By symmetry of nae-satisfying truth assignments, we may assume that $\beta(x) = F$ and $\beta(y) = T$. Then, β nae-satisfies the first six clauses if and only if β satisfies (not necessarily nae-satisfies) the following set of 2-clauses:

$$\{\{a, e\}, \{b, d\}, \{c, f\}, \{\bar{a}, \bar{b}\}, \{\bar{c}, \bar{e}\}, \{\bar{d}, \bar{f}\}\}$$

Now, using resolution we obtain clauses $\{\bar{b}, e\}, \{\bar{e}, f\}, \{\bar{f}, b\}$ which are satisfied if β satisfies the above set of 2-clauses. Since the inferred clauses form a cyclic implication chain, we have $\beta(b) = \beta(e) = \beta(f)$. Thus, clause 12 is not nae-satisfied which is a contradiction to the assumption that β nae-satisfies EQ(x, y, z, u). Hence, $\beta(x) = \beta(y)$ and, by symmetry of nae-satisfying truth assignments, we may assume that $\beta(x) = \beta(y) = F$. If $\beta(z) = \beta(u) = F$, we are done. Let us consider the three remaining cases:

- If $\beta(z) = \beta(u) = T$, then $\beta(b) = F$ by clause 9. By clauses 2 and 4, we have $\beta(d) = T$ and $\beta(a) = T$, respectively. Then, by clause 7 and 11, we have $\beta(f) = F$ and $\beta(e) = F$, respectively. Thus, clause 12 is not nae-satisfied. Again, this is a contradiction to the assumption that β nae-satisfies EQ(x, y, z, u).
- If $\beta(z) = T$ and $\beta(u) = F$, then β nae-satisfies clauses 2, 6, 7, 8, 10 and 11 if and only if β satisfies (again, not necessarily nae-satisfies) the following set of 2-clauses:

  $$\{\{b, d\}, \{d, f\}, \{\bar{a}, \bar{f}\}, \{\bar{c}, \bar{d}\}, \{a, c\}, \{d, e\}\}.$$

  Using resolution, we obtain clauses $\{\{\bar{f}, c\}, \{\bar{c}, f\}, \{\bar{c}, b\}, \{\bar{c}, e\}\}$ which are satisfied by β since β satisfies the above set of 2-clauses. Now, by the first two inferred clauses and clause 3 (recall that $\beta(x) = F$), we have $\beta(c) = \beta(f) = T$. Then, by the latter two inferred clauses, we have $\beta(b) = \beta(e) = T$. Thus, clause 12 is not nae-satisfied, a contradiction.
- If $\beta(z) = F$ and $\beta(u) = T$, then β nae-satisfies clauses 1, 2, 6, 8, 10 and 11 if and only if β satisfies the following set of 2-clauses:

  $$\{\{a, e\}, \{b, d\}, \{d, f\}, \{c, d\}, \{\bar{a}, \bar{c}\}, \{\bar{d}, \bar{e}\}\}.$$

  Using resolution, we obtain clauses $\{\{\bar{c}, e\}, \{\bar{e}, c\}, \{\bar{e}, b\}, \{\bar{e}, f\}\}$ which leads to a contradiction in a similar way as in the previous case (i.e., β does not nae-satisfy clause 12).

Hence, we conclude that $\beta(x) = \beta(y) = \beta(z) = \beta(u)$ for each assignment β that nae-satisfies EQ(x, y, z, u). A truth assignment β for $\{x, y, z, u\}$ can, thus, be extended to a truth assignment β' for $\{x, y, z, u\} \cup V_{\text{aux}}$ that nae-satisfies EQ(x, y, z, u) if and only if $\beta(x) = \beta(y) = \beta(z) = \beta(u)$.

By considering each pair of distinct clauses in EQ(x, y, z, u) it is easy to verify that the set of clauses is linear if the variables x, y, z, u are pairwise distinct. □

Theorem 2. Monotone NAE-3-Sat-E4 *is NP-complete for linear formulas.*

Proof. We show NP-hardness by reduction from Monotone NAE-3-Sat-E4, for which NP-hardness was established in Theorem 1. Let $\mathcal{I} = (V, C)$ be an instance of Monotone NAE-3-Sat-E4. Let $n := |V|$ denote the number of variables, $m := |C|$ the number of clauses and let the set of variables be given as $V := \{x_1, x_2, \ldots, x_n\}$. For each variable $x_i \in V$, we replace the jth appearance with a new variable $x_{i,j}$. Then, we make sure that, for each $x_i \in V$, all variables in $\{x_{i,1}, x_{i,2}, x_{i,3}, x_{i,4}\}$ are mapped to the same truth value in any nae-satisfying truth assignment by introducing the clauses

$$\bigcup_{i=1}^{n} \text{EQ}(x_{i,1}, x_{i,2}, x_{i,3}, x_{i,4}),$$

where EQ$(x_{i,1}, x_{i,2}, x_{i,3}, x_{i,4})$ is the equality gadget defined in Lemma 3. The gadgets do not share any variables, i.e., each instance of the equality gadget has its own newly created auxiliary variables. Note that each variable still appears exactly four times, once in the original clause set and three times in an equality gadget. Further, since the variables $x_{i,1}, x_{i,2}, x_{i,3}, x_{i,4}$ are pairwise distinct, the subformulas defined by the equality gadgets are linear (see Lemma 3). Observe that the clauses of the original instance are pairwise disjoint after the variable replacement and each of these clauses shares at most one variable with any clause introduced by the gadgets. Note that each clause, except clause 9, in the ith instance of the equality gadget contains at most one variable that appears in the original clause set, i.e., at most one variable $x_{i,j}$ with $1 \leq i \leq n$ and $1 \leq j \leq 4$. Even though clause 9 (see the clause set introduced in Lemma 3) contains two variables $x_{i,3}$ and $x_{i,4}$ that appear outside the gadget, there is no other clause that contains both of them (otherwise some clause of the given instance of Monotone NAE-3-Sat-E4 contains the variable x_i twice, a contradiction). Hence, the constructed formula is linear. By Lemma 3 it follows that the constructed instance is nae-satisfiable if and only if $\mathcal{I}$ is nae-satisfiable. □

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] A. Darmann, J. Döcker, On simplified NP-complete variants of Not-All-Equal 3-SAT and 3-SAT, arXiv:1908.04198, 2019.
[2] A. Dehghan, M. Sadeghi, A. Ahadi, On the complexity of deciding whether the regular number is at most two, Graphs Comb. 31 (5) (Sep 2015) 1359–1365.
[3] A. Horbach, T. Bartsch, D. Briskorn, Using a SAT-solver to schedule sports leagues, J. Sched. 15 (1) (2012) 117–125.
[4] M. Karpinski, K. Piecuch, On vertex coloring without monochromatic triangles, in: F.V. Fomin, V.V. Podolskii (Eds.), Computer Science - Theory and Applications - Proceedings of the 13th International Computer Science Symposium in Russia (CSR'18), in: Lecture Notes in Computer Science, vol. 10846, Springer, 2018, pp. 220–231.
[5] H. Kautz, B. Selman, Pushing the envelope: planning, propositional logic, and stochastic search, in: Proceedings of the 13th National Conference on Artificial Intelligence (AAAI'96), AAAI Press, 1996, pp. 1194–1201.
[6] G. Nam, K.A. Sakallah, R.A. Rutenbar, Satisfiability-based layout revisited: detailed routing of complex FPGAs via search-based boolean SAT, in: Proceedings of the ACM/SIGDA 7th International Symposium on Field Programmable Gate Arrays (FPGA '99), ACM, New York, NY, USA, 1999, pp. 167–175.
[7] S. Porschen, B. Randerath, E. Speckenmeyer, Linear time algorithms for some not-all-equal satisfiability problems, in: Theory and Applications of Satisfiability Testing – SAT 2004, Springer, 2004, pp. 256–257.
[8] S. Porschen, T. Schmidt, E. Speckenmeyer, A. Wotzlaw, XSAT and NAE-SAT of linear CNF classes, Discrete Appl. Math. 167 (2014) 1–14.
[9] T.J. Schaefer, The complexity of satisfiability problems, in: Proceedings of the 10th Annual ACM Symposium on Theory of Computing, ACM, 1978, pp. 216–226.

1.8 Placing quantified variants of 3-SAT and Not-All-Equal 3-SAT in the polynomial hierarchy

The following paper [DDLS20] is also available online at the following URL: `https://doi.org/10.1016/j.tcs.2020.04.003`.

Theoretical Computer Science 822 (2020) 72–91

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Placing quantified variants of 3-SAT and Not-All-Equal 3-SAT in the polynomial hierarchy

Janosch Döcker [a,*], Britta Dorn [a], Simone Linz [b], Charles Semple [c]

[a] Department of Computer Science, University of Tübingen, Tübingen, Germany
[b] School of Computer Science, University of Auckland, Auckland, New Zealand
[c] School of Mathematics and Statistics, University of Canterbury, Christchurch, New Zealand

ARTICLE INFO

Article history:
Received 14 August 2019
Received in revised form 26 December 2019
Accepted 7 April 2020
Available online 16 April 2020
Communicated by V.Th. Paschos

Keywords:
3-Sat
Not-All-Equal 3-Sat
Quantified satisfiability
Polynomial hierarchy
Bounded variable appearances
Computational complexity

ABSTRACT

The complexity of variants of 3-SAT and Not-All-Equal 3-SAT is well studied. However, in contrast, very little is known about the complexity of the problems' quantified counterparts. In the first part of this paper, we show that ∀∃ 3-SAT is Π_2^P-complete even if (1) each variable appears exactly twice unnegated and exactly twice negated, (2) each clause is a disjunction of exactly three distinct variables, and (3) the number of universal variables is equal to the number of existential variables. Furthermore, we show that the problem remains Π_2^P-complete if (1a) each universal variable appears exactly once unnegated and exactly once negated, (1b) each existential variable appears exactly twice unnegated and exactly twice negated, and (2) and (3) remain unchanged. On the other hand, the problem becomes NP-complete for certain variants in which each universal variable appears exactly once. In the second part of the paper, we establish Π_2^P-completeness for ∀∃ Not-All-Equal 3-SAT even if (1′) the Boolean formula is linear and monotone, (2′) each universal variable appears exactly once and each existential variable appears exactly three times, and (3′) each clause is a disjunction of exactly three distinct variables that contains at most one universal variable. On the positive side, we uncover variants of ∀∃ Not-All-Equal 3-SAT that are co-NP-complete or solvable in polynomial time.

1. Introduction

The Boolean satisfiability problem SAT plays a major role in the theory of NP-completeness. It was the first problem shown to be complete for the class NP (Cook's Theorem [3]) and many NP-hardness results are established by using this problem, or restricted variants thereof, as a starting point for polynomial-time reductions. Restricted variants of a problem that remain NP-complete are particularly useful as they allow for the possibility of simpler proofs and stronger results.

The most prominent NP-complete variant of the Boolean satisfiability problem is 3-SAT. Here we are given a conjunction of clauses such that each clause contains exactly three literals, where a literal is a propositional variable or its negation. An instance of 3-SAT is a yes-instance if there is a truth assignment to the propositional variables[1] such that at least one

* Corresponding author.
E-mail addresses: janosch.doecker@uni-tuebingen.de (J. Döcker), britta.dorn@uni-tuebingen.de (B. Dorn), s.linz@auckland.ac.nz (S. Linz), charles.semple@canterbury.ac.nz (C. Semple).

[1] From now on, we simply say variable instead of propositional variable since all variables used in the paper take only values representing true and false.

https://doi.org/10.1016/j.tcs.2020.04.003
0304-3975/

literal of each clause evaluates to true. Interestingly, even within 3-SAT, we can restrict the problem further. For example, for instances of 3-SAT in which each clause contains exactly three distinct variables, Tovey [17, Theorem 2.3] proved that 3-SAT remains NP-complete if each variable appears in at most four clauses. Furthermore, this result also holds if each variable appears exactly twice unnegated and exactly twice negated [1, Theorem 1]. On the other hand, the problem becomes trivial, i.e., the answer is always yes, if each variable appears at most three times [17, Theorem 2.4].

A popular NP-complete variant of 3-SAT called Not-All-Equal 3-SAT (NAE-3-SAT) asks whether we can assign truth values to the variables such that at least one, but not all, of the literals of each clause evaluate to true. Schaefer [15] first established NP-completeness of NAE-3-SAT in the setting where each clause contains at most three literals. Subsequently, Karpinski and Piecuch [9,10] showed that NAE-3-SAT is NP-complete if each variable appears at most four times. Furthermore, Porschen et al. [12, Theorem 3] showed that NAE-3-SAT remains NP-complete if (i) each literal appears at most once in any clause, and (ii) the input formula is *linear* and *monotone*, that is, each pair of distinct clauses share at most one variable and no clause contains a literal that is the negation of some variable. Following on from this last result, Darmann and Döcker [5] showed recently that NAE-3-SAT remains NP-complete if, in addition to (i) and (ii), each variable appears exactly four times. By contrast, if a monotone conjunction of clauses has the property that each variable appears at most three times, NAE-3-SAT can be decided in linear time [11, Theorem 4, p. 186].

In this paper, we consider generalized variants of 3-SAT and NAE-3-SAT, namely ∀∃ 3-SAT and ∀∃ NAE-3-SAT, respectively. Briefly, ∀∃ 3-SAT is a quantified variant of 3-SAT, where each variable is either *universal* or *existential*. The decision problem ∀∃ 3-SAT asks if, for every assignment of truth values to the universal variables, there exists an assignment of truth values to the existential variables such that at least one literal of each clause evaluates to true. Observe that, if an instance of ∀∃ 3-SAT does not contain a universal variable, then this instance reduces to an instance of 3-SAT. Analogously, we can think of ∀∃ NAE-3-SAT as a generalized variant of NAE-3-SAT. Formal definitions of both problems are given in the next section.

Stockmeyer [16] and Dahlhaus et al. [6] showed, respectively, that ∀∃ 3-SAT and ∀∃ NAE-3-SAT are complete for the second level of the polynomial hierarchy or, more precisely, complete for the class Π_2^P. In this paper, we establish Π_2^P-completeness for restricted variants of these two quantified problems. In particular, we show that ∀∃ 3-SAT is Π_2^P-complete if each universal variable appears exactly once unnegated and exactly once negated, and each existential variable appears exactly twice unnegated and exactly once negated or each existential variable appears exactly once unnegated and exactly twice negated. Although we do not consider approximation aspects in this paper, by way of comparison, Haviv et al. [8] showed that approximating a particular optimization version of ∀∃ 3-SAT is Π_2^P-hard even if each universal variable appears at most twice and each existential variable appears at most three times. Whether optimization versions of the Π_2^P-complete problems presented in this paper are Π_2^P-hard to approximate is a question that we leave for future research. Furthermore, we establish Π_2^P-completeness for ∀∃ 3-SAT if each universal variable appears exactly s_1 times unnegated and exactly s_2 times negated, each existential variable appears exactly t_1 times unnegated and exactly t_2 times negated, and the following three properties are satisfied: (i) $s_1 = s_2$, (ii) $s_1 \in \{1, 2\}$, and (iii) $t_1 = t_2 = 2$. These latter completeness results hold even if each clause is a disjunction of exactly three distinct variables and the number of universal and existential variables is *balanced*, that is, the number of universal and existential variables are the same.

Turning to ∀∃ NAE-3-SAT, we show that the problem remains Π_2^P-complete if each universal variable appears exactly once, each clause contains at most one universal variable, each existential variable appears exactly three times, and the conjunction of clauses is both linear and monotone. Interestingly, while one appearance of each universal variable is enough to obtain a Π_2^P-hardness result in this setting, the same is not true for ∀∃ 3-SAT unless the polynomial hierarchy collapses [8, p. 55].

The remainder of the paper is organized as follows. The next section introduces notation and terminology, and formally states three variants of ∀∃ 3-SAT and ∀∃ Not-All-Equal 3-SAT that are the main focus of this paper. Section 3 (resp. Section 4) investigates the computational complexity of ∀∃ 3-SAT (resp. ∀∃ Not-All-Equal 3-SAT). Both sections start with a subsection on enforcers that are needed for the subsequent hardness proofs and that we expect to be of independent interest in future work.

2. Preliminaries

This section introduces notation and terminology that is used throughout the paper.

Let $V = \{x_1, x_2, \ldots, x_n\}$ be a set of variables. A *literal* is a variable or its negation. We denote the set $\{x_i, \bar{x}_i : i \in \{1, 2, \ldots, n\}\}$ of all literals that correspond to elements in V by $\mathcal{L}_V$. A *clause* is a disjunction of a subset of $\mathcal{L}_V$. If a clause contains exactly k distinct literals for $k \geq 1$, then it is called a *k-clause*. For example, $(x_1 \vee \bar{x}_2 \vee x_4)$ is a 3-clause. A *Boolean formula in conjunctive normal form* (CNF) is a conjunction of clauses, i.e., an expression of the form $\varphi = \bigwedge_{j=1}^{m} C_j$, where C_j is a clause for all j. In what follows, we refer to a Boolean formula in conjunctive normal form simply as a *Boolean formula*. Now, let φ be a Boolean formula. We say that φ is *linear* if any pair of distinct clauses share at most one variable and that it is *monotone* if no clause contains an element in $\{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n\}$. Furthermore, if each clause contains at most k literals, it is said to be in *k-CNF*. For each variable $x_i \in V$, we denote the number of appearances of x_i in φ plus the number of appearances of $\bar{x}_i$ in φ by $a(x_i)$. A *variable assignment* or, equivalently, *truth assignment* for V is a mapping $\beta\colon V \to \{T, F\}$, where T represents the truth value True and F represents the truth value False. A truth assignment β *satisfies* φ if at least one literal of each clause evaluates to T under β. If there exists a truth assignment that satisfies φ, we say that φ is *satisfiable*. For a

truth assignment β that satisfies φ and has the additional property that at least one literal of each clause evaluates to F, we say that β *nae-satisfies* φ. Lastly, let V and V' be two disjoint sets of variables, let β be a truth assignment for V, and let β' be a truth assignment for $V \cup V'$. We say that β' *extends* β (or, alternatively, that β extends to $V \cup V'$) if $\beta(x_i) = \beta'(x_i)$ for each $x_i \in V$.

A *quantified Boolean formula* Φ over a set $V = \{x_1, x_2, \ldots, x_n\}$ of variables is a formula of the form

$$\forall x_1 \cdots \forall x_p \exists x_{p+1} \cdots \exists x_n \bigwedge_{j=1}^{m} C_j.$$

The variables $x_1, x_2, \ldots, x_p$ are *universal* variables of Φ and the variables $x_{p+1}, x_{p+2}, \ldots, x_n$ are *existential* variables of Φ. Furthermore, for variables $x_i, x_{i+1}, \ldots, x_{i'}$ with $1 \leq i < i' \leq p$ and $x_{i''}, x_{i''+1}, \ldots, x_{i'''}$ with $p+1 \leq i'' < i''' \leq n$, we define

$$\forall X_i^{i'} := \forall x_i \cdots \forall x_{i'} \quad \text{and} \quad \exists X_{i''}^{i'''} := \exists x_{i''} \cdots \exists x_{i'''},$$

respectively and, similarly,

$$X_i^{i'} := \{x_i, \ldots, x_{i'}\} \quad \text{and} \quad X_{i''}^{i'''} := \{x_{i''}, \ldots, x_{i'''}\},$$

respectively.

We next introduce notation that transforms a Boolean formula φ into another such formula. Specifically, we use $\varphi[x \mapsto y]$ to denote the Boolean formula obtained from φ by replacing each appearance of variable x with variable y (i.e., replace x with y and replace $\overline{x}$ with $\overline{y}$). For pairwise distinct pairs $(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)$ of variables, we use $\varphi[x_1 \mapsto y_1, \ldots, x_k \mapsto y_k]$ to denote the Boolean formula obtained from φ by simultaneously replacing each appearance of variable x_i by variable y_i for $1 \leq i \leq k$. Since the variables are pairwise distinct, note that this operation is well-defined. Lastly, for a constant $b \in \{T, F\}$, the Boolean formula $\varphi[x \mapsto b]$ is obtained from φ by replacing each appearance of variable x by b.

The polynomial hierarchy. An *oracle* for a complexity class A is a black box that, in constant time, outputs the answer to any given instance of a decision problem contained in A. The *polynomial hierarchy* is a system of nested complexity classes that are defined recursively as follows. Set

$$\Sigma_0^P = \Pi_0^P = \mathrm{P},$$

and define, for all $k \geq 0$,

$$\Sigma_{k+1}^P = \mathrm{NP}^{\Sigma_k^P} \quad \text{and} \quad \Pi_{k+1}^P = \text{co-NP}^{\Sigma_k^P},$$

where a problem is in $\mathrm{NP}^{\Sigma_k^P}$ (resp. $\text{co-NP}^{\Sigma_k^P}$) if we can verify an appropriate certificate of a yes-instance (resp. no-instance) in polynomial time when given access to an oracle for Σ_k^P. By definition, $\Sigma_1^P = \mathrm{NP}$ and $\Pi_1^P = \text{co-NP}$. We say that the classes Σ_k^P and Π_k^P are on the *k-th level* of the polynomial hierarchy.

For all $k \geq 0$, there are complete problems under polynomial-time many-one reductions for Σ_k^P and Π_k^P. However, while, for example, the complexity class Π_2^P generalizes both NP and co-NP, it remains an open question whether $\Sigma_k^P \neq \Sigma_{k+1}^P$ or $\Pi_k^P \neq \Pi_{k+1}^P$ for any $k \geq 0$. For further details of the polynomial hierarchy, we refer the interested reader to Garey and Johnson's book [7], an article by Stockmeyer [16], as well as to the compendium by Schaefer and Umans [14] for a collection of problems that are complete for the second or higher levels of the polynomial hierarchy.

The following two Π_2^P-complete problems are the starting points for the work presented in this paper.

∀∃ 3-SAT
Input. A quantified Boolean formula

$$\forall x_1 \cdots \forall x_p \exists x_{p+1} \cdots \exists x_n \bigwedge_{j=1}^{m} C_j$$

over a set $V = \{x_1, x_2, \ldots, x_n\}$ of variables, where each clause C_j is a disjunction of at most three literals.
Question. For every truth assignment for $\{x_1, x_2, \ldots, x_p\}$, does there exist a truth assignment for $\{x_{p+1}, x_{p+2}, \ldots, x_n\}$ such that each clause of the formula is satisfied?

∀∃ Not-All-Equal 3-SAT (∀∃ NAE-3-SAT)
Input. A quantified Boolean formula

$$\forall x_1 \cdots \forall x_p \exists x_{p+1} \cdots \exists x_n \bigwedge_{j=1}^{m} C_j$$

over a set $V = \{x_1, x_2, \ldots, x_n\}$ of variables, where each clause C_j is a disjunction of at most three literals.
Question. For every truth assignment for $\{x_1, x_2, \ldots, x_p\}$, does there exist a truth assignment for $\{x_{p+1}, x_{p+2}, \ldots, x_n\}$ such that each clause of the formula is nae-satisfied?

Stockmeyer [16], and Eiter and Gottlob [6] established Π_2^P-completeness for ∀∃ 3-SAT and ∀∃ NAE-3-SAT, respectively.

The main focus of this paper are the following three restricted variants of ∀∃ 3-SAT and ∀∃ NAE-3-SAT. For the first two problems, s_1, s_2, t_1, t_2 are non-negative integers.

Balanced ∀∃ 3-SAT-(s_1, s_2, t_1, t_2)
Input. A quantified Boolean formula

$$\forall x_1 \cdots \forall x_p \exists x_{p+1} \cdots \exists x_n \bigwedge_{j=1}^{m} C_j$$

over a set $V = \{x_1, x_2, \ldots, x_n\}$ of variables such that (i) $n = 2p$, (ii) each C_j is a 3-clause that contains three *distinct* variables, and (iii), amongst the clauses, each universal variable appears unnegated exactly s_1 times and negated exactly s_2 times, and each existential variable appears unnegated exactly t_1 times and negated exactly t_2 times.
Question. For every truth assignment for $\{x_1, x_2, \ldots, x_p\}$, does there exist a truth assignment for $\{x_{p+1}, x_{p+2}, \ldots, x_n\}$ such that each clause of the formula is satisfied?

We also consider the decision problem that is obtained from Balanced ∀∃ 3-SAT-(s_1, s_2, t_1, t_2) by omitting property (i) in the statement of the input. We refer to the resulting problem as ∀∃ 3-SAT-(s_1, s_2, t_1, t_2). Lastly, we consider the following problem, where s and t are non-negative integers.

Monotone ∀∃ NAE-3-SAT-(s, t)
Input. A monotone quantified Boolean formula

$$\forall x_1 \cdots \forall x_p \exists x_{p+1} \cdots \exists x_n \bigwedge_{j=1}^{m} C_j$$

over a set $V = \{x_1, x_2, \ldots, x_n\}$ of variables such that (i) each C_j is a 3-clause that contains three *distinct* variables and (ii), amongst the clauses, each universal variable appears exactly s times and each existential variable appears exactly t times.
Question. For every truth assignment for $\{x_1, x_2, \ldots, x_p\}$, does there exist a truth assignment for $\{x_{p+1}, x_{p+2}, \ldots, x_n\}$ such that each clause of the formula is nae-satisfied?

Enforcers. To establish the results of this paper, we will frequently use the concept of enforcers. An *enforcer* (sometimes also called a *gadget*) [1] is a Boolean formula, where the formula itself and each truth assignment that satisfies it has a certain structure. Enforcers are used in polynomial-time reductions to add additional restrictions on how yes-instances can be obtained.

We next detail two unquantified enforcers that were introduced by Berman et al. [1, p. 3] and that lay the foundation for several other enforcers that are new to this paper and will be introduced in the following sections. First, let ℓ_1, ℓ_2 and ℓ_3 be three, not necessarily distinct, literals. Without loss of generality, we may assume that $\ell_1 \in \{x_1, \overline{x}_1\}$, $\ell_2 \in \{x_2, \overline{x}_2\}$, and $\ell_3 \in \{x_3, \overline{x}_3\}$. Now consider the following enforcer to which we refer to as $\mathcal{S}$-*enforcer*:

$$\mathcal{S}(\ell_1, \ell_2, \ell_3) = (\ell_1 \vee \overline{a} \vee b) \wedge (\ell_2 \vee \overline{b} \vee c) \wedge (\ell_3 \vee a \vee \overline{c}) \wedge (a \vee b \vee c) \wedge (\overline{a} \vee \overline{b} \vee \overline{c}),$$

where a, b, and c are new variables such that $\{x_1, x_2, x_3\} \cap \{a, b, c\} = \emptyset$. Let $\beta\colon \{x_1, x_2, x_3, a, b, c\} \to \{T, F\}$ be a truth assignment. The next observation is an immediate consequence from the fact that, if $\beta(\ell_1) = \beta(\ell_2) = \beta(\ell_3) = F$, then, as the first three clauses form a cyclic implication chain which can only be satisfied by setting $\beta(a) = \beta(b) = \beta(c)$, either the fourth or fifth clause is not satisfied.

Observation 2.1. *Consider the Boolean formula $\mathcal{S}(\ell_1, \ell_2, \ell_3)$, where $\ell_i \in \{x_i, \overline{x_i}\}$, and let V be its associated set of variables. A truth assignment β for the variables $\{x_1, x_2, x_3\}$ can be extended to a truth assignment β' for V that satisfies $\mathcal{S}(\ell_1, \ell_2, \ell_3)$ if and only if $\beta(\ell_i) = T$ for some $i \in \{1, 2, 3\}$.*

Remark. We denote the gadget obtained from $\mathcal{S}(\ell_1, \ell_2, \ell_3)$ by viewing x_1 as a universal variable and all other variables in $\{x_2, x_3, a, b, c\}$ as existential variables by $\mathcal{S}_u(\ell_1, \ell_2, \ell_3)$. Depending on the truth value assigned to the universal variable x_1, the existential variables $V - \{x_1\}$ may take different values in a truth assignment, say β, to satisfy the Boolean formula $\mathcal{S}(\ell_1, \ell_2, \ell_3)$. In particular, following Observation 2.1, for every truth assignment for x_1 there exist truth assignments for x_2 and x_3 such that the resulting truth assignment, say β, for $\{x_1, x_2, x_3\}$ can be extended to a truth assignment β' for V that satisfies $\mathcal{S}_u(\ell_1, \ell_2, \ell_3)$ if and only if $\beta(\ell_i) = T$ for some $i \in \{1, 2, 3\}$. Note that the variables in $\{x_1, x_2, x_3\}$ may be shared between different instances of the gadget. Hence, although each individual instance can be satisfied even if $\beta(\ell_1) = F$, it may be impossible to satisfy them all at once.

In what follows, we will use enforcers that are built of several copies of the $\mathcal{S}$-enforcer. In such a case, for each pair of $\mathcal{S}$-enforcer copies, the two 3-element sets of new variables are disjoint.

Again following the constructions from Berman et al. [1], consider a second enforcer:

$$x^{(2)} = \mathcal{S}(x, y, y) \wedge \mathcal{S}(x, \bar{y}, \bar{y}).$$

Note that $x^{(2)}$ is a Boolean formula over eight variables. Moreover, each clause contains three distinct variables since the copies of y and $\bar{y}$ are distributed over different clauses in $\mathcal{S}(x, y, y)$ and $\mathcal{S}(x, \bar{y}, \bar{y})$, respectively. Lastly, each variable, except for x, appears exactly twice unnegated and twice negated in $x^{(2)}$. Now, the next observation follows by construction and Observation 2.1.

Observation 2.2. *Consider the Boolean formula $x^{(2)}$ over a set V of eight variables, where $x, y \in V$. A truth assignment β for $\{x\}$ can be extended to a truth assignment β' for V that satisfies $x^{(2)}$ if and only if $\beta(x) = T$.*

We will use the $\mathcal{S}$-enforcer and $x^{(2)}$ as well as extensions thereof in the proofs of several results established in this paper.

3. Hardness of balanced and unbalanced versions of ∀∃ 3-SAT-(s_1, s_2, t_1, t_2)

3.1. New enforcers

We start by describing three new enforcers, with the first one building upon the enforcers introduced in the previous section. Consider the following gadget:

$$E(x) = \mathcal{S}(x, y, y) \wedge \mathcal{S}(x, \bar{y}, \bar{y}) \wedge \mathcal{S}(\bar{x}, z, \bar{z}) \wedge \mathcal{S}(z, \bar{z}, u) \wedge \mathcal{S}(u, \bar{u}, \bar{u})$$

which is an extended variant of the enforcer $x^{(2)}$. We call x the *enforcer variable* of $E(x)$. Note that every variable in $\{u, y, z\}$ appears exactly twice unnegated and exactly twice negated in $E(x)$, and that x appears exactly twice unnegated and exactly once negated in $E(x)$. Moreover, by construction and Observation 2.2, it follows that $E(x)$ is satisfiable by setting x to be T, and by setting all remaining 18 variables appropriately.

Observation 3.1. *Consider the gadget $E(x)$, and let V be its associated set of variables. A truth assignment β for $\{x\}$ can be extended to a truth assignment β' for V that satisfies $E(x)$ if and only if $\beta(x) = T$.*

We now turn to two quantified enforcers whose purpose is to increase the number of universal variables by one and three, respectively, relative to the number of existential variables. First, let

$$\begin{aligned} Q^1 = &(u \vee v \vee a) \wedge (u \vee v \vee b) \wedge (\overline{u} \vee \overline{v} \vee \overline{a}) \wedge (\overline{u} \vee \overline{v} \vee \overline{b}) \wedge (a \vee \overline{b} \vee r) \wedge (\overline{a} \vee b \vee r) \wedge \\ &(c \vee \overline{d} \vee \overline{r}) \wedge (\overline{c} \vee d \vee \overline{r}) \wedge (w \vee q \vee c) \wedge (w \vee q \vee d) \wedge (\overline{w} \vee \overline{q} \vee \overline{c}) \wedge (\overline{w} \vee \overline{q} \vee \overline{d}), \end{aligned}$$

where u, v, w, q, r are universal variables, and a, b, c, d are existential variables. Observe that each variable of Q^1 appears exactly twice unnegated and exactly twice negated. Second, let

$$Q^3 = (u \vee r \vee a) \wedge (\overline{u} \vee \overline{a} \vee \overline{b}) \wedge (v \vee q \vee b) \wedge (\overline{v} \vee \overline{r} \vee \overline{a}) \wedge (w \vee a \vee b) \wedge (\overline{w} \vee \overline{q} \vee \overline{b}),$$

where u, v, w, q, r are universal variables and a, b are existential variables. Observe that each universal variable of Q^3 appears exactly once unnegated and exactly once negated, and that each existential variable of Q^3 appears exactly twice unnegated and exactly twice negated.

Lemma 3.2. *The quantified Boolean formula*

$$\forall u\, \forall v\, \forall w\, \forall q\, \forall r\, \exists a\, \exists b\, \exists c\, \exists d\ Q^1$$

is a yes-instance of ∀∃ 3-SAT.

Proof. Let $U = \{u, v, w, q, r\}$, and let $E = \{a, b, c, d\}$. Furthermore, let $\beta\colon U \to \{T, F\}$ be a truth assignment for U. We extend β to $\beta'\colon U \cup E \to \{T, F\}$ as follows:

$$\beta'(a) = \beta'(b) = \overline{\beta(u)}, \quad \beta'(c) = \beta'(d) = \overline{\beta(w)}.$$

It is now easy to verify that β' satisfies all clauses. Thus, Q^1 is a yes-instance of ∀∃ 3-SAT. □

Lemma 3.3. *The quantified Boolean formula*

$$\forall u\, \forall v\, \forall w\, \forall q\, \forall r\, \exists a\, \exists b\ Q^3$$

is a yes-instance of ∀∃ 3-SAT.

Proof. Let $U = \{u, v, w, q, r\}$, and let $E = \{a, b\}$. Futhermore, let $\beta\colon U \to \{T, F\}$ be a truth assignment for U. We extend β to a truth assignment $\beta'\colon U \cup E \to \{T, F\}$ for Q^3 as follows:

$$\beta'(a) = \overline{\beta(u)} \wedge \overline{\beta(r)}, \quad \beta'(b) = \overline{\beta(w)} \vee \overline{\beta(q)}.$$

It is now straightforward to check that Q^3 is a yes-instance of ∀∃ 3-SAT. □

3.2. Hardness of Balanced ∀∃ 3-SAT-(s_1, s_2, t_1, t_2)

In this section, we show that Balanced ∀∃ 3-SAT-(s_1, s_2, t_1, t_2) is Π_2^p-complete when

$$(s_1, s_2, t_1, t_2) \in \{(2, 2, 2, 2), (1, 1, 2, 2)\}.$$

To this end, for a clause C, we use $\overline{C}$ to denote the clause obtained from C by replacing each literal with its negation and call $\overline{C}$ the *complement* of C. For example, if $C = (x_1 \vee \bar{x}_2 \vee \bar{x}_3)$, then $\overline{C} = (\bar{x}_1 \vee x_2 \vee x_3)$.

Theorem 3.4. Balanced ∀∃ 3-SAT-$(2, 2, 2, 2)$ *is* Π_2^p*-complete.*

Proof. Noting that Balanced ∀∃ 3-SAT-$(2, 2, 2, 2)$ is a special case of ∀∃ 3-SAT, we deduce that the problem is in Π_2^p. We show that the problem is Π_2^p-hard by a reduction from ∀∃ NAE-3-SAT. Let

$$\Phi_1 = \forall X_1^p \exists X_{p+1}^n \varphi$$

be an instance of ∀∃ NAE-3-SAT, where

$$\varphi = \bigwedge_{j=1}^{m} C_j$$

is a Boolean formula over a set $V_1 = \{x_1, x_2, \ldots, x_n\}$ of variables such that C_j is a disjunction of at most three literals for all $j \in \{1, 2, \ldots, m\}$ and no C_j contains only a single literal since, otherwise, Φ_1 is a no-instance. Following Schaefer [13, p. 298] and noting that his reduction translates without changes to ∀∃ NAE-3-SAT, we first modify Φ_1 using the following transformation that turns every universal variable x_i of Φ_1 into an existential variable y_i and introduces the set of new universal variables $\{z_1, z_2, \ldots, z_p\}$:

$$\Phi_2 = \forall Z_1^p \exists X_{p+1}^n \exists Y_1^p \varphi[x_1 \mapsto y_1, \ldots, x_p \mapsto y_p] \wedge \bigwedge_{i=1}^{p} ((\bar{z}_i \vee y_i) \wedge (z_i \vee \bar{y}_i)) = \forall Z_1^p \exists X_{p+1}^n \exists Y_1^p \varphi',$$

where $\varphi' = \varphi[x_1 \mapsto y_1, \ldots, x_p \mapsto y_p] \wedge \bigwedge_{i=1}^{p} ((\bar{z}_i \vee y_i) \wedge (z_i \vee \bar{y}_i))$. Let V_2 be the set of variables of Φ_2.

3.4.1. Φ_1 *is a yes-instance of* ∀∃ NAE-3-SAT *if and only if* Φ_2 *is a yes-instance of* ∀∃ NAE-3-SAT.

Proof. First, suppose that Φ_1 is a yes-instance of ∀∃ NAE-3-SAT. Let β_1 be a truth assignment for V_1 that nae-satisfies Φ_1, and let β_2 be the following truth assignment for V_2:

(i) set $\beta_2(x_i) = \beta_1(x_i)$ for each $i \in \{p+1, p+2, \ldots, n\}$;
(ii) set $\beta_2(y_i) = \beta_1(x_i)$ for each $i \in \{1, 2, \ldots, p\}$;
(iii) set $\beta_2(z_i) = \beta_2(y_i)$ for each $i \in \{1, 2, \ldots, p\}$.

By (iii), it is straightforward to check that β_2 nae-satisfies Φ_2 and that, for every truth assignment for Z_1^p, there exists a truth assignment for $X_{p+1}^n \cup Y_1^p$ that nae-satisfies Φ_2. Hence, Φ_2 is a yes-instance of ∀∃ NAE-3-SAT.

Second, suppose that Φ_2 is a yes-instance of ∀∃ NAE-3-SAT. Let β_2 be a truth assignment for V_2 that nae-satisfies Φ_2. By construction of Φ_2, it follows that $\beta_2(z_i) = \beta_2(y_i)$ for each $i \in \{1, 2, \ldots, p\}$. Hence β_1 with $\beta_1(x_i) = \beta_2(y_i)$ for each $i \in \{1, 2, \ldots, p\}$, and $\beta_1(x_i) = \beta_2(x_i)$ for each $i \in \{p+1, p+2, \ldots, n\}$ is a truth assignment for V_1 that nae-satisfies Φ_2. Thus Φ_1 is a yes-instance of ∀∃ NAE-3-SAT. □

For each $w_i \in X_{p+1}^n \cup Y_1^p$, recall that $a(w_i)$ denotes the number of appearances of w_i in φ'. Next, we apply, in turn, the following transformation adapted from Berman et al. [1, p. 4] yielding an instance of ∀∃ 3-SAT.

1. Replace $\exists X_{p+1}^n$ in Φ_2 with the following list of existential variables:

   $$\exists x_{p+1,1} \exists x_{p+1,2} \cdots \exists x_{p+1,a(x_{p+1})} \cdots \exists x_{n,1} \exists x_{n,2} \cdots \exists x_{n,a(x_n)}$$

   Similarly, replace $\exists Y_1^p$ in Φ_2 with the following list of existential variables:

   $$\exists y_{1,1} \exists y_{1,2} \cdots \exists y_{1,a(y_1)} \cdots \exists y_{p,1} \exists y_{p,2} \cdots \exists y_{p,a(y_p)}.$$

   Lastly, for each existential variable $w_i \in X_{p+1}^n \cup Y_1^p$ and all $k \in \{1, \ldots, a(w_i)\}$, replace the k-th appearance of w_i in φ' by $w_{i,k}$.
2. Replace each clause C_j with $C_j \wedge \overline{C}_j$.
3. For each $w_i \in X_{p+1}^n \cup Y_1^p$, introduce the clauses

   $$(\overline{w_{i,1}} \vee w_{i,2}) \wedge (\overline{w_{i,2}} \vee w_{i,3}) \wedge \cdots \wedge (\overline{w_{i,a(w_i)-1}} \vee w_{i,a(w_i)}) \wedge (\overline{w_{i,a(w_i)}} \vee w_{i,1}).$$

4. Replace each 2-clause $(\ell_1 \vee \ell_2)$ by $(\ell_1 \vee \ell_2 \vee \overline{u}) \wedge E(u)$, where u and all 18 variables introduced by $E(u)$ are new existential variables. Append all 19 new variables to the list of existential variables.

Let Φ_3 denote the formula constructed by the preceding four-step procedure, and let V_3 be the set of variables of Φ_3. To illustrate the construction of Φ_3 from Φ_1, we present an example after the proof of this theorem.

3.4.2. *Φ_2 is a yes-instance of ∀∃ NAE-3-SAT if and only if Φ_3 is a yes-instance of ∀∃ 3-SAT.*

Proof. First, suppose that Φ_2 is a yes-instance of ∀∃ NAE-3-SAT. Let β_2 be a truth assignment for V_2 that nae-satisfies Φ_2. Obtain a truth assignment β_3 for V_3 as follows:

(i) set $\beta_3(z_i) = \beta_2(z_i)$ for each $i \in \{1, 2, \ldots, p\}$;
(ii) set $\beta_3(x_{i,k}) = \beta_2(x_i)$ for each $i \in \{p+1, p+2, \ldots, n\}$ and $k \in \{1, 2, \ldots, a(x_i)\}$;
(iii) set $\beta_3(y_{i,k}) = \beta_2(y_i)$ for each $i \in \{1, 2, \ldots, p\}$ and $k \in \{1, 2, \ldots, a(y_i)\}$.

Additionally, for each 2-clause $C = (\ell_1 \vee \ell_2)$ that is replaced with the $\mathcal{S}$-enforcer $(\ell_1 \vee \ell_2 \vee \overline{u}) \wedge E(u)$ in Step 4, set $\beta_3(u) = T$, and set all 18 existential variables introduced by $E(u)$ such that the 25 clauses of $E(u)$ are satisfied. By construction of $E(u)$ and Observation 3.1, this is always possible. If C is a 2-clause of Φ_2, then, as C is nae-satisfied by β_2, it follows that β_3 satisfies $(\ell_1 \vee \ell_2 \vee \overline{u}) \wedge E(u)$. If C is initially a 2-clause introduced in Step 3 and then replaced in Step 4, it follows by (ii) and (iii) that β_3 satisfies $(\ell_1 \vee \ell_2 \vee \overline{u}) \wedge E(u)$. Noting that if a truth assignment nae-satisfies a clause, then it also nae-satisfies its complement, it is now straightforward to check that β_3 satisfies Φ_3 and, hence, Φ_3 is a yes-instance of ∀∃ 3-SAT.

Second, suppose that Φ_3 is a yes-instance of ∀∃ 3-SAT. Let β_3 be a truth assignment that satisfies Φ_3. Let u be an enforcer variable such that the 25 clauses associated with $E(u)$ are clauses of Φ_3 but not of Φ_2. By construction of $E(u)$ and Observation 3.1, we have $\beta_3(u) = T$. Now let $w_i \in X_{p+1}^n \cup Y_1^p$. As β_3 satisfies Φ_3 and each enforcer variable that is contained in V_3 is assigned to T under β_3, it follows from the clauses introduced in Step 3 that

$$\beta_3(w_{i,1}) = \beta_3(w_{i,2}) = \cdots = \beta_3(w_{i,a(w_i)}).$$

Let β_2 be the truth assignment for Φ_2 that is obtained from β_3 as follows:

(i) $\beta_2(z_i) = \beta_3(z_i)$ for each $i \in \{1, 2, \ldots, p\}$,
(ii) $\beta_2(x_i) = \beta_3(x_{i,1})$ for each $i \in \{p+1, p+2, \ldots, n\}$, and
(iii) $\beta_2(y_i) = \beta_3(y_{i,1})$ for each $i \in \{1, 2, \ldots, p\}$.

As β_3 satisfies Φ_3, it immediately follows that β_2 satisfies Φ_2. We complete the proof by showing that β_2 nae-satisfies Φ_2. Assume that there exists a clause C in Φ_2 whose literals all evaluate to T under β_2. Let D be the clause obtained from C by applying Step 1. If C contains exactly three literals, then all three literals of $\overline{D}$ evaluate to F; thereby contradicting that β_3 satisfies Φ_3. On the other hand, if C contains exactly two literals, then D is replaced with a 3-clause, say D', and an enforcer, say $E(u')$, in Step 4 and, similarly, $\overline{D}$ is replaced with a 3-clause, say D'', and an enforcer, say $E(u'')$, in Step 4. Note that D'' is not the complement of D'. Furthermore, again by Observation 3.1, we have $\beta_3(u') = \beta_3(u'') = T$. Now, as each literal of C evaluates to T, each literal of D'' evaluates to F under β_3; a contradiction. Hence β_2 nae-satisfies Φ_2, and so Φ_2 is a yes-instance of ∀∃ NAE-3-SAT. □

We next obtain a quantified Boolean formula Φ_4 from Φ_3 such that the number of universal variables in Φ_4 is equal to the number of existential variables in Φ_4. Let p_e be the number of existential variables in V_3, and let p_u be the number of universal variables in V_3. By construction, observe that $p_u = p \geq 0$. Since a new existential variable y_i has been introduced for each universal variable x_i in V_1 with $i \in \{1, 2, \ldots, p\}$, we have $p_e \geq p_u$. Let Q_k^1 be the enforcer with variables $\{a_k, b_k, c_k, d_k, q_k, r_k, u_k, v_k, w_k\}$ as introduced in Section 3.1. Obtain Φ_4 from Φ_3 by adding Q_k^1 to the Boolean formula, appending $\exists a_k \exists b_k \exists c_k \exists d_k$ to the list of existential variables, and appending $\forall q_k \forall r_k \forall u_k \forall v_k \forall w_k$ to the list of universal variables for each $k \in \{1, 2, \ldots, p_e - p_u\}$. It now follows that Φ_4 contains $p_e + 4(p_e - p_u) = 5p_e - 4p_u$ existential variables and $p_u + 5(p_e - p_u) = 5p_e - 4p_u$ universal variables. Moreover, by Lemma 3.2, we have that Φ_3 is a yes-instance of ∀∃ 3-SAT if and only if Φ_4 is a yes-instance of ∀∃ 3-SAT.

We complete the proof by showing that Φ_4 is an instance of Balanced ∀∃ 3-SAT-$(2, 2, 2, 2)$. Let V_4 be the set of variables of Φ_4. By the transformation of Φ_1 into Φ_3 and the construction of Q_k^1, it is easily checked that each universal variable in V_4 appears exactly twice unnegated and exactly twice negated in Φ_4. Now, consider the following three sets of existential variables:

(I) $S_1 = \bigcup_{k=1}^{p_e - p_u} \{a_k, b_k, c_k, d_k\}$,
(II) $S_2 = \bigcup_{i=p+1}^{n} \{x_{i,1}, x_{i,2}, \ldots, x_{i,a(x_i)}\}$ and
(III) $S_3 = \bigcup_{i=1}^{p} \{y_{i,1}, y_{i,2}, \ldots, y_{i,a(y_i)}\}$.

It follows again from the construction of Q_k^1 that each variable in S_1 appears exactly twice unnegated and exactly twice negated in Φ_4. Furthermore, by Steps 1–3 in the construction of Φ_3, it follows that each variable in $S_2 \cup S_3$ appears exactly twice unnegated and exactly twice negated in Φ_4. Lastly, each existential variable in $V_4 - (S_1 \cup S_2 \cup S_3)$ has been introduced by replacing a 2-clause $(\ell_1 \vee \ell_2)$ with $(\ell_1 \vee \ell_2 \vee \overline{u}) \wedge E(u)$ in Step 4 of the construction of Φ_3. Recall that u appears unnegated exactly twice and negated exactly once in $E(u)$, and that each of the 18 remaining variables introduced by $E(u)$ appears exactly twice unnegated and exactly twice negated in $E(u)$. It now follows that Φ_4 is an instance of Balanced ∀∃ 3-SAT-$(2, 2, 2, 2)$. We complete the proof of this theorem by noting that each clause of Φ_4 is a 3-clause that contains three distinct variables and that the size of Φ_4 is polynomial in the size of Φ_1. □

Example. We next present a simple example that illustrates the construction of Φ_3 from Φ_1 as described in the proof of Theorem 3.4. Using the same notation as in this proof, let

$$\Phi_1 = \forall X_1^1 \exists X_2^4 (x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_3 \vee \overline{x_4}).$$

Loosely speaking, we obtain Φ_2 from Φ_1 by turning each universal variable into an existential variable and introduce new universal variables. Specifically, we have

$$\Phi_2 = \forall Z_1^1 \exists X_2^4 \exists Y_1^1 (y_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{y_1} \vee x_3 \vee \overline{x_4}) \wedge (\overline{z_1} \vee y_1) \wedge (z_1 \vee \overline{y_1}).$$

To obtain Φ_3 from Φ_2, we apply the four-step construction as described immediately after the proof of Statement 3.4.1. It follows that

$$\begin{aligned}\Phi_2' = \forall Z_1^1 \exists x_{2,1} \exists x_{3,1} \exists x_{3,2} \exists x_{4,1} \exists y_{1,1} \exists y_{1,2} \exists y_{1,3} \exists y_{1,4} \\ (y_{1,1} \vee x_{2,1} \vee \overline{x_{3,1}}) \wedge (\overline{y_{1,2}} \vee x_{3,2} \vee \overline{x_{4,1}}) \wedge (\overline{z_1} \vee y_{1,3}) \wedge (z_1 \vee \overline{y_{1,4}}) \wedge \\ (\overline{y_{1,1}} \vee \overline{x_{2,1}} \vee x_{3,1}) \wedge (y_{1,2} \vee \overline{x_{3,2}} \vee x_{4,1}) \wedge (z_1 \vee \overline{y_{1,3}}) \wedge (\overline{z_1} \vee y_{1,4}) \wedge \\ (\overline{x_{2,1}} \vee x_{2,1}) \wedge (\overline{x_{3,1}} \vee x_{3,2}) \wedge (\overline{x_{3,2}} \vee x_{3,1}) \wedge (\overline{x_{4,1}} \vee x_{4,1}) \wedge \\ (\overline{y_{1,1}} \vee y_{1,2}) \wedge (\overline{y_{1,2}} \vee y_{1,3}) \wedge (\overline{y_{1,3}} \vee y_{1,4}) \wedge (\overline{y_{1,4}} \vee y_{1,1})\end{aligned}$$

is the formula obtained after the first three steps. Recalling that Step 4 introduces a copy of the enforcer $E(u)$ for each of the twelve 2-clauses in Φ_2', we have

$$\Phi_3 = \forall Z_1^1 \exists x_{2,1} \exists x_{3,1} \exists x_{3,2} \exists x_{4,1} \exists y_{1,1} \exists y_{1,2} \exists y_{1,3} \exists y_{1,4} \exists U_1^{12} \exists H_1^{216}$$
$$(y_{1,1} \vee x_{2,1} \vee \overline{x_{3,1}}) \wedge (\overline{y_{1,2}} \vee x_{3,2} \vee \overline{x_{4,1}}) \wedge (\overline{z_1} \vee y_{1,3} \vee \overline{u_1}) \wedge E(u_1) \wedge (z_1 \vee \overline{y_{1,4}} \vee \overline{u_2}) \wedge E(u_2) \wedge$$
$$(\overline{y_{1,1}} \vee \overline{x_{2,1}} \vee x_{3,1}) \wedge (y_{1,2} \vee \overline{x_{3,2}} \vee x_{4,1}) \wedge (z_1 \vee \overline{y_{1,3}} \vee \overline{u_3}) \wedge E(u_3) \wedge (\overline{z_1} \vee y_{1,4} \vee \overline{u_4}) \wedge E(u_4) \wedge$$
$$(\overline{x_{2,1}} \vee x_{2,1} \vee \overline{u_5}) \wedge E(u_5) \wedge (\overline{x_{3,1}} \vee x_{3,2} \vee \overline{u_6}) \wedge E(u_6) \wedge$$
$$(\overline{x_{3,2}} \vee x_{3,1} \vee \overline{u_7}) \wedge E(u_7) \wedge (\overline{x_{4,1}} \vee x_{4,1} \vee \overline{u_8}) \wedge E(u_8) \wedge$$
$$(\overline{y_{1,1}} \vee y_{1,2} \vee \overline{u_9}) \wedge E(u_9) \wedge (\overline{y_{1,2}} \vee y_{1,3} \vee \overline{u_{10}}) \wedge E(u_{10}) \wedge$$
$$(\overline{y_{1,3}} \vee y_{1,4} \vee \overline{u_{11}}) \wedge E(u_{11}) \wedge (\overline{y_{1,4}} \vee y_{1,1} \vee \overline{u_{12}}) \wedge E(u_{12}),$$

where $u_1, \ldots, u_{12}$ are the twelve existential enforcer variables and $h_1, \ldots, h_{216}$ are the new existential variables introduced by $E(u_1), E(u_2), \ldots, E(u_{12})$.

In Theorem 3.4, we have imposed the same bound on existential and universal variables, i.e. $s_1 = s_2 = t_1 = t_2$. By allowing separate bounds, i.e. $s_1 = s_2$ and $t_1 = t_2$, we obtain the following stronger result.

Theorem 3.5. Balanced ∀∃ 3-SAT-$(1, 1, 2, 2)$ *is* Π_2^p*-complete.*

Proof. Clearly, Balanced ∀∃ 3-SAT-$(1, 1, 2, 2)$ is in Π_2^p. We establish Π_2^p-hardness via a reduction from Balanced ∀∃ 3-SAT-$(2, 2, 2, 2)$. Let

$$\Phi_1 = \forall X_1^p \exists X_{p+1}^{2p} \varphi$$

be an instance of Balanced ∀∃ 3-SAT-$(2, 2, 2, 2)$. Let m be the number of 3-clauses of φ. As $3m = 2p \cdot 4$, observe that p is divisible by 3. Following a similar strategy as in the proof of Theorem 3.4, we apply the following 4-step process to transform Φ_1 into an instance Φ_4 of Balanced ∀∃ 3-SAT-$(1, 1, 2, 2)$.

1. Obtain
$$\Phi_2 = \forall C_1^p \exists X_{p+1}^{2p} \exists Y_1^p \exists Z_1^{6p} \varphi[x_1 \mapsto y_1, \ldots, x_p \mapsto y_p] \wedge \bigwedge_{i=1}^{p} \left(\mathcal{S}_u(\overline{c_i}, y_i, y_i) \wedge \mathcal{S}_u(c_i, \overline{y_i}, \overline{y_i}) \right),$$
by turning each universal variable in $x_i \in X_1^p$ into an existential variable y_i, adding new universal variables $c_1, c_2, \ldots, c_p$, and adding new existential variables $z_1, z_2, \ldots, z_{6p}$ that are introduced as new variables by copies of the $\mathcal{S}$-enforcer. By construction, each $y_i \in Y_1^p$ appears exactly four times unnegated and exactly four times negated in Φ_2.
2. For each $y_i \in Y_1^p$ and $k \in \{1, 2, 3, 4\}$, replace the k-th negated appearance of y_i with $\overline{y_{i,k}}$ and replace the k-th unnegated appearance of y_i with $y_{i,k}$. Then replace $\exists Y_1^p$ in Φ_2 with the following list of existential variables
$$\exists y_{1,1} \exists y_{1,2} \exists y_{1,3} \exists y_{1,4} \cdots \exists y_{p,1} \exists y_{p,2} \exists y_{p,3} \exists y_{p,4}.$$
3. Add the following clauses to the Boolean formula resulting from Step 2:
$$\bigwedge_{i=1}^{p} \left[(\overline{y_{i,1}} \vee y_{i,2} \vee \overline{d_{i,1}}) \wedge (\overline{y_{i,2}} \vee y_{i,3} \vee \overline{d_{i,1}}) \wedge d_{i,1}^{(2)} \wedge (\overline{y_{i,3}} \vee y_{i,4} \vee \overline{d_{i,2}}) \wedge (\overline{y_{i,4}} \vee y_{i,1} \vee \overline{d_{i,2}}) \wedge d_{i,2}^{(2)} \right],$$
where $d_{i,1}$ and $d_{i,2}$ are new existential variables with $i \in \{1, 2 \ldots, p\}$, and $d_{i,1}^{(2)}$ and $d_{i,2}^{(2)}$ are the corresponding enforcers as introduced in Section 2. Then append
$$\exists d_{1,1} \exists d_{1,2} \exists d_{2,1} \exists d_{2,2} \cdots \exists d_{p,1} \exists d_{p,2} \exists E_1^{14p}$$
to the list of existential variables, where E_1^{14p} is the set of new variables introduced by these enforcers (each of $d_{i,1}^{(2)}$ and $d_{i,2}^{(2)}$ introduces seven such variables). Let Φ_3 denote the resulting quantified Boolean formula.
4. Note that each universal variable of Φ_3 appears exactly once unnegated and exactly once negated, and that each existential variable of Φ_3 appears exactly twice unnegated and exactly twice negated. Let p_e (resp. p_u) be the number of existential (resp. universal) variables in Φ_3. Then
$$p_e = p + 4p + 6p + 2p + 14p = 27p \quad \text{and} \quad p_u = p.$$
Evidently, $p_e \geq p_u$. Furthermore, as p is divisible by 3, it follows that p_e and p_u are both divisible by 3. Let $\Delta = (p_e - p_u)/3$. Now, for each $k \in \{1, 2, \ldots, \Delta\}$, add the enforcer Q_k^3 as introduced in Section 3.1 to Φ_3, append $\exists a_k \exists b_k$ to the list of existential variables, and append $\forall q_k \forall r_k \forall u_k \forall v_k \forall w_k$ to the list of universal variables.

Let Φ_4 denote the formula resulting from the preceding 4-step process. By construction, each clause in Φ_4 is a 3-clause that contains three distinct variables. Moreover, since, for each k, the enforcer Q_k^3 increases the number of universal variables by five and the number of existential variables by two, it follows that Φ_4 is an instance of Balanced ∀∃ 3-SAT-$(1, 1, 2, 2)$.

Noting that the size of Φ_4 is polynomial in the size of Φ_1, we complete the proof by establishing the following statement.

3.5.1. *Φ_1 is a yes-instance of* Balanced ∀∃ 3-SAT-$(2, 2, 2, 2)$ *if and only if Φ_4 is a yes-instance of* Balanced ∀∃ 3-SAT-$(1, 1, 2, 2)$.

Proof. Let V_1 be the set of variables of Φ_1, and let V_4 be the set of variables of Φ_4. First, suppose that Φ_1 is a yes-instance of Balanced ∀∃ 3-SAT-$(2, 2, 2, 2)$. Let β_1 be a truth assignment that satisfies Φ_1. We obtain a truth assignment β_4 for a subset of V_4, say V_4', from β_1 as follows:

(i) for each $i \in \{1, 2, \ldots, p\}$, set $\beta_4(c_i) = \beta_1(x_i)$;
(ii) for each $i \in \{p+1, p+2, \ldots, 2p\}$, set $\beta_4(x_i) = \beta_1(x_i)$;
(iii) for each $i \in \{1, 2, \ldots, p\}$ and $k \in \{1, 2, 3, 4\}$, set $\beta_4(y_{i,k}) = \beta_4(c_i)$;
(iv) for each $i \in \{1, 2, \ldots, p\}$ and $k \in \{1, 2\}$, set $\beta_4(d_{i,k}) = T$.

It is straightforward to check that each clause in Φ_4 that does not contain a variable in

$$(A_1^\Delta \cup B_1^\Delta \cup E_1^{14p} \cup Q_1^\Delta \cup R_1^\Delta \cup U_1^\Delta \cup V_1^\Delta \cup W_1^\Delta \cup Z_1^{6p})$$

is satisfied by β_4. We next extend β_4 in three steps. First, by (iv) and Observation 2.2, it follows that β_4 extends to $V_4' \cup E_1^{14p}$ such that, for each $i \in \{1, 2, \ldots, p\}$, the clauses of $d_{i,1}^{(2)}$ and $d_{i,2}^{(2)}$ are satisfied. Second, by Lemma 3.3, β_4 also extends to

$$V_4' \cup A_1^\Delta \cup B_1^\Delta \cup Q_1^\Delta \cup R_1^\Delta \cup U_1^\Delta \cup V_1^\Delta \cup W_1^\Delta$$

such that each clause in $Q_1^3 \wedge Q_2^3 \wedge \cdots \wedge Q_\Delta^3$ is satisfied. Third, by (i), (iii), and Observation 2.1 together with its subsequent remark, it follows that β_4 extends to $V_4' \cup Z_1^{6p}$ such that the clauses in

$$\bigwedge_{i=1}^{p} \left(\mathcal{S}_u(\overline{c_i}, y_i, y_i) \wedge \mathcal{S}_u(c_i, \overline{y_i}, \overline{y_i})\right)$$

are satisfied. We deduce that Φ_4 is satisfiable.

Second, suppose that Φ_4 is a yes-instance of Balanced ∀∃ 3-SAT-$(1, 1, 2, 2)$. Let β_4 be a truth assignment that satisfies Φ_4. It follows from Observation 2.2, that $\beta_4(d_{i,1}) = \beta_4(d_{i,2}) = T$ for each $i \in \{1, 2, \ldots, p\}$. Hence, the clauses introduced in Step 3 imply that

$$\beta_4(y_{i,1}) = \beta_4(y_{i,2}) = \beta_4(y_{i,3}) = \beta_4(y_{i,4}).$$

It is now easy to check that the truth assignment β_1 for V_1 obtained from β_4 by setting

(i) $\beta_1(x_i) = \beta_4(y_{i,1})$ for each $i \in \{1, 2, \ldots, p\}$ and
(ii) $\beta_1(x_i) = \beta_4(x_i)$ for each $i \in \{p+1, p+2, \ldots, 2p\}$

satisfies Φ_1. Thus, Statement 3.5.1 holds. □

This completes the proof of Theorem 3.5. □

We end this section by remarking that Haviv et al. [8, p. 55] showed that ∀∃ 3-SAT-(s_1, s_2, t_1, t_2) is in NP if $s_1 + s_2 \leq 1$ and in co-NP if $t_1 + t_2 \leq 2$. The latter result implies that Balanced ∀∃ 3-SAT-$(1, 1, 1, 1)$ is in co-NP. Hence, unless the polynomial hierarchy collapses, the balanced bounds on the number of appearances of universal and existential variables established in Theorems 3.4 and 3.5 are the best possible ones (i.e., for smaller values, the problems can be placed on a lower level of the polynomial hierarchy).

3.3. Hardness of ∀∃ 3-SAT-(s_1, s_2, t_1, t_2)

Following on from the results by Haviv et al. [8, p. 55] mentioned in the last paragraph, ∀∃ 3-SAT-(s_1, s_2, t_1, t_2) with $s_1 + s_2 \leq 1$ or $t_1 + t_2 \leq 2$ is not Π_2^P-hard unless the polynomial hierarchy collapses. In this section, we show which instances of ∀∃ 3-SAT-(s_1, s_2, t_1, t_2) are NP-complete and which are Π_2^P-complete. Specifically, we show that ∀∃ 3-SAT-(s_1, s_2, t_1, t_2) is NP-complete for when $s_1 + s_2 = 1$ and $(t_1, t_2) \in \{(1, 2), (2, 1)\}$, and Π_2^P-complete for when $s_1 = s_2 = 1$ and $(t_1, t_2) \in \{(1, 2), (2, 1)\}$.

Let Φ be an instance of $\forall\exists$ 3-SAT-(s_1, s_2, t_1, t_2) with $s_1 + s_2 = 1$, and let Y_1^p be the set of universal variables of Φ. As noted by Haviv et al. [8, p. 55], we can obtain an equivalent unquantified Boolean formula from Φ by deleting all literals in $\{y_i, \overline{y}_i : i \in \{1, 2, \ldots, p\}\}$ in the clauses of Φ. Hence, if Φ has the additional property that $t_1 + t_2 \leq 2$, it follows from results by Tovey [17, Section 3] that it can be decided in polynomial time whether or not Φ is a yes-instance. Hence, $\forall\exists$ 3-SAT-(s_1, s_2, t_1, t_2) with $s_1 + s_2 = 1$ and $t_1 + t_2 \leq 2$ is polynomial-time solvable. The next theorem shows that $\forall\exists$ 3-SAT-(s_1, s_2, t_1, t_2) with $s_1 + s_2 = 1$ becomes NP-complete if $(t_1, t_2) \in \{(1, 2), (2, 1)\}$. To establish this result, we use a variant of 3-SAT in which each clause is either a 2-clause or a 3-clause, and each variable appears exactly twice unnegated and exactly once negated, or exactly once unnegated and exactly twice negated. We refer to this variant as 3-SAT-(3). It was shown by Dahlhaus et al. [4, p. 877f] that 3-SAT-(3) is NP-complete. To establish the next theorem, we impose the following two restrictions on an instance φ of 3-SAT-(3).

(R1) Each 2-clause (resp. 3-clause) contains 2 (resp. 3) distinct variables.
(R2) Amongst the clauses, each variable appears exactly twice unnegated and exactly once negated.

Indeed, it follows immediately from Dahlhaus et al. [4, p. 877f] construction that φ satisfies (R1). Moreover, standard pre-processing that replaces each literal of a variable that appears exactly once unnegated and exactly twice negated with its negation can be used to obtain an instance φ' from φ that satisfies (R2) and that is equivalent to φ. We hence obtain the following theorem.

Theorem 3.6. $\forall\exists$ 3-SAT-(s_1, s_2, t_1, t_2) *is* NP*-complete if* $s_1 + s_2 = 1$ *and* $(t_1, t_2) \in \{(1, 2), (2, 1)\}$.

Proof. It was shown by Haviv et al. [8, p. 55] that $\forall\exists$ 3-SAT-(s_1, s_2, t_1, t_2) with $s_1 + s_2 = 1$ is in NP. We first establish NP-completeness for $\forall\exists$ 3-SAT-$(1, 0, 2, 1)$ via a reduction from 3-SAT-(3).

Let

$$\varphi = \bigwedge_{j=1}^{p} C_j^2 \wedge \bigwedge_{j=p+1}^{m} C_j^3$$

be an instance of 3-SAT-(3) over a set X_1^n of variables and such that each clause C_j^k is a k-clause with $k \in \{2, 3\}$. As described prior to the statement of Theorem 3.6, we may assume that φ satisfies (R1) and (R2). Construct the following quantified Boolean formula Φ from φ:

$$\Phi = \forall Y_1^p \exists X_1^n \left(\bigwedge_{j=1}^{p} (C_j^2 \vee y_i) \wedge \bigwedge_{j=p+1}^{m} C_j^3 \right).$$

Since φ satisfies (R2), Φ is an instance of $\forall\exists$ 3-SAT-$(1, 0, 2, 1)$. First, suppose that φ is satisfiable. Then there is a truth assignment β that satisfies each clause in φ. In particular, β satisfies each clause C_j^2 and, hence, any extension of β to Y_1^p with $i \in \{1, 2, \ldots, p\}$ is a truth assignment that satisfies Φ. Second, suppose that Φ is satisfiable. Let β' be a truth assignment for Φ such that $\beta'(y_i) = F$ for each $i \in \{1, 2, \ldots, p\}$. By the existence of β', it follows that $\beta(x_i) = \beta'(x_i)$ for each $i \in \{1, 2, \ldots, n\}$ is a truth assignment that satisfies each clause in φ. As the size of Φ is polynomial in the size of φ, NP-completeness of $\forall\exists$ 3-SAT-$(1, 0, 2, 1)$ now follows. To see that $\forall\exists$ 3-SAT-(s_1, s_2, t_1, t_2) is also NP-complete for when

(I) $(s_1, s_2, t_1, t_2) = (0, 1, 2, 1)$,
(II) $(s_1, s_2, t_1, t_2) = (1, 0, 1, 2)$, or
(III) $(s_1, s_2, t_1, t_2) = (0, 1, 1, 2)$,

observe that an instance of any of (I)-(III) is equivalent to an instance of $\forall\exists$ 3-SAT-$(1, 0, 2, 1)$ by replacing each literal of a universal or an existential (or both) variable with its negation. □

Theorem 3.7. $\forall\exists$ 3-SAT-$(1, 1, t_1, t_2)$ *with* $(t_1, t_2) \in \{(1, 2), (2, 1)\}$ *is* Π_2^p*-complete.*

Proof. We first establish the theorem for $(t_1, t_2) = (2, 1)$. Throughout the proof, we make use of the following quantified enforcer for an existential variable $d_{i,k}$:

$$E_\forall(d_{i,k}) = (d_{i,k} \vee u_{i,k} \vee v_{i,k}) \wedge (d_{i,k} \vee \overline{u_{i,k}} \vee \overline{v_{i,k}}),$$

where $u_{i,k}$ and $v_{i,k}$ are new universal variables for some $i, k \in \mathbb{Z}^+$. The following property of $E_\forall(d_{i,k})$ is easy to verify.

(P1) The Boolean formula $\forall u_{i,k} \forall v_{i,k} \exists d_{i,k} E_\forall(d_{i,k})$ is a yes-instance of $\forall\exists$ 3-SAT. In particular, if a truth assignment β for $\{d_{i,k}, u_{i,k}, v_{i,k}\}$ has the property that $\beta(d_{i,k}) = T$, then β satisfies $E_\forall(d_{i,k})$. Furthermore, if β satisfies $E_\forall(d_{i,k})$ and $\beta(u_{i,k}) = \beta(v_{i,k})$, then this implies that $\beta(d_{i,k}) = T$.

As ∀∃ 3-SAT-(1, 1, 2, 1) is a special case of ∀∃ 3-SAT, it follows that the former problem is in Π_2^P. We show Π_2^P-hardness by a reduction from Balanced ∀∃ 3-SAT-(1, 1, 2, 2), for which Π_2^P-completeness was established in Theorem 3.5. Let

$$\Phi_1 = \forall X_1^p \exists Y_{p+1}^{2p} \varphi$$

be an instance of Balanced ∀∃ 3-SAT-(1, 1, 2, 2). The reduction has two steps:

1. For each existential variable y_i of Φ_1 with $i \in \{p+1, p+2, \ldots, 2p\}$, replace the first (resp. second) unnegated appearance of y_i with $y_{i,1}$ (resp. $y_{i,2}$), replace the first (resp. second) negated appearance of y_i with the negation of $y_{i,3}$ (resp. $y_{i,4}$), and add the new clauses

$$(\overline{y_{i,1}} \vee y_{i,2} \vee \overline{d_{i,1}}) \wedge E_\forall(d_{i,1}) \wedge (\overline{y_{i,2}} \vee y_{i,3} \vee \overline{d_{i,2}}) \wedge E_\forall(d_{i,2}) \wedge$$
$$(\overline{y_{i,3}} \vee y_{i,4} \vee \overline{d_{i,3}}) \wedge E_\forall(d_{i,3}) \wedge (\overline{y_{i,4}} \vee y_{i,1} \vee \overline{d_{i,4}}) \wedge E_\forall(d_{i,4}),$$

   to Φ_1, where each $d_{i,k}$ with $k \in \{1, 2, 3, 4\}$ is a new existential variable. For all $i \in \{p+1, p+2, \ldots, 2p\}$ and $k \in \{1, 2, 3, 4\}$, append $y_{i,k}$ and $d_{i,k}$ to the list of existential variables and append $u_{i,k}$ and $v_{i,k}$ to the list of universal variables.
2. For each existential variable $y_{i,k}$ with $k \in \{3, 4\}$, replace each literal $y_{i,k}$ with $\overline{y_{i,k}}$ and each literal $\overline{y_{i,k}}$ with $y_{i,k}$.

Let Φ_2 be the resulting quantified Boolean formula, and let V_2 be the set of variables of Φ_2. (An example of this construction is given after the proof of the theorem.) Note that each existential variable $y_{i,k}$ with $k \in \{3, 4\}$ appears exactly once unnegated and exactly twice negated in the Boolean formula resulting from Step 1. Hence, due to Step 2, it follows that Φ_2 is an instance of ∀∃ 3-SAT-(1, 1, 2, 1). Furthermore, for each $i \in \{p+1, p+2, \ldots, 2p\}$, the clauses introduced in Step 1 are replaced with the following clauses in Step 2:

$$(\overline{y_{i,1}} \vee y_{i,2} \vee \overline{d_{i,1}}) \wedge E_\forall(d_{i,1}) \wedge (\overline{y_{i,2}} \vee \overline{y_{i,3}} \vee \overline{d_{i,2}}) \wedge E_\forall(d_{i,2}) \wedge$$
$$(y_{i,3} \vee \overline{y_{i,4}} \vee \overline{d_{i,3}}) \wedge E_\forall(d_{i,3}) \wedge (y_{i,4} \vee y_{i,1} \vee \overline{d_{i,4}}) \wedge E_\forall(d_{i,4}).$$

We complete the proof for $(t_1, t_2) = (2, 1)$ by establishing the following statement.

3.7.1. *Φ_1 is a yes-instance of* Balanced ∀∃ 3-SAT-(1, 1, 2, 2) *if and only if Φ_2 is a yes-instance of* ∀∃ 3-SAT-(1, 1, 2, 1).

Proof. First, suppose that Φ_1 is a yes-instance of Balanced ∀∃ 3-SAT-(1, 1, 2, 2). Let β_1 be a truth assignment that satisfies Φ_1. For every truth assignment β_2' for the universal variables in

$$\{u_{i,k}, v_{i,k} : i \in \{p+1, p+2, \ldots, 2p\} \text{ and } k \in \{1, 2, 3, 4\}\},$$

we extend β_2' to a truth assignment β_2 for V_2 as follows:

(i) set $\beta_2(x_i) = \beta_1(x_i)$ for each $i \in \{1, 2, \ldots, p\}$;
(ii) set $\beta_2(y_{i,k}) = \beta_1(y_i)$ for each $i \in \{p+1, p+2, \ldots, 2p\}$ and $k \in \{1, 2\}$;
(iii) set $\beta_2(y_{i,k}) = \overline{\beta_1(y_i)}$ for each $i \in \{p+1, p+2, \ldots, 2p\}$ and $k \in \{3, 4\}$;
(iv) set $\beta_2(d_{i,k}) = T$ for each $i \in \{p+1, p+2, \ldots, 2p\}$ and $k \in \{1, 2, 3, 4\}$.

Due to (iv) and Property (P1), it is now easily checked that Φ_2 is a yes-instance of ∀∃ 3-SAT-(1, 1, 2, 1).

Second, suppose that Φ_2 is a yes-instance of ∀∃ 3-SAT-(1, 1, 2, 1). Let β_2 be a truth assignment that satisfies Φ_2 such that $\beta_2(u_{i,k}) = \beta_2(v_{i,k})$ for each $i \in \{p+1, p+2, \ldots 2p\}$ and $k \in \{1, 2, 3, 4\}$. Since Φ_2 is a yes-instance, this implies that $\beta_2(d_{i,k}) = T$ by Property (P1). Moreover, by construction, we have

$$\beta_2(y_{i,1}) = \beta_2(y_{i,2}) \quad \text{and} \quad \overline{\beta_2(y_{i,1})} = \beta_2(y_{i,3}) = \beta_2(y_{i,4})$$

for each $i \in \{p+1, p+2, \ldots 2p\}$. It now follows that β_1 with

(i) $\beta_1(x_i) = \beta_2(x_i)$ for each $i \in \{1, 2, \ldots, p\}$ and
(ii) $\beta_1(y_i) = \beta_2(y_{i,1})$ for each $i \in \{p+1, p+2, \ldots, 2p\}$

is a truth assignment for the set of variables of Φ_1 that satisfies each clause in Φ_1 and, thus, Φ_1 is a yes-instance of Balanced ∀∃ 3-SAT-(1, 1, 2, 2). □

Noting that the size of Φ_2 is polynomial in the size of Φ_1, the theorem now follows for $(t_1, t_2) = (2, 1)$. Moreover, by observing that an instance of ∀∃ 3-SAT-$(1, 1, 1, 2)$ is equivalent to an instance of ∀∃ 3-SAT-$(1, 1, 2, 1)$ by replacing each literal of an existential variable with its negation, the theorem also follows for $(t_1, t_2) = (1, 2)$. □

Example. We next present a simple example that illustrates the construction of Φ_2 from Φ_1 as described in the proof of Theorem 3.7. Using the same notation as in this proof, let

$$\Phi_1 = \forall X_1^2 \exists Y_3^4 (x_1 \vee y_3 \vee \overline{y_4}) \wedge (x_2 \vee \overline{y_3} \vee y_4) \wedge (\overline{x_1} \vee \overline{y_3} \vee \overline{y_4}) \wedge (\overline{x_2} \vee y_3 \vee y_4).$$

Note that Φ_1 is an instance of Balanced ∀∃ 3-SAT-$(1, 1, 2, 2)$. In the first step of the reduction, we reduce the number of appearances for each existential variable by one and obtain

$$\begin{aligned}
\Phi_1' =& \forall X_1^2 \forall u_{3,1} \forall u_{3,2} \forall u_{3,3} \forall u_{3,4} \forall u_{4,1} \forall u_{4,2} \forall u_{4,3} \forall u_{4,4} \\
& \forall v_{3,1} \forall v_{3,2} \forall v_{3,3} \forall v_{3,4} \forall v_{4,1} \forall v_{4,2} \forall v_{4,3} \forall v_{4,4} \\
& \exists y_{3,1} \exists y_{3,2} \exists y_{3,3} \exists y_{3,4} \exists y_{4,1} \exists y_{4,2} \exists y_{4,3} \exists y_{4,4} \\
& \exists d_{3,1} \exists d_{3,2} \exists d_{3,3} \exists d_{3,4} \exists d_{4,1} \exists d_{4,2} \exists d_{4,3} \exists d_{4,4} \\
& (x_1 \vee y_{3,1} \vee \overline{y_{4,3}}) \wedge (x_2 \vee \overline{y_{3,3}} \vee y_{4,1}) \wedge (\overline{x_1} \vee \overline{y_{3,4}} \vee \overline{y_{4,4}}) \wedge (\overline{x_2} \vee y_{3,2} \vee y_{4,2}) \wedge \\
& (\overline{y_{3,1}} \vee y_{3,2} \vee \overline{d_{3,1}}) \wedge E_\forall(d_{3,1}) \wedge (\overline{y_{3,2}} \vee y_{3,3} \vee \overline{d_{3,2}}) \wedge E_\forall(d_{3,2}) \wedge \\
& (\overline{y_{3,3}} \vee y_{3,4} \vee \overline{d_{3,3}}) \wedge E_\forall(d_{3,3}) \wedge (\overline{y_{3,4}} \vee y_{3,1} \vee \overline{d_{3,4}}) \wedge E_\forall(d_{3,4}) \wedge \\
& (\overline{y_{4,1}} \vee y_{4,2} \vee \overline{d_{4,1}}) \wedge E_\forall(d_{4,1}) \wedge (\overline{y_{4,2}} \vee y_{4,3} \vee \overline{d_{4,2}}) \wedge E_\forall(d_{4,2}) \wedge \\
& (\overline{y_{4,3}} \vee y_{4,4} \vee \overline{d_{4,3}}) \wedge E_\forall(d_{4,3}) \wedge (\overline{y_{4,4}} \vee y_{4,1} \vee \overline{d_{4,4}}) \wedge E_\forall(d_{4,4}),
\end{aligned}$$

where $u_{i,k}$ and $v_{i,k}$ are the universal variables appearing in the quantified enforcer $E_\forall(d_{i,k})$ for $i \in \{3, 4\}$ and $k \in \{1, 2, 3, 4\}$. Now, each existential variable appears either twice unnegated and once negated or once unnegated and twice negated. In the second step, we negate all appearances of each existential variable that appears twice negated and once negated in Φ_1', which yields the quantified Boolean formula

$$\begin{aligned}
\Phi_2 =& \forall X_1^2 \forall u_{3,1} \forall u_{3,2} \forall u_{3,3} \forall u_{3,4} \forall u_{4,1} \forall u_{4,2} \forall u_{4,3} \forall u_{4,4} \\
& \forall v_{3,1} \forall v_{3,2} \forall v_{3,3} \forall v_{3,4} \forall v_{4,1} \forall v_{4,2} \forall v_{4,3} \forall v_{4,4} \\
& \exists y_{3,1} \exists y_{3,2} \exists y_{3,3} \exists y_{3,4} \exists y_{4,1} \exists y_{4,2} \exists y_{4,3} \exists y_{4,4} \\
& \exists d_{3,1} \exists d_{3,2} \exists d_{3,3} \exists d_{3,4} \exists d_{4,1} \exists d_{4,2} \exists d_{4,3} \exists d_{4,4} \\
& (x_1 \vee y_{3,1} \vee y_{4,3}) \wedge (x_2 \vee y_{3,3} \vee y_{4,1}) \wedge (\overline{x_1} \vee y_{3,4} \vee y_{4,4}) \wedge (\overline{x_2} \vee y_{3,2} \vee y_{4,2}) \wedge \\
& (\overline{y_{3,1}} \vee y_{3,2} \vee \overline{d_{3,1}}) \wedge E_\forall(d_{3,1}) \wedge (\overline{y_{3,2}} \vee \overline{y_{3,3}} \vee \overline{d_{3,2}}) \wedge E_\forall(d_{3,2}) \wedge \\
& (y_{3,3} \vee \overline{y_{3,4}} \vee \overline{d_{3,3}}) \wedge E_\forall(d_{3,3}) \wedge (y_{3,4} \vee y_{3,1} \vee \overline{d_{3,4}}) \wedge E_\forall(d_{3,4}) \wedge \\
& (\overline{y_{4,1}} \vee y_{4,2} \vee \overline{d_{4,1}}) \wedge E_\forall(d_{4,1}) \wedge (\overline{y_{4,2}} \vee \overline{y_{4,3}} \vee \overline{d_{4,2}}) \wedge E_\forall(d_{4,2}) \wedge \\
& (y_{4,3} \vee \overline{y_{4,4}} \vee \overline{d_{4,3}}) \wedge E_\forall(d_{4,3}) \wedge (y_{4,4} \vee y_{4,1} \vee \overline{d_{4,4}}) \wedge E_\forall(d_{4,4}).
\end{aligned}$$

4. Hardness of Monotone ∀∃ NAE-3-SAT-(s, t) with bounded variable appearances

4.1. Enforcers

In this section, we describe four monotone enforcers that have recently been introduced in an unquantified context by Darmann and Döcker [5]. For the purposes of this section, we use their enforcers in a quantified setting. Specifically, for the first three enforcers, x can be a universally or existentially quantified variable.

Auxiliary non-equality gadget. First, consider the auxiliary non-equality gadget

$$\mathrm{NE}_{\mathrm{aux}}(x, y) = (x \vee y \vee a) \wedge (x \vee y \vee b) \wedge (a \vee b \vee u) \wedge (a \vee b \vee v) \wedge (a \vee b \vee w) \wedge (u \vee v \vee w),$$

where a, b, u, v, w are five new existential variables, y is an existential variable, and x is a universal or existential variable. To nae-satisfy the last clause, at least one variable in $\{u, v, w\}$ is set to be T and at least one is set to be F. Then, by the three preceding clauses, we have that a truth assignment that nae-satisfies $\mathrm{NE}_{\mathrm{aux}}(x, y)$ assigns different truth values to a and b. The next observation follows by construction of the first two clauses.

Observation 4.1. *Consider the gadget* $\mathrm{NE}_{\mathrm{aux}}(x, y)$*, and let* V *be its associated set of variables. A truth assignment* β *for* $\{x, y\}$ *can be extended to a truth assignment* β' *for* V *that nae-satisfies* $\mathrm{NE}_{\mathrm{aux}}(x, y)$ *if and only if* $\beta(x) \neq \beta(y)$*.*

Equality gadget. The second enforcer is the equality gadget

$$\mathrm{EQ}(x, y) = \mathrm{NE}_{\mathrm{aux}}(p, q) \wedge \mathrm{NE}_{\mathrm{aux}}(p, r) \wedge (x \vee q \vee r) \wedge (y \vee q \vee r),$$

where p, q, r are three new existential variables, y is an existential variable, and x is a universal or existential variable. By construction and Observation 4.1, a truth assignment that nae-satisfies $\mathrm{EQ}(x, y)$ assigns the same truth value to q and r. The next observation follows by construction of the last two clauses in the equality gadget.

Observation 4.2. *Consider the gadget* $\mathrm{EQ}(x, y)$*, and let* V *be its associated set of variables. A truth assignment* β *for* $\{x, y\}$ *can be extended to a truth assignment* β' *for* V *that nae-satisfies* $\mathrm{EQ}(x, y)$ *if and only if* $\beta(x) = \beta(y)$*.*

Non-equality gadget. Combining the first and second enforcer, we now obtain another non-equality gadget:

$$\mathrm{NE}(x, y) = \mathrm{EQ}(x, p) \wedge \mathrm{EQ}(y, q) \wedge \mathrm{NE}_{\mathrm{aux}}(p, q),$$

where p and q are two new existential variables, y is an existential variable, and x is a universal or existential variable. The next observation follows immediately by construction, and Observations 4.1 and 4.2.

Observation 4.3. *Consider the gadget* $\mathrm{NE}(x, y)$*, and let* V *be its associated set of variables. A truth assignment* β *for* $\{x, y\}$ *can be extended to a truth assignment* β' *for* V *that nae-satisfies* $\mathrm{NE}(x, y)$ *if and only if* $\beta(x) \neq \beta(y)$*.*

The next observation follows by construction of the last three enforcers.

Observation 4.4. *Let* $\mathcal{E}$ *be an enforcer in* $\{\mathrm{NE}_{\mathrm{aux}}(x, y), \mathrm{EQ}(x, y), \mathrm{NE}(x, y)\}$*. Then each variable introduced by* $\mathcal{E}$ *appears at most four times in* $\mathcal{E}$*.*

Padding gadget. The fourth enforcer is the gadget

$$\mathrm{P1}(x) = (x \vee a \vee b) \wedge (a \vee c \vee d) \wedge (a \vee b \vee e) \wedge (a \vee d \vee e) \wedge (b \vee c \vee d) \wedge (b \vee c \vee e) \wedge (c \vee d \vee e),$$

where x is an existential variable, and a, b, c, d, e are five new existential variables each of which appears exactly four times in the gadget. For the truth assignment β for $\{a, b, c, d, e\}$ with $\beta(a) = \beta(c) = \beta(e) = T$ and $\beta(b) = \beta(d) = F$, the next observation follows immediately by construction.

Observation 4.5. *The gadget* $\mathrm{P1}(x)$ *is nae-satisfiable. Moreover, every truth assignment for* $\{x\}$ *can be extended to a truth assignment for* $\{a, b, c, d, e, x\}$ *that nae-satisfies* $\mathrm{P1}(x)$*.*

Intuitively, $P1(x)$ is used to increase the number of appearances of existential variables in a Boolean formula until each variable appears exactly four times.

4.2. Hardness of Monotone ∀∃ NAE-3-SAT-(s, t)

In this section, we establish that a monotone and linear Boolean formula φ of ∀∃ NAE-3-SAT is complete for the second level of the polynomial hierarchy even if each clause in φ contains at most one universal variable and, amongst the clauses in φ, each universal variable appears exactly once and each existential variable appears exactly three times. We start by establishing a slightly weaker result without linearity.

Proposition 4.6. Monotone ∀∃ NAE-3-SAT-$(1, 4)$ *is* Π_2^P*-complete if each clause contains at most one universal variable.*

Proof. Clearly, the decision problem Monotone ∀∃ NAE-3-SAT-$(1, 4)$ as described in the statement of the proposition is in Π_2^P. We show that it is Π_2^P-complete by a reduction from ∀∃ NAE-3-SAT. For the latter problem, Π_2^P-completeness was established by Eiter and Gottlob [6]. Let

$$\Phi_1 = \forall X_1^p \exists Y_{p+1}^n \varphi$$

be an instance of ∀∃ NAE-3-SAT over a set $V_1 = X_1^p \cup Y_{p+1}^n$ of variables. We may assume that each clause contains exactly three literals and at most one duplicate literal.

In what follows, we construct two quantified Boolean formulas that include copies of the enforcers introduced in Section 4.1. Each such enforcer adds several new existential variables. For ease of exposition throughout this proof, we use A to denote the set of all new existential variables that are introduced by a copy of an enforcer in

$$\{\mathrm{NE}_{\mathrm{aux}}(x, y), \mathrm{EQ}(x, y), \mathrm{NE}(x, y), \mathrm{P1}(x)\}.$$

In particular, A is initially empty and, each time we use a new enforcer copy, we add the newly introduced variables to A and append them to the list of existential variables without mentioning it explicitly. We remark that it will always be clear from the context that the number of elements in A is polynomial in the size of Φ_1.

Now, let

$$\Phi_2 = \forall Z_1^p \exists Y_1^n \exists A \varphi[x_1 \mapsto y_1, \ldots, x_p \mapsto y_p] \wedge \bigwedge_{i=1}^{p} \mathrm{EQ}(z_i, y_i)$$

be the quantified Boolean formula obtained from Φ_1 by first creating a copy z_i of each of the universal variables x_i, replacing each universal variable x_i of Φ_1 with a new existential variable y_i, and then, for all $i \in \{1, 2, \ldots, p\}$, adding the enforcer $\mathrm{EQ}(z_i, y_i)$, where z_i is a new universal variable. Furthermore, let $V_2' = Z_1^p \cup Y_1^n$, and let $V_2 = V_2' \cup A$. By construction, each clause in Φ_2 contains at most one universal variable and each universal variable appears exactly once in Φ_2.

4.6.1. *Φ_1 is a yes-instance of ∀∃ NAE-3-SAT if and only if Φ_2 is a yes-instance of ∀∃ NAE-3-SAT.*

Proof. First, suppose that Φ_1 is a yes-instance of ∀∃ NAE-3-SAT. Let β_1 be a truth assignment for V_1 that nae-satisfies Φ_1, and let β_2' be the following truth assignment for V_2':

(i) set $\beta_2'(y_i) = \beta_1(y_i)$ for each $i \in \{p+1, p+2, \ldots, n\}$;
(ii) set $\beta_2'(z_i) = \beta_1(x_i)$ for each $i \in \{1, 2, \ldots, p\}$;
(iii) set $\beta_2'(y_i) = \beta_1(x_i)$ for each $i \in \{1, 2, \ldots, p\}$.

By (iii) and Observation 4.2, it follows that there is a truth assignment β_2 for V_2 that extends β_2' such that, for each $i \in \{1, 2, \ldots, p\}$, the clauses of $\mathrm{EQ}(z_i, y_i)$ are nae-satisfied. Furthermore, since Φ_1 is nae-satisfiable for every truth assignment for X_1^p, if follows that Φ_2 is nae-satisfiable for every truth assignment for Z_1^p. Hence, Φ_2 is a yes instance of ∀∃ NAE-3-SAT.

Second, suppose that Φ_2 is a yes-instance of ∀∃ NAE-3-SAT. Let β_2 be a truth assignment for V_2 that nae-satisfies Φ_2. By Observation 4.2, it follows that $\beta_2(z_i) = \beta_2(y_i)$ for each $i \in \{1, 2, \ldots, p\}$. Hence β_1 with $\beta_1(x_i) = \beta_2(z_i)$ for each $i \in \{1, 2, \ldots, p\}$, and $\beta_1(y_i) = \beta_2(y_i)$ for each $i \in \{p+1, p+2, \ldots, n\}$ is a truth assignment for V_1 that nae-satisfies Φ_1. Furthermore, since Φ_2 is nae-satisfiable for every truth assignment for Z_1^p, it follows that Φ_1 is a yes-instance of ∀∃ NAE-3-SAT. □

Next, following Darmann and Döcker [5, Theorem 1], we transform Φ_2 into a new quantified Boolean formula in four steps:

1. To remove all negated variables, we start by replacing each appearance of an existential variable in Y_1^n with a new unnegated variable. Specifically, for each existential variable $y_i \in Y_1^n$, let $u(y_i)$ and $n(y_i)$ be the number of unnegated and negated appearances, respectively, of y_i in the Boolean formula of Φ_2. Recall that $u(y_i) + n(y_i) = a(y_i)$. Now, for each $j \in \{1, 2, \ldots, u(y_i)\}$, replace the j-th unnegated appearance of y_i in Φ_2 with $y_{i,j}$. Similarly, for each $j \in \{1, 2, \ldots, n(y_i)\}$, replace the j-th negated appearance of y_i in Φ_2 with $y_{i,u(y_i)+j}$. Lastly, for all $i \in \{1, 2, \ldots, n\}$, append

   $$\exists y_{i,1} \exists y_{i,2} \cdots \exists y_{i,a(y_i)}$$

   to the list of existential variables and remove the obsolete variables $\exists Y_1^n$.
2. If $u(y_i) > 1$, introduce the clauses

   $$\bigwedge_{i=1}^{n} \bigwedge_{j=1}^{u(y_i)-1} \mathrm{EQ}(y_{i,j}, y_{i,j+1}).$$

   Similarly, if $n(y_i) > 1$, introduce the clauses

   $$\bigwedge_{i=1}^{n} \bigwedge_{j=u(y_i)+1}^{a(y_i)-1} \mathrm{EQ}(y_{i,j}, y_{i,j+1}).$$

3. For each $i \in \{1, 2, \ldots, n\}$ with $u(y_i) \notin \{0, a(y_i)\}$, add the gadget

   $$\mathrm{NE}(y_{i,u(y_i)}, y_{i,u(y_i)+1}).$$

4. Let Φ_2' be the quantified Boolean formula resulting from the last three steps. For $i \in \{1, 2, \ldots, n\}$, consider an existential variable y_i in Φ_2. If y_i appears exactly once in Φ_2, then $y_{i,1}$ only appears once in Φ_2' because Steps 2 and 3 do not introduce any gadget that adds an additional appearance of $y_{i,1}$. Otherwise, if y_i appears at least twice in Φ_2, then the enforcers introduced in the previous two steps increase the number of appearances for each variable $y_{i,j}$ with $j \in \{1, 2, \ldots, a(y_i)\}$ by at least one and at most two. Hence, each variable $y_{i,j}$ appears at most three times in Φ_2'. Moreover, by construction and Observation 4.4, each variable in A appears at most four times in Φ_2'. Now, for each existential variable v in Φ_2' (this includes all variables in A), add the clauses

$$\bigwedge_{k=1}^{4-a(v)} \mathrm{P1}(v)$$

to Φ_2', where $a(v)$ denotes the number of appearances of v in Φ_2'.

Let Φ_3 be the quantified Boolean formula constructed by the preceding four-step procedure. Furthermore, let V_3 be the set of variables of Φ_3, and let $V_3' = V_3 - A$.

4.6.2. *Φ_2 is a yes-instance of* ∀∃ NAE-3-SAT *if and only if Φ_3 is a yes-instance of* ∀∃ NAE-3-SAT.

Proof. First, suppose that Φ_2 is a yes-instance of ∀∃ NAE-3-SAT. Let β_2 be a truth assignment for V_2 that nae-satisfies Φ_2. Obtain a truth assignment β_3' for V_3' as follows:

(i) set $\beta_3'(z_i) = \beta_2(z_i)$ for each $i \in \{1, 2, \ldots, p\}$;
(ii) set $\beta_3'(y_{i,j}) = \beta_2(y_i)$ for each $i \in \{1, 2, \ldots, n\}$ and $j \in \{1, 2, \ldots, u(y_i)\}$;
(iii) set $\beta_3'(y_{i,j}) = \overline{\beta_2(y_i)}$ for each $i \in \{1, 2, \ldots, n\}$ and $j \in \{u(y_i) + 1, u(y_i) + 2, \ldots, a(y_i)\}$.

By (ii) and (iii) as well as Observations 4.2, 4.3, and 4.5, it follows that there is a truth assignment β_3 for V_3 that extends β_3' and nae-satisfies Φ_3. Moreover, it follows by construction that for every truth assignment for Z_1^p, there exists a truth assignment for $V_3 - Z_1^p$ that nae-satisfies Φ_3. Hence, Φ_3 is a yes-instance of ∀∃ NAE-3-SAT.

Second, suppose that Φ_3 is a yes-instance of ∀∃ NAE-3-SAT. Let β_3 be a truth assignment for V_3 that nae-satisfies Φ_3. By Steps 2 and 3 of the construction, and by Observations 4.2 and 4.3, we have

(I) $\beta_3(y_{i,1}) = \beta_3(y_{i,2}) = \cdots = \beta_3(y_{i,u(y_i)})$,
(II) $\beta_3(y_{i,u(y_i)}) \neq \beta_3(y_{i,u(y_i)+1})$, and
(III) $\beta_3(y_{i,u(y_i)+1}) = \beta_3(y_{i,u(y_i)+2}) = \cdots = \beta_3(y_{i,a(y_i)})$

for all $i \in \{1, 2, \ldots, n\}$. Now, obtain a truth assignment β_2 for V_2 as follows:

(i) set $\beta_2(z_i) = \beta_3(z_i)$ for each $i \in \{1, 2, \ldots, p\}$;
(ii) set $\beta_2(y_i) = \beta_3(y_{i,1})$ for each $i \in \{1, 2, \ldots, n\}$ with $u(y_i) \geq 1$;
(iii) set $\beta_2(y_i) = \overline{\beta_3(y_{i,1})}$ for each $i \in \{1, 2, \ldots, n\}$ with $u(y_i) = 0$;
(iv) set $\beta_2(a) = \beta_3(a)$ for each $a \in A$ with $a \in V_2$.

It is now straightforward to check that β_2 nae-satisfies Φ_2 and, hence, Φ_2 is a yes-instance of ∀∃ NAE-3-SAT. □

We complete the proof by showing that Φ_3 has the desired properties. First, since all enforcers introduced in Section 4.1 are monotone, it follows from Step 1 in the construction of Φ_3 from Φ_2 that Φ_3 is monotone. Second, again by Step 1 in the construction of Φ_3 from Φ_2, it follows that each clause in Φ_3 is a 3-clause that contains three distinct variables. Third, turning to the universal variables in Φ_3 and as mentioned in the construction of Φ_2, each clause in Φ_2, and hence in Φ_3, contains at most one universal variable and each universal variable in Φ_2, and hence in Φ_3, appears exactly once. Fourth, recalling Step 4 in the construction of Φ_3 from Φ_2 and that each new existential variable of $\mathrm{P1}(v)$ appears exactly four times in the seven clauses associated with $\mathrm{P1}(v)$, it follows that each existential variable appears exactly four times in Φ_3. Noting that the size of Φ_3 is polynomial in the size of Φ, this establishes the proposition. □

We are now in a position to establish the main result of this section.

Theorem 4.7. Monotone ∀∃ NAE-3-SAT-(1, 3) *is Π_2^p-complete if the Boolean formula is linear and each clause contains at most one universal variable.*

Proof. Clearly, the decision problem Monotone ∀∃ NAE-3-SAT-(1, 3) as described in the statement of the theorem is in Π_2^p. We show Π_2^p-completeness by a reduction from Monotone ∀∃ NAE-3-SAT-(1, 4). Let

$$\Phi_1 = \forall Z_1^p \exists Z_{p+1}^n \varphi$$

be an instance of Monotone ∀∃ NAE-3-SAT-(1, 4). By Proposition 4.6, we may assume that each clause in Φ_1 contains at most one universal variable.

We start by defining the following four sets of variables which we use to construct an instance Φ_2 of Montone ∀∃ NAE-3-SAT-(1, 3). Let

$$U = \{u_{i,k} : i \in \{p+1, p+2, \ldots, n\} \text{ and } k \in \{1, 2, \ldots, 8\}\} \text{ and}$$
$$V = \{v_{i,k} : i \in \{p+1, p+2, \ldots, n\} \text{ and } k \in \{1, 2, \ldots, 8\}\}$$

be two sets of universal variables, and let

$$E = \{e_{i,k} : i \in \{p+1, p+2, \ldots, n\} \text{ and } k \in \{1, 2, \ldots, 8\}\} \text{ and}$$
$$Z = \{z_{i,k} : i \in \{p+1, p+2, \ldots, n\} \text{ and } k \in \{1, 2, \ldots, 8\}\}$$

be two sets of existential variables. Now, for each $i \in \{p+1, p+2, \ldots, n\}$, we replace the j-th appearance of z_i with $z_{i,j}$ for all $j \in \{1, 2, 3, 4\}$ and introduce the clauses

$$\bigwedge_{k=1}^{7} \big((z_{i,k} \vee e_{i,k} \vee u_{i,k}) \wedge (e_{i,k} \vee z_{i,k+1} \vee v_{i,k})\big) \wedge (z_{i,8} \vee e_{i,8} \vee u_{i,8}) \wedge (e_{i,8} \vee z_{i,1} \vee v_{i,8}) \wedge$$
$$(z_{i,5} \vee e_{i,1} \vee e_{i,2}) \wedge (z_{i,6} \vee e_{i,7} \vee e_{i,8}) \wedge (z_{i,7} \vee e_{i,3} \vee e_{i,4}) \wedge (z_{i,8} \vee e_{i,5} \vee e_{i,6}).$$

Furthermore, we append each element in $U \cup V$ to the list of universal variables, append each element in $E \cup Z$ to the list of existential variables, and delete the obsolete variables Z_{p+1}^n. Let Φ_2 denote the resulting formula. By construction, it is straightforward to check that Φ_2 is an instance of Monotone ∀∃ NAE-3-SAT-(1, 3) with at most one universal variable per clause and whose set of variables is

$$U \cup V \cup Z_1^p \cup E \cup Z.$$

Moreover, if any pair of clauses in Φ_1 have two variables in common, then both are existential variables and, hence, again by construction, Φ_2 is linear. Since Φ_2 has all desired properties and the size of Φ_2 is polynomial in the size of Φ_1, it remains to show that the following statement holds.

4.7.1. *Φ_1 is a yes-instance of* Monotone ∀∃ NAE-3-SAT-(1, 4) *if and only if Φ_2 is a yes-instance of* Monotone ∀∃ NAE-3-SAT-(1, 3).

Proof. First, suppose that Φ_1 is a yes-instance of Monotone ∀∃ NAE-3-SAT-(1, 4). Let β_1 be a truth assignment for $Z_1^p \cup Z_{p+1}^n$ that nae-satisfies Φ_1. Obtain a truth assignment β_2 for $Z_1^p \cup E \cup Z$ as follows:

(i) set $\beta_2(z_i) = \beta_1(z_i)$ for each $i \in \{1, 2, \ldots, p\}$;
(ii) set $\beta_2(z_{i,k}) = \beta_1(z_i)$ for each $i \in \{p+1, p+2, \ldots, n\}$ and $k \in \{1, 2, \ldots, 8\}$;
(iii) set $\beta_2(e_{i,k}) = \overline{\beta_1(z_i)}$ for each $i \in \{p+1, p+2, \ldots, n\}$ and $k \in \{1, 2, \ldots, 8\}$.

It is easily checked that every truth assignment for $U \cup V \cup Z_1^p \cup E \cup Z$ that extends β_2 nae-satisfies Φ_2, and thus Φ_2 is a yes-instance of Monotone ∀∃ NAE-3-SAT-(1, 3).

Second, suppose that Φ_2 is a yes-instance Monotone ∀∃ NAE-3-SAT-(1, 3). Let β_2 be a truth assignment that nae-satisfies Φ_2 such that $\beta_2(u_{i,k}) = F$ and $\beta_2(v_{i,k}) = T$ for each $i \in \{p+1, p+2, \ldots, n\}$ and $k \in \{1, 2, \ldots, 8\}$. Since $u_{i,k}$ and $v_{i,k}$ are universal variables, β_2 exists. We next show that β_2 satisfies the property

$$\beta_2(z_{i,1}) = \beta_2(z_{i,2}) = \beta_2(z_{i,3}) = \beta_2(z_{i,4})$$

for each $i \in \{p+1, p+2, \ldots, n\}$. To this end, consider the subset of clauses

$$(z_{i,8} \vee e_{i,8} \vee F) \wedge (e_{i,8} \vee z_{i,1} \vee T) \wedge \bigwedge_{k=1}^{7} \big((z_{i,k} \vee e_{i,k} \vee F) \wedge (e_{i,k} \vee z_{i,k+1} \vee T)\big)$$

of Φ_2, where the universal variables are set according to β_2. If $\beta_2(z_{i,1}) = F$, then the clause $(z_{i,1} \vee e_{i,1} \vee F)$ implies that $\beta_2(e_{i,1}) = T$ and, hence, by the aforementioned subset of clauses, $\beta_2(z_{i,j}) = F$ for each $j \in \{1, 2, 3, 4\}$. Otherwise, if $\beta_2(z_{i,1}) = T$, then the clause $(e_{i,8} \vee z_{i,1} \vee T)$ implies that $\beta_2(e_{i,8}) = F$ and, hence, again by the aforementioned subset of clauses, $\beta_2(z_{i,j}) = T$ for each $j \in \{1, 2, 3, 4\}$. It now follows that the truth assignment β_1 for $Z_1^p \cup Z_{p+1}^n$ with

(i) $\beta_1(z_i) = \beta_2(z_i)$ for each $i \in \{1, 2, \ldots, p\}$ and

(ii) $\beta_1(z_i) = \beta_2(z_{i,1})$ for each $i \in \{p+1, p+2, \ldots, n\}$

nae-satisfies Φ_1, and so Φ_1 is a yes-instance of Monotone ∀∃ NAE-3-SAT-$(1,4)$. □

This completes the proof of Theorem 4.7. □

4.3. *Restrictions that alleviate the complexity of* Monotone ∀∃ NAE-3-SAT-(s,t)

In this section, we discuss variants of Monotone ∀∃ NAE-3-SAT-(s,t) that are in co-NP or solvable in polynomial time. More precisely, we investigate the complexity of Monotone ∀∃ NAE-3-SAT-$(s,2)$. First note that Monotone ∀∃ NAE-3-SAT-$(0,t)$ is a special case of NAE-3-SAT and therefore in NP. Furthermore, Monotone ∀∃ NAE-3-SAT-$(s,1)$ can be solved in polynomial time since an instance of this problem is a yes-instance if and only if each clause contains at least one existential variable. Now, let t be a non-negative integer. Consider the following decision problem that allows for a set of variables and the set $\{F,T\}$ of constants.

Monotone-with-Constants-NAE-3-SAT-t (MC-NAE-3-SAT-t)
Input. A set $V = \{x_1, x_2, \ldots, x_n\}$ of variables and a monotone Boolean formula

$$\bigwedge_{j=1}^{m} C_j$$

such that each clause contains exactly three distinct elements in $V \cup \{F,T\}$ and, amongst the clauses, each element in V appears exactly t times.
Question. Does there exist a truth assignment $\beta\colon V \to \{T,F\}$ such that each clause of the formula is nae-satisfied?

Boolean formulas that include variables and the two constants F and T were, for example, previously considered in the context of NAE-3-SAT [2, p. 275f]. In particular, let φ be an instance of NAE-3-SAT that allows for constants. Bonet et al. [2] showed that, given a solution to φ, it is NP-complete to decide if a second solution to φ exists. This result was in turn used to prove that two problems arising in computational biology are NP-complete. Note that in the special case in which φ does not contain any constant, a second solution can always be obtained from a given truth assignment that nae-satisfies φ by simply interchanging T and F.

We next show that MC-NAE-3-SAT-t is solvable in polynomial time if $t = 2$.

Proposition 4.8. MC-NAE-3-SAT-2 *is in* P.

Proof. Let $\varphi = \bigwedge_{j=1}^{m} C_j$ be an instance of MC-NAE-3-SAT-2 over a set $V \cup \{F,T\}$ of variables and constants, where $V = \{x_1, x_2, \ldots, x_n\}$. To establish the proposition, we adapt ideas presented by Porschen et al. [11] who developed a linear-time algorithm to decide if an instance of NAE-SAT, i.e. a Boolean formula in CNF, is nae-satisfiable if each variable appears at most twice.

Using the notation $C_j = (\ell_{j,1} \vee \ell_{j,2} \vee \ell_{j,3})$ to denote the j-th clause in φ for each $j \in \{1,2,\ldots,m\}$, we next present an algorithm to decide whether or not φ is a yes-instance of MC-NAE-3-SAT-2. At each step of the algorithm, φ is transformed into a simpler Boolean formula.

1. For each clause C_j with $\ell_{j,k} = x_i$ for some $i \in \{1,2,\ldots,n\}$ and $\ell_{j,k'}, \ell_{j,k''} \in \{F,T\}$ with $\{k,k',k''\} = \{1,2,3\}$, do the following.
   (I) If $\ell_{j,k'} \neq \ell_{j,k''}$, remove C_j from φ.
   (II) If $\ell_{j,k'} = \ell_{j,k''} = F$, remove C_j and reset φ to be $\varphi[x_i \mapsto T]$.
   (III) If $\ell_{j,k'} = \ell_{j,k''} = T$, remove C_j and reset φ to be $\varphi[x_i \mapsto F]$.
2. For each pair of variables $x_i, x_{i'} \in V$ with $i \neq i'$ that both appear in two distinct clauses C_j and $C_{j'}$, remove C_j and $C_{j'}$ from φ.
3. For each variable x_i that appears in exactly one clause C_j, remove C_j from φ.
4. For each clause C_j such that $\ell_{j,1}, \ell_{j,2}, \ell_{j,3} \in \{T,F\}$, do the following.
   (I) If $\{\ell_{j,1}, \ell_{j,2}, \ell_{j,3}\} = \{T,F\}$, remove C_j from φ.
   (II) Otherwise, stop and return "φ is a no-instance".
5. Stop and return "φ is a yes-instance".

The algorithm clearly terminates within polynomial time. Moreover, as a variable that appears exactly once in a Boolean formula can be assigned to either T or F without affecting any other clause, it is straightforward to check that each step in the algorithm returns a Boolean formula that is equivalent to φ. Hence, if a clause contains three equal constants, then the algorithm correctly returns that φ is a no-instance in Step 4. Now, suppose that the algorithm returns "φ is a yes-instance".

Let φ' be the Boolean formula that is obtained at the end of the last iteration of Step 4(I). Then, φ' is an instance of MC-NAE-3-SAT-2 such that each clause contains at most one constant and each pair of clauses have at most one variable in common. Hence, φ' is linear. It remains to show that φ' is a yes-instance of MC-NAE-3-SAT-2.

Before continuing with the proof, we pause to give an overview of a result established by Porschen et al. [11]. Let $\psi = \bigwedge_{j=1}^{m'} C'_j$ be a linear and monotone Boolean formula where each variable appears exactly twice and each clause contains at least two distinct variables. Furthermore, let G_ψ be the *clause graph* for ψ whose set of vertices is $\{C'_1, C'_2, \ldots, C'_{m'}\}$ and, for each pair $j, j' \in \{1, 2, \ldots, m'\}$ with $j \neq j'$, there is an edge $\{C'_j, C'_{j'}\}$ in G_ψ precisely if C'_j and $C'_{j'}$ have a variable in common. Then ψ is nae-satisfiable if and only if there exists an edge coloring of G_ψ that uses exactly two colors c_1 and c_2 such that each vertex is incident to an edge that is colored c_1 and incident to an edge that is colored c_2. Moreover, if ψ does not have a connected component that is isomorphic to a cycle of odd length, then such an edge coloring exists.

We now continue with the proof of the proposition. Let φ'' be the Boolean formula obtained from φ' by omitting all constants. By construction, each clause in φ'' contains either two or three distinct variables. It follows that, if φ'' is nae-satisfiable, then φ' is nae-satisfiable. Let $G_{\varphi''}$ be the clause graph for φ''. First, assume that $G_{\varphi''}$ does not have a connected component that is isomorphic to a cycle of odd length. Then it immediately follows from the result by Porschen et al. [11] that φ'' is nae-satisfiable and, hence, φ' is also nae-satisfiable. Second, assume that $G_{\varphi''}$ has a connected component that is isomorphic to a cycle of odd length. Then the vertices of this component are of the form

$$(x_{i_1} \vee x_{i_2}), (x_{i_2} \vee x_{i_3}), \ldots, (x_{i_{p-1}} \vee x_{i_p}), (x_{i_p} \vee x_{i_1}),$$

where $p \geq 3$ is an odd integer and $x_{i_j} \in V$. In other words,

$$\mathcal{C}_{\varphi''} = \bigwedge_{j=1}^{p-1} (x_{i_j} \vee x_{i_{j+1}}) \wedge (x_{i_p} \vee x_{i_1}),$$

is contained in φ''. Although $\mathcal{C}_{\varphi''}$ is not nae-satisfiable, we next show that the corresponding clauses $\mathcal{C}_{\varphi'}$ in φ' are nae-satisfiable since each such clause contains exactly one constant.

Consider

$$\mathcal{C}_{\varphi'} = \bigwedge_{j=1}^{p-1} (x_{i_j} \vee x_{i_{j+1}} \vee b_j) \wedge (x_{i_p} \vee x_{i_1} \vee b_p),$$

where $b_j \in \{T, F\}$ for each $j \in \{1, 2, \ldots, p\}$. Let β be the following truth assignment for $\{x_{i_1}, x_{i_2}, \ldots, x_{i_p}\}$:

(i) set $\beta(x_{i_j}) = \overline{b_p}$ for each $j \in \{1, 2, \ldots, p\}$ with j being odd;
(ii) set $\beta(x_{i_j}) = b_p$ for each $j \in \{1, 2, \ldots, p\}$ with j being even.

It follows that β nae-satisfies $\mathcal{C}_{\varphi'}$. An analogous argument can be applied to every other connected component in $G_{\varphi''}$ that is isomorphic to a cycle of odd length. Furthermore, it again follows from Porschen et al.'s result [11] that the edge set of each connected component in $G_{\varphi''}$ that is not isomorphic to a cycle of odd length corresponds to a subset of clauses in φ'' that is nae-satisfiable. Altogether, φ'' is nae-satisfiable and, hence, φ' is also nae-satisfiable. This completes the proof of the proposition. □

We next establish three corollaries that pinpoint the complexity of Monotone ∀∃ NAE-3-SAT-$(s, 2)$.

Corollary 4.9. Monotone ∀∃ NAE-3-SAT-$(s, 2)$ *is in* co-NP *for any fixed positive integer s.*

Proof. A no-instance of Monotone ∀∃ NAE-3-SAT-$(s, 2)$ can be identified by taking an assignment of the universal variables and applying the algorithm presented in Proposition 4.8 to verify in polynomial time whether or not the resulting MC-NAE-3-SAT-2 Boolean formula (with omitted lists of universal and existential quantifies) is not nae-satisfiable. □

Corollary 4.10. Monotone ∀∃ NAE-3-SAT-$(s, 2)$ *is trivially a yes-instance for any fixed positive integer s if each clause contains at most one universal variable.*

Proof. Let Φ be an instance of Monotone ∀∃ NAE-3-SAT-$(s, 2)$ such that each clause contains at most one universal variable. We follow ideas that are similar to those presented in the algorithm described in the proof of Proposition 4.8. First, if there are two existential variables that both appear in two distinct clauses C_j and $C_{j'}$, obtain a new Boolean formula by removing C_j and $C_{j'}$ from Φ. Repeat this step until no such pair of variables remains. Then, if there is an existential variable that appears in exactly one clause C_j, obtain a new Boolean formula by removing C_j. Similar to the proof of Proposition 4.8, it follows that the resulting Boolean formula, say Φ', is an instance of Monotone ∀∃ NAE-3-SAT-$(s, 2)$ such that each clause

contains at most one universal variable and the formula is linear. Moreover, Φ is a yes-instance if and only if Φ' is a yes-instance. If Φ' is empty, then Φ is a yes-instance by correctness of the applied transformations. Otherwise, it follows from the properties of Φ' and the proof of Proposition 4.8 that Φ' and, hence, Φ are yes-instances. □

Corollary 4.11. Monotone ∀∃ NAE-3-SAT-(1, 2) *is in* P.

Proof. Let Φ be an instance of Monotone ∀∃ NAE-3-SAT-(1, 2). To decide whether or not Φ is a yes-instance, we apply the following algorithm to Φ.

1. If there exists a clause that contains three distinct universal variables, then stop and return "Φ is a no-instance".
2. For a clause C_j that contains one existential variable, say x, and two distinct universal variables, say u and u', let $C_{j'}$ be the unique clause that contains the second appearance of x. Then remove C_j and turn x in $C_{j'}$ into a new universal variable.
   (I) If $C_{j'}$ now contains three distinct universal variables, stop and return "Φ is a no-instance".
   (II) Otherwise, repeat until there are no clauses with two universal variables.
3. Stop and return "Φ is a yes-instance".

Since each universal variable appears exactly once in Φ and each existential variable appears exactly twice in Φ, it follows that the Boolean formula obtained after each iteration of Step 2 is an instance of Monotone ∀∃ NAE-3-SAT-(1, 2). Therefore, if the algorithm eventually produces a Boolean formula, Φ' say, then Φ' has at most one universal variable in each clause and, by Corollary 4.10, Φ' is a yes-instance. Hence, to see that the algorithm works correctly, it suffices to show that an iteration of Step 2 preserves yes-instances. Suppose that Φ_1 is the quantified Boolean formula at the start of an iteration of Step 2 and $C_j = (x \vee u \vee u')$ is a clause in Φ_1 as described in Step 2. Let β be a truth assignment that nae-satisfies Φ. If $\beta(u) = \beta(u') = F$, then it follows that $\beta(x) = T$. On the other hand, if $\beta(u) = \beta(u') = T$, then this implies that $\beta(x) = F$. It now follows that the Boolean formula, Φ_2 say, obtained by turning x into a universal variable is also a yes-instance. Conversely, by reversing this argument, if Φ_2 is a yes-instance, then Φ_1 is a yes-instance. Thus Step 2 preserves yes-instances. We now establish the corollary by noting that the described algorithm has a running time that is polynomial in the size of Φ. □

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We thank Krzysztof Piecuch and the two anonymous referees for their constructive comments. The third and fourth authors thank the New Zealand Marsden Fund for their financial support.

References

[1] P. Berman, M. Karpinski, A.D. Scott, Approximation hardness of short symmetric instances of MAX-3SAT, Electronic Colloquium on Computational Complexity. Report No. 49, 2003.
[2] M.L. Bonet, S. Linz, K.St. John, The complexity of finding multiple solutions to betweenness and quartet compatibility, IEEE/ACM Trans. Comput. Biol. Bioinform. 9 (2012) 273–285.
[3] S.A. Cook, The complexity of theorem-proving procedures, in: Proceedings of the Third Annual ACM Symposium on Theory of Computing, ACM, New York, 1971, pp. 151–158.
[4] E. Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour, M. Yannakakis, The complexity of multiterminal cuts, SIAM J. Comput. 23 (1994) 864–894.
[5] A. Darmann, J. Döcker, On simplified NP-complete variants of Not-All-Equal 3-SAT and 3-SAT, arXiv preprint, arXiv:1908.04198, 2019.
[6] T. Eiter, G. Gottlob, Note on the complexity of some eigenvector problems, Technical Report CD-TR 95/89, Christian Doppler Laboratory for Expert Systems, TU Vienna, 1995.
[7] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Company, New York, 1979.
[8] I. Haviv, O. Regev, A. Ta-Shma, On the hardness of satisfiability with bounded occurrences in the polynomial-time hierarchy, Theory Comput. 3 (2007) 45–60.
[9] M. Karpinski, K. Piecuch, On vertex coloring without monochromatic triangles, arXiv preprint, arXiv:1710.07132, 2017.
[10] M. Karpinski, K. Piecuch, On vertex coloring without monochromatic triangles, in: F. Fomin, V. Podolskii (Eds.), Computer Science: Theory and Applications, CSR 2018, in: Lecture Notes in Computer Science, vol. 10846, Springer, 2018, pp. 220–231.
[11] S. Porschen, B. Randerath, E. Speckenmeyer, Linear time algorithms for some not-all-equal satisfiability problems, in: E. Giunchiglia, A. Tacchella (Eds.), Theory and Applications of Satisfiability Testing, SAT 2003, in: Lecture Notes in Computer Science, vol. 2919, Springer, 2004, pp. 172–187.
[12] S. Porschen, T. Schmidt, E. Speckenmeyer, A. Wotzlaw, XSAT and NAE-SAT of linear CNF classes, Discrete Appl. Math. 167 (2014) 1–14.
[13] M. Schaefer, Graph Ramsey theory and the polynomial hierarchy, J. Comput. Syst. Sci. 62 (2001) 290–322.
[14] M. Schaefer, C. Umans, Completeness in the polynomial-time hierarchy: a compendium, SIGACT News 33 (2002) 32–49.
[15] T.J. Schaefer, The complexity of satisfiability problems, in: Proceedings of the Tenth Annual ACM Symposium on Theory of Computing, ACM, New York, 1978, pp. 216–226.
[16] L.J. Stockmeyer, The polynomial-time hierarchy, Theor. Comput. Sci. 3 (1976) 1–22.
[17] C.A. Tovey, A simplified NP-complete satisfiability problem, Discrete Appl. Math. 8 (1984) 85–89.

2 Additional Manuscripts

2.1 Monotone 3-SAT-(2,2) is NP-complete

The following manuscript [Döc19] is also available online at the following URL: `https://arxiv.org/abs/1912.08032`.

Monotone 3-Sat-$(2, 2)$ is NP-complete

Janosch Döcker

December 18, 2019

Abstract

We show that Monotone 3-Sat remains NP-complete if (i) each clause contains exactly three distinct variables, (ii) each clause is unique, i.e., there are no duplicates of the same clause, and (iii), amongst the clauses, each variable appears unnegated exactly twice and negated exactly twice. Darmann and Döcker [6] recently showed that this variant of Monotone 3-Sat is either trivial or NP-complete. In the first part of the paper, we construct an unsatisfiable instance which answers one of their open questions (Challenge 1) and places the problem in the latter category.

Then, we adapt gadgets used in the construction to (1) sketch two reductions that establish NP-completeness in a more direct way, and (2), to show that $\forall\exists$ 3-SAT remains Π_2^P-complete for quantified Boolean formulas with the following properties: (a) each clause is monotone (i.e., no clause contains an unnegated and a negated variable) and contains exactly three distinct variables, (b) each universal variable appears exactly once unnegated and exactly once negated, (c) each existential variable appears exactly twice unnegated and exactly twice negated, and (d) the number of universal and existential variables is equal. Furthermore, we show that the variant where (b) is replaced with (b') each universal variable appears exactly twice unnegated and exactly twice negated, and where (a), (c) and (d) are unchanged, is Π_2^P-complete as well. Thereby, we improve upon two recent results by Döcker et al. [8] that establish Π_2^P-completeness of these variants in the non-monotone setting.

We also discuss a special case of Monotone 3-Sat-$(2, 2)$ that corresponds to a variant of Not-All-Equal Sat, and we show that all such instances are satisfiable.

Keywords: Monotone 3-Sat, bounded variable appearances, balanced variable appearances, quantified satisfiability, polynomial hierarchy, computational complexity.

1 Introduction

The satisfiability problem for Boolean formulas is one of the go-to problems when choosing a base problem for polynomial reductions. Indeed, it was the

first problem shown to be NP-complete [5]. The seminal book by Garey and Johnson [9] contains a large list of known NP-complete problems and an extensive introduction into the theoretical foundation of NP-completeness. A very popular variant of the satisfiability problem is 3-SAT, where each clause contains exactly three variables. This problem remains NP-complete even if further restrictions are imposed (see Table 1). In this article, we consider variants of 3-SAT where each clause contains exactly three *distinct* variables. Hence, unless we explicitly say otherwise, the considered instances have this property (the same goes for references regarding 3-SAT variants).

Clauses		Variables				Complexity
unique	monotone	E4	3P1N, 1P3N	3P1N	2P2N	
✓				✓		NP-c [6, Cor. 11]
✓					✓	NP-c [2, Thm. 1]
✓	✓	✓				NP-c [7, Cor. 4]
✓	✓		✓			NP-c [6, Thm. 9]
✓	✓			✓		?
	✓				✓	NP-c [6, Thm. 5]
✓	✓				✓	NP-c (Thm. 1)

Table 1: Overview of complexity results for (monotone) 3-SAT. A checkmark in the "unique" subcolumn means that each clause contains exactly three *distinct* variables. The headings of the subcolumns in the "Variables" column denote the following properties: E4 := each variable appears exactly four times; 3P1N, 1P3N := each variable appears exactly four times and either exactly once unnegated or exactly once negated; 3P1N := each variable appears exactly three times unnegated and once negated; 2P2N := each variable appears exactly twice unnegated and exactly twice negated. In the last column we use the abbreviation NP-c for NP-complete. Note that we only ticked the strongest restrictions, e.g., a checkmark in the 3P1N subcolumn implies a checkmark in the two preceding subcolumns. Moreover, by symmetry we can omit the 1P3N case (identical to 3P1N).

Recently, Darmann and Döcker [6, Cor. 2] showed that for each fixed $k \geq 3$ Monotone 3-Sat is NP-complete if each variable appears exactly k times unnegated und exactly k times negated. Further, they were able to prove that the case $k = 2$ is either trivial or NP-complete. In other words, finding a single unsatisfiable instance is enough to prove that the problem remains NP-complete for $k = 2$. Hence, by constructing an unsatisfiable instance for $k = 2$, we settle this case and thus, one of their open problems (Challenge 1). As the problem is trivial for $k = 1$ [6, p. 32] by a result from Tovey [17, Thm. 2.4], our result closes the last remaining gap for this variant of Monotone 3-Sat.

The gadgets used in the construction of the unsatisfiable instance can

also be used to obtain a more direct way of establishing NP-completeness for the case $k = 2$ (we describe two reductions in this article). Then, we use one of the new gadgets to show that two recent results from Döcker et al. [8, Thm. 3.1 and Thm. 3.2] hold even in the monotone setting. First, we show that $\forall\exists$ 3-SAT remains Π_2^P-complete if (i) each clause is monotone (ii) each universal variable appears exactly once unnegated and exactly once negated, (iii) each existential variable appears exactly twice unnegated and exactly twice negated, and (iv) the number of universal and existential variables is equal. Second, we show that the variant where (ii) is replaced with (ii') each universal variable appears exactly twice unnegated and exactly twice negated, and where (i), (ii) and (iv) are unchanged, is Π_2^P-complete, too.

The article is structured as follows: In Section 2, we recall important definitions and concepts. Then, in Section 3, we construct an unsatisfiable instance of Monotone 3-Sat-$(2, 2)$. Section 4 contains two reductions that can be used to obtain the main result in a more direct way and one of the involved gadgets is subsequently used in Section 5 to show that a restricted variant of $\forall\exists$ 3-SAT remains Π_2^P-complete. The appendix contains proofs of two Lemmas used in Section 3, and a representation of

- a gadget on which several of our results are based, and
- the constructed unsatisfiable instance of Monotone 3-Sat-$(2, 2)$,

which can be used to verify our results with the help of a SAT Solver (e.g., using the PySAT Toolkit [11]).

2 Preliminaries

Let $V = \{x_1, x_2, \ldots, x_n\}$ be a set of n variables. We also write X_1^i to denote the set $\{x_1, x_2, \ldots, x_i\}$ for $i \geq 1$. A positive literal is an element of $\mathcal{L}_+ = V$, a negative literal is an element of $\mathcal{L}_- = \{\overline{x_i} \mid x_i \in V\}$, and the set of literals is denoted by $\mathcal{L} = \mathcal{L}_+ \cup \mathcal{L}_-$. A clause is a subset of $\mathcal{L}$. We say that a clause $C_j \subseteq \mathcal{L}$ is a k-clause if $|C_j| = k$ and C_j is monotone if $C_j \subseteq \mathcal{L}_+$ or $C_j \subseteq \mathcal{L}_-$. A Boolean formula is a set of m clauses

$$\bigcup_{j=1}^{m} \{C_j\}.$$

A Boolean formula is monotone if C_j is monotone for each $j \in \{1, \ldots, m\}$. A truth assignment $\beta \colon V \to \{T, F\}$ maps each variable to the truth value T (True) or F (False). A formula is satisfied for a truth assignment $\beta \colon V \to \{T, F\}$ if β sets at least one literal in each clause true (e.g., a negative literal evaluates to true if β sets the corresponding variable false). If such a truth assignment exists, we say that the formula is satisfiable; otherwise the formula is unsatisfiable. Further, a formula is nae-satisfiable if and only if

there exists a truth assignment β that sets at least one literal in each clause true and at least one false. The main result concerns the following decision problem.

MONOTONE 3-SAT-$(2,2)$
Input. A Boolean formula

$$\bigcup_{j=1}^{m}\{C_j\}$$

over a set $V = \{x_1, x_2, \ldots, x_n\}$ of variables such that (i) each C_j is a unique monotone 3-clause that contains exactly three *distinct* variables, and (ii), amongst the clauses, each variable appears unnegated exactly twice and negated exactly twice.
Question. Does there exist a truth assignment for V such that each clause of the formula is satisfied?

Remark. A monotone 3-clause always contains exactly three distinct variables.

In one instance, we reduce from MONOTONE 3-SAT*-$(2,2)$ [6] which is the variant of MONOTONE 3-SAT-$(2,2)$ where variables may appear more than once in a clause. Note that we can assume that each variable appears at most twice in a given clause, since each clause is monotone and there are only two unnegated and two negated appearances of any variable.
Enforcers. In the construction of an unsatisfiable instance of MONOTONE 3-SAT-$(2,2)$ and the reductions after that, we make use of gadgets that enforce truth assignments to have certain properties (gadgets also go by the name of *enforcers* [2]). As an example, we consider an enforcer introduced by Berman et al. [2, p. 3]:

$$\begin{aligned}\mathcal{S}(\ell_1,\ell_2,\ell_3) =&(\ell_1 \vee \overline{a} \vee b) \wedge (\ell_2 \vee \overline{b} \vee c) \wedge (\ell_3 \vee a \vee \overline{c}) \wedge \\ &(a \vee b \vee c) \wedge (\overline{a} \vee \overline{b} \vee \overline{c}),\end{aligned}$$

where a, b, c are new variables. The enforcer $\mathcal{S}(\ell_1, \ell_2, \ell_3)$ can not be satisfied by a truth assignment β that sets all literals in $\{\ell_1, \ell_2, \ell_3\}$ false. On the other hand, if at least one literal in $\{\ell_1, \ell_2, \ell_3\}$ evaluates to true, we can find truth values for the variables a, b, c such that all clauses of the enforcer are satisfied. In other words, $\mathcal{S}(\ell_1, \ell_2, \ell_3)$ simulates a clause but has the advantage that we can allow duplicates since each literal in $\{\ell_1, \ell_2, \ell_3\}$ ends up in a different clause (cf. [2, p. 3]). Note that this enforcer is not monotone. In this article, we construct a monotone version with 99 new variables and 133 clauses (instead of 3 new variables and 5 clauses in the setting above).

3 Construction of an unsatisfiable instance of Monotone 3-Sat-$(2,2)$

In this section, we construct an unsatisfiable instance of Monotone 3-Sat-$(2,2)$. First, we construct an enforcer $\mathcal{M}^{(i)}(u_1, \overline{u_2}, \overline{u_3})$ that, intuitively, consists of three smaller gadgets. The first gadget is only satisfiable by truth assignments for the corresponding variables that can be placed in one of two categories. Depending on the category of the truth assignment (and the restrictions imposed by them), it is not possible to find a truth assignment for the variables contained in the second or the third gadget such that all clauses are satisfied. The second and the third gadget (see Lemmas 1 and 2) have been found via computer search. The basic idea of the implemented Python code is the following: start with a collection of random candidates and try to improve them by swapping literals of differenct clauses, where this operation preserves the properties of an instance of Monotone 3-Sat-$(2,2)$ (a reduction in the number of satisfying truth assignments is considered an improvement here). We used the PySAT Toolkit [11] to (1) obtain a list of all satisfying truth assignments for a given collection of clauses, and (2), to verify some of our constructions (see appendix). Finally, we combine several instances of the enforcer $\mathcal{M}^{(i)}(u_1, \overline{u_2}, \overline{u_3})$ to obtain an unsatisfiable instance of Monotone 3-Sat-$(2,2)$.

We start with the construction of the first gadget. Let $\mathcal{F}_2$ denote the set consisting of the following 2-clauses:

1. $\{x_1, x_2\}$
2. $\{\overline{x_2}, \overline{x_3}\}$
3. $\{\overline{x_2}, \overline{x_4}\}$

Further let $\mathcal{F}_3$ denote the set consisting of the following 3-clauses:

4. $\{\overline{x_3}, \overline{x_5}, \overline{x_6}\}$
5. $\{\overline{x_4}, \overline{x_5}, \overline{x_6}\}$
6. $\{x_5, x_7, x_8\}$
7. $\{x_6, x_7, x_8\}$
8. $\{\overline{x_7}, \overline{z_1}, \overline{z_2}\}$
9. $\{\overline{x_7}, \overline{z_3}, \overline{z_4}\}$
10. $\{\overline{x_8}, \overline{z_1}, \overline{z_2}\}$
11. $\{\overline{x_8}, \overline{z_3}, \overline{z_4}\}$

First, the 2-clauses in $\mathcal{F}_2$ are equivalent to the implications

$$\overline{x_1} \Rightarrow x_2, \quad x_2 \Rightarrow \overline{x_3}, \quad x_2 \Rightarrow \overline{x_4}.$$

Hence, if $\beta(x_1) = F$ then $\beta(x_2) = T$ and consequently $\beta(x_3) = \beta(x_4) = F$. Next, we introduce a set of clauses for which no satisfying truth assignment exists that sets $\beta(x_3) = F$ and $\beta(x_4) = F$. To this end, let $\mathcal{G}$ be the set consisting of the following 3-clauses:

12. $\{x_3, y_1, y_2\}$
13. $\{x_3, y_3, y_4\}$
14. $\{x_4, y_5, y_6\}$
15. $\{x_4, y_7, y_8\}$
16. $\{y_1, y_4, y_7\}$
17. $\{y_2, y_5, y_9\}$
18. $\{y_3, y_8, y_9\}$
19. $\{\overline{y_1}, \overline{y_5}, \overline{y_8}\}$
20. $\{\overline{y_1}, \overline{y_6}, \overline{y_9}\}$
21. $\{\overline{y_2}, \overline{y_3}, \overline{y_6}\}$
22. $\{\overline{y_2}, \overline{y_4}, \overline{y_8}\}$
23. $\{\overline{y_3}, \overline{y_5}, \overline{y_7}\}$

24. $\{\overline{y_4}, \overline{y_7}, \overline{y_9}\}$

Note that for $\beta(x_3) = \beta(x_4) = F$, omitting the appearances of x_3 and x_4 in $\mathcal{G}$ has no effect on the satisfiability. We deferred the proof that the resulting instance is unsatisfiable to the appendix (see Lemma 1). Now, for at least one $x_i \in \{x_3, x_4\}$ we have $\beta(x_i) = T$ and we may assume that $\beta(x_1) = T$ and $\beta(x_2) = F$. Next, by clauses 4 and 5 we have $\beta(x_j) = F$ for at least one $x_j \in \{x_5, x_6\}$. Then, clauses 6 and 7 imply $\beta(x_k) = T$ for at least one $x_k \in \{x_7, x_8\}$. Hence, by clauses 8, 9, 10 and 11 we get two clauses $\{F, \overline{z_1}, \overline{z_2}\}$ and $\{F, \overline{z_3}, \overline{z_4}\}$ which is equivalent to $\{\overline{z_1}, \overline{z_2}\}$ and $\{\overline{z_3}, \overline{z_4}\}$. Recalling that $\beta(x_1) = T$ and $\beta(x_2) = F$, the first three clauses in the following set $\mathcal{H}$ of 3-clauses evaluate to $\{F, \overline{z_5}, \overline{z_6}\}$, $\{F, \overline{z_7}, \overline{z_8}\}$ and $\{F, z_7, z_{15}\}$, respectively.

25. $\{\overline{x_1}, \overline{z_5}, \overline{z_6}\}$
26. $\{\overline{x_1}, \overline{z_7}, \overline{z_8}\}$
27. $\{x_2, z_7, z_{15}\}$
28. $\{z_1, z_6, z_8\}$
29. $\{z_1, z_{11}, z_{12}\}$
30. $\{z_2, z_6, z_8\}$
31. $\{z_2, z_{11}, z_{12}\}$
32. $\{z_3, z_5, z_9\}$
33. $\{z_3, z_{13}, z_{14}\}$
34. $\{z_4, z_5, z_{14}\}$
35. $\{z_4, z_9, z_{10}\}$
36. $\{z_7, z_{10}, z_{13}\}$
37. $\{\overline{z_5}, \overline{z_8}, \overline{z_{15}}\}$
38. $\{\overline{z_6}, \overline{z_7}, \overline{z_9}\}$
39. $\{\overline{z_9}, \overline{z_{11}}, \overline{z_{13}}\}$
40. $\{\overline{z_{10}}, \overline{z_{11}}, \overline{z_{14}}\}$
41. $\{\overline{z_{10}}, \overline{z_{12}}, \overline{z_{14}}\}$
42. $\{\overline{z_{12}}, \overline{z_{13}}, \overline{z_{15}}\}$

Now, the inferred 2-clauses

$$\{\overline{z_1}, \overline{z_2}\},\ \{\overline{z_3}, \overline{z_4}\},\ \{\overline{z_5}, \overline{z_6}\},\ \{\overline{z_7}, \overline{z_8}\} \text{ and } \{z_7, z_{15}\}$$

in conjunction with the clauses $\mathcal{H} \setminus \{\{\overline{x_1}, \overline{z_5}, \overline{z_6}\}, \{\overline{x_1}, \overline{z_7}, \overline{z_8}\}, \{x_2, z_7, z_{15}\}\}$ are unsatisfiable (again, the proof is deferred to the appendix; see Lemma 2).

Hence, the constructed set of 42 clauses

$$\mathcal{M} := \{\{x_1, x_2\}, \{\overline{x_2}, \overline{x_3}\}, \{\overline{x_2}, \overline{x_4}\}\} \cup \mathcal{F}_3 \cup \mathcal{G} \cup \mathcal{H}$$

over the set of variables $V := X_1^8 \cup Y_1^9 \cup Z_1^{15}$ is unsatisfiable. We note that each literal appears at most twice in $\mathcal{M}$. The only variables that appear less than 4 times are x_1, x_5, x_6, y_6 and z_{15} each of which appear once unnegated and twice negated. Consider the following enforcer

$$\mathcal{M}^{(i)}(u_1, \overline{u_2}, \overline{u_3}) := \{\{x_1^i, x_2^i, u_1^i\}, \{\overline{x_2^i}, \overline{x_3^i}, \overline{u_2^i}\}, \{\overline{x_2^i}, \overline{x_4^i}, \overline{u_3^i}\}\} \cup \mathcal{F}_3^i \cup \mathcal{G}^i \cup \mathcal{H}^i,$$

where $\mathcal{F}_3^i, \mathcal{G}^i, \mathcal{H}^i$ is obtained from $\mathcal{F}_3, \mathcal{G}, \mathcal{H}$ by replacing each variable, say v, with v^i (e.g. z_1 is replaced with z_1^i). The enforcer $\mathcal{M}^{(i)}(u_1, \overline{u_2}, \overline{u_3})$ has two properties that we use to construct an unsatisfiable instance of MONOTONE 3-SAT-$(2, 2)$. First, as alluded to in Section 2, we can deal with duplicates

in a clause $\{u_1, \overline{u_2}, \overline{u_3}\}$, i.e., if $\overline{u_2} = \overline{u_3}$ and, second, we can transform a mixed clause into a monotone clause. Further, we obtain a second enforcer $\overline{\mathcal{M}}^{(i)}(\overline{u_1}, u_2, u_3)$ by negating every literal in $\mathcal{M}^{(i)}(u_1, \overline{u_2}, \overline{u_3})$.

It is easy to verify that the following collection of clauses is unsatisfiable (we do not use set notation here since the clauses contain duplicates):

$$(\overline{a} \vee \overline{d} \vee \overline{f}) \wedge (b \vee d \vee e) \wedge (e \vee \overline{b} \vee \overline{b}) \wedge (d \vee \overline{f} \vee \overline{c}) \\ \wedge (a \vee \overline{c} \vee \overline{e}) \wedge (\overline{e} \vee c \vee c) \wedge (\overline{d} \vee a \vee b) \wedge (\overline{a} \vee f \vee f).$$

Now, we are in a position to construct an unsatisfiable instance $\mathcal{U}$ of Monotone 3-Sat-$(2,2)$:

$$\begin{aligned}\mathcal{U} := \{\{\overline{a}, \overline{d}, \overline{f}\}, \{b, d, e\}\} &\cup \mathcal{M}^{(1)}(e, \overline{b}, \overline{b}) \cup \mathcal{M}^{(2)}(d, \overline{f}, \overline{c}) \cup \mathcal{M}^{(3)}(a, \overline{c}, \overline{e}) \\ &\cup \overline{\mathcal{M}}^{(4)}(\overline{e}, c, c) \cup \overline{\mathcal{M}}^{(5)}(\overline{d}, a, b) \cup \overline{\mathcal{M}}^{(6)}(\overline{a}, f, f) \\ &\cup \bigcup_{i \in \{1,5,6\}} \{\{x_i^1, x_i^2, x_i^3\}, \{\overline{x_i^4}, \overline{x_i^5}, \overline{x_i^6}\}\} \\ &\cup \{\{y_6^1, y_6^2, y_6^3\}, \{\overline{y_6^4}, \overline{y_6^5}, \overline{y_6^6}\}, \{z_{15}^1, z_{15}^2, z_{15}^3\}, \{\overline{z_{15}^4}, \overline{z_{15}^5}, \overline{z_{15}^6}\}\}\end{aligned}$$

Proposition 1. *There is an unsatisfiable instance of* Monotone 3-Sat-$(2,2)$ *with 198 variables and 264 clauses.*

Now, with the result from Darmann and Döcker [6, Thm. 4] we get the following theorem as a consequence of the existence of an unsatisfiable instance of Monotone 3-Sat-$(2,2)$.

Theorem 1. Monotone 3-Sat-$(2,2)$ *is NP-complete.*

Since Monotone 3-Sat-(k,k) is known to be NP-complete for each fixed $k \geq 3$ [6, Cor. 2], we get the following corollary.

Corollary 1. Monotone 3-Sat-(k,k) *is NP-complete for each fixed* $k \geq 2$.

A special case that is always satisfiable

We briefly consider instances of Monotone 3-Sat-(k,k) with the property that for each clause $C = \{x, y, z\}$ the instance also contains $\overline{C} = \{\overline{x}, \overline{y}, \overline{z}\}$. Noting that this is Monotone NAE 3-SAT with exactly k appearances of each variable, it follows that this problem is hard for $k = 4$ (see [6, Cor. 1]).
Remark. In the context of NAE SAT monotone means that negations are completely absent. This is no restriction since the two clauses $\{x, y, z\}$ and $\{\overline{x}, \overline{y}, \overline{z}\}$ impose exactly the same restrictions in this setting.

Porschen et al. [14, Thm. 4] show that for $k = 3$ the corresponding Monotone NAE 3-SAT problem can be solved in linear time. In particular, they show that such an instance is nae-satisfiable if and only if the

variable graph has no component isomorphic to the complete graph K_7 on 7 vertices [14, Cor. 4]. The variable graph (cf., e.g., [12, p. 2] and [14, p. 175]) of an instance of NAE 3-SAT (resp. 3-SAT), contains a vertex for each variable and an edge between two vertices if the corresponding variables appear together in some clause of the instance. For example, the variable graph of the following instance is isomorphic to the K_7 and is, thus, not nae-satisfiable:

$$\begin{aligned}\mathcal{U}_{\text{NAE}} = \{&\{x_1, x_2, x_7\}, \{x_1, x_3, x_6\}, \{x_1, x_4, x_5\},\\ &\{x_2, x_3, x_4\}, \{x_2, x_5, x_6\}, \{x_3, x_5, x_7\}, \{x_4, x_6, x_7\}\}.\end{aligned}$$

Let us now consider $k = 2$. We show that the property mentioned above leads to a trivial instance of Monotone NAE 3-SAT with exactly two appearances of each variable and, hence, Monotone 3-Sat-$(2, 2)$ is always satisfiable if clauses always appear in pairs $\{C, \overline{C}\}$. Jain [12, p. 2] observed that instances of Monotone NAE 3-SAT are in P if the variable graph is 4-colorable. Indeed, such instances are trivial since we can associate each truth value with exactly two colors such that a 4-coloring corresponds to a truth assignment that sets at least one variable of each clause false and at least one true (since each clause contains exactly three distinct variables, all clauses are satisfied). Pilz [13, Thm. 12] used an approach based on this idea to show that every instance of Planar SAT in which each clause contains at least three negated or at least three unnegated appearances of distinct variables is satisfiable. He transformed the incidence graph of the formula into a certain subgraph of the variable graph, showed that this transformation preserves planarity, and then applied the Four Color Theorem [1] to obtain a 4-coloring. Hence, all we need to show is that the variable graph of an instance of Monotone NAE 3-SAT where each variable appears exactly twice is always 4-colorable. First, observe that a vertex corresponding to a variable x in the variable graph of such an instance has degree 2 if and only if x is contained in two clauses

$$\{x, y, z\}, \{x, y, z\}$$

for some variables y, z such that x, y, z are pairwise distinct (otherwise x has at least three neighbours). Such clauses can simply be removed as it is trivial to nae-satisfy them. Hence, we can assume that the variable graph has no cycles and, in particular, no cycles of odd length. Furthermore, it is easy to see that each instance has a number of variables that is divisible by 3 and, hence, each connected component in the variable graph contains a number of vertices that is a multiple of 3. Now, there is no component with 3 vertices since we already removed the clauses that would result in such a subgraph (the K_3 is a cycle of odd length). Noting that the degree of each vertex is bounded by 4, we conclude that no component with 6 or more vertices is a complete graph. Consequently, we can assume that the

variable graph of an instance of MONOTONE NAE 3-SAT does not contain a component that is a complete graph or a cycle of odd length. Hence, the variable graph is 4-colorable by Brooks' Theorem [4] and we get the following theorem.

Theorem 2. *All instances of* MONOTONE NAE 3-SAT*, where each variable appears exactly twice, are satisfiable.*

Corollary 2. *Let* $\mathcal{I} = \bigcup_{j=1}^{m} \{C_j\}$ *be an instance of* MONOTONE 3-SAT-$(2,2)$*. If the instance* $\mathcal{I}$ *has the property*

$$C_j \in \mathcal{I} \Rightarrow \overline{C_j} \in \mathcal{I},$$

where $\overline{C_j}$ *is obtained from* C_j *by negating each literal, then* $\mathcal{I}$ *is satisfiable.*

4 More ways to obtain the main result

It is also possible to show NP-hardness of MONOTONE 3-SAT-$(2,2)$ by reduction from MONOTONE 3-SAT*-$(2,2)$, for which NP-hardness was established by Darmann and Döcker [6, Thm. 5]. To this end, let

$$\mathcal{N}^{(i)}(\overline{u_i}, \overline{u_i}) := \{\{x_1^i, x_2^i\}, \{\overline{x_2^i}, \overline{x_3^i}, \overline{u_i}\}, \{\overline{x_2^i}, \overline{x_4^i}, \overline{u_i}\}\} \cup \mathcal{F}_3^i \cup \mathcal{G}^i \cup \mathcal{H}^i,$$

By construction, this set of clauses is not satisfied for any truth assignment β that sets $\beta(u_1) = T$. Now, we can construct another enforcer which has exactly three positive 2-clauses:

$$\begin{aligned}
\mathcal{S}(v_1, v_2, v_3) = &\{\{x_1^1, x_2^1, v_1\}, \{x_1^2, x_2^2, v_2\}, \{x_1^3, x_2^3, v_3\}\} \\
&\cup \mathcal{N}^{(1)}(\overline{u_1}, \overline{u_1}) \setminus \{\{x_1^1, x_2^1\}\} \cup \mathcal{N}^{(2)}(\overline{u_2}, \overline{u_2}) \setminus \{\{x_1^2, x_2^2\}\} \\
&\cup \mathcal{N}^{(3)}(\overline{u_3}, \overline{u_3}) \setminus \{\{x_1^3, x_2^3\}\} \cup \{\{u_1, u_2, u_3\}\} \\
&\cup \bigcup_{i \in \{1,5,6\}} \{\{x_i^1, x_i^2, x_i^3\}\} \\
&\cup \{\{y_6^1, z_{15}^1, u_1\}, \{y_6^2, z_{15}^2, u_2\}, \{y_6^3, z_{15}^3, u_3\}\}
\end{aligned}$$

Let $V_{\mathcal{S}}$ denote the set of variables that appear in $\mathcal{S}(v_1, v_2, v_3)$. Each variable $v \in V_{\mathcal{S}} \setminus \{v_1, v_2, v_3\}$ appears exactly twice unnegated and twice negated. For each instance of $\mathcal{S}(v_1, v_2, v_3)$, we create new variables $V_{\mathcal{S}} \setminus \{v_1, v_2, v_3\}$ (we omitted additional indices to improve readability). By negating each literal in $v \in V_{\mathcal{S}} \setminus \{v_1, v_2, v_3\}$ we obtain a second enforcer $\overline{\mathcal{S}}(\overline{v_1}, \overline{v_2}, \overline{v_3})$. By construction, the enforcer $\mathcal{S}(v_1, v_2, v_3)$ has no satisfying truth assignment β with $\beta(v_1) = \beta(v_2) = \beta(v_3) = F$. On the other hand, if $\beta(v_i) = T$ for at least one $v_i \in \{v_1, v_2, v_3\}$, we can assign truth values to the remaining variables of $\mathcal{S}(v_1, v_2, v_3)$ such that all clauses of the enforcer are satisfied (this is straightforward to verify with a SAT solver).

Given an instance $\mathcal{I}$ of Monotone 3-Sat*-$(2,2)$, we replace each positive (resp. negative) clause with a duplicate, say $(p \vee p \vee q)$ (resp. $(\overline{p} \vee \overline{p} \vee \overline{q})$), by an enforcer $\mathcal{S}(p,p,q)$ (resp. $\overline{\mathcal{S}}(\overline{p},\overline{p},\overline{q})$). The result is an instance of Monotone 3-Sat-$(2,2)$ that is satisfiable if and only if $\mathcal{I}$ is satisfiable.

Yet another approach is the following. We can also reduce from 3-Sat-$(2,2)$, for which NP-hardness was established by Berman et al. [2, Thm. 1], and use an extended version of the enforcers $\mathcal{M}^{(i)}(u_1, \overline{u_2}, \overline{u_3})$ and $\overline{\mathcal{M}}^{(i)}(\overline{u_1}, u_2, u_3)$ to transform mixed clauses that may be present in a given instance into monotone clauses. To this end, consider

$$\begin{aligned}\mathfrak{M}_j :=& \mathcal{M}^{(3j)}(u_1, \overline{u_2}, \overline{u_3}) \cup \mathcal{M}^{(3j+1)}(u_4, \overline{u_5}, \overline{u_6}) \cup \mathcal{M}^{(3j+2)}(u_7, \overline{u_8}, \overline{u_9}) \\ & \cup \{\{x_1^{3j}, x_5^{3j}, x_6^{3j}\}, \{y_6^{3j}, z_{15}^{3j}, x_1^{3j+1}\}, \{x_5^{3j+1}, x_6^{3j+1}, y_6^{3j+1}\}\} \\ & \cup \{\{z_{15}^{3j+1}, x_1^{3j+2}, x_5^{3j+2}\}, \{x_6^{3j+2}, y_6^{3j+2}, z_{15}^{3j+2}\}\}\end{aligned}$$

Combining three instances of the enforcer $\mathcal{M}^{(i)}(u_1, \overline{u_2}, \overline{u_3})$ in this way has the advantage that each instance of $\mathfrak{M}_j$ introduces only variables that appear exactly twice unnegated and twice negated. A second enforcer $\overline{\mathfrak{M}_j}$ is again obtained by negating all literals. In order to be able to use these enforcers to replace all mixed clauses in a given instance of 3-Sat-$(2,2)$ we need the number of clauses with a positive (resp. negative) duplicate to be divisible by 3. This can be achieved by simply taking three copies of the original instance on pairwise disjoint sets of variables. With the help of a SAT solver it is easy to verify that $\mathfrak{M}_j$ has only satisfying truth assignments that set at least one literal in each of $\{u_1, \overline{u_2}, \overline{u_3}\}$, $\{u_4, \overline{u_5}, \overline{u_6}\}$ and $\{u_7, \overline{u_8}, \overline{u_9}\}$ true.

5 On a restricted variant of $\forall\exists$ 3-SAT

In this section, we consider the monotone variant of the following problem and show that it remains Π_2^P-complete in restricted settings. We assume the reader is familiar with basic concepts regarding the polynomial hierarchy and, in particular, with the complexity class Π_2^P. For an in-depth introduction to this theory, we refer to Stockmeyer [16] (see [15] for a list containing many problems that are known to be Π_2^P-complete). We use the same notation defined in [8], e.g., for $i \leq i'$, let

$$X_i^{i'} := \{x_i, x_{i+1}, \ldots, x_{i'}\},$$

and

$$QX_i^{i'} := Qx_i Qx_{i+1} \cdots Qx_{i'}, \quad Q \in \{\forall, \exists\}.$$

Let s_1, s_2, t_1, t_2 be four non-negative integers.

BALANCED $\forall\exists$ 3-SAT-(s_1, s_2, t_1, t_2) [8, p. 6f]
Input. A quantified Boolean formula

$$\forall X_1^p \exists X_{p+1}^n \bigcup_{j=1}^{m} \{C_j\}$$

over a set $V = \{x_1, x_2, \ldots, x_n\}$ of variables such that (i) $n = 2p$, (ii) each C_j is a 3-clause that contains three *distinct* variables, and (iii), amongst the clauses, each universal variable appears unnegated exactly s_1 times and negated exactly s_2 times, and each existential variable appears unnegated exactly t_1 times and negated exactly t_2 times.
Question. For every truth assignment for $\{x_1, x_2, \ldots, x_p\}$, does there exist a truth assignment for $\{x_{p+1}, x_{p+2}, \ldots, x_n\}$ such that each clause of the formula is satisfied?

Recently, Döcker et al. [8, Thm. 3.1 and Thm. 3.2] showed that BALANCED $\forall\exists$ 3-SAT-$(2,2,2,2)$ and BALANCED $\forall\exists$ 3-SAT-$(1,1,2,2)$ are both Π_2^P-complete. We use the gadgets $\mathfrak{M}_j$ and $\overline{\mathfrak{M}_j}$ to show that these results also hold for instances, where each clause is monotone (i.e., each clause consists of exactly three unnegated variables or exactly three negated variables, respectively). Since the transformation is virtually identical for both cases, we focus on the second result and mention the necessary adaption to obtain the first result. Consider an instance of BALANCED $\forall\exists$ 3-SAT-$(1,1,2,2)$, i.e. a quantified Boolean formula

$$\Phi = \forall X_1^p \exists X_{p+1}^n \varphi,$$

with $\varphi = \bigcup_{j=1}^{m} \{C_j\}$. Let φ' and φ'' be the sets of clauses obtained from φ by replacing x_i with y_i and z_i, respectively (y_i and z_i are distinct new variables). It is easy to see that the following quantified Boolean formula is a yes-instance if and only if Φ is a yes-instance.

$$\Phi' = \forall (X_1^p \cup Y_1^p \cup Z_1^p) \exists (X_{p+1}^n \cup Y_{p+1}^n \cup Z_{p+1}^n)(\varphi \cup \varphi' \cup \varphi'').$$

Now, the number of mixed clauses with two negative (resp. positive) literals is divisible by 3. Hence, we can replace such clauses in triples using $\mathfrak{M}_j$ and $\overline{\mathfrak{M}_j}$, respectively. For example, we replace first triple of mixed clauses, e.g.,

$$\{x_i, \overline{x_j}, \overline{x_k}\}, \{y_i, \overline{y_j}, \overline{y_k}\}, \{z_i, \overline{z_j}, \overline{z_k}\},$$

with the following collection of monotone clauses

$$\begin{aligned}\mathfrak{M}_0 :=& \mathcal{M}^{(0)}(x_i, \overline{x_j}, \overline{x_k}) \cup \mathcal{M}^{(1)}(y_i, \overline{y_j}, \overline{y_k}) \cup \mathcal{M}^{(2)}(z_i, \overline{z_j}, \overline{z_k}) \\ & \cup \{\{x_1^0, x_5^0, x_6^0\}, \{y_6^0, z_{15}^0, x_1^1\}, \{x_5^1, x_6^1, y_6^1\}\} \\ & \cup \{\{z_{15}^1, x_1^2, x_5^2\}, \{x_6^2, y_6^2, z_{15}^2\}\}.\end{aligned}$$

Note that we introduce $3 \cdot 32 = 96$ new existential variables with each instance of $\mathfrak{M}_j$ or $\overline{\mathfrak{M}_j}$. By construction, the resulting quantified Boolean formula Φ'' is a yes-instance if and only if Φ' is a yes-instance. Since we introduced a number of existential variables that is divisible by 3, we can use multiple instances (each with new variables) of the following quantified enforcer introduced by Döcker et al. [8, p. 9]

$$Q^3 = \{u, r, a\}, \{\overline{u}, \overline{b}, \overline{a}\}, \{v, q, b\}, \{\overline{v}, \overline{r}, \overline{a}\}, \{w, a, b\}, \{\overline{w}, \overline{q}, \overline{b}\},$$

where u, v, w, q, r are universal variables and a, b are existential variables, to obtain a quantified Boolean formula with the same number of existential and universal variables. Since Q^3 is a yes-instance [8, Lem. 3.2], the resulting quantified Boolean formula is a yes-instance if and only if Φ'' is a yes-instance. Noting that the transformation is polynomial, we get the following theorem.

Theorem 3. Balanced Monotone $\forall\exists$ 3-SAT-$(1, 1, 2, 2)$ *is* Π_2^P*-complete.*

The only difference in the reduction from Balanced $\forall\exists$ 3-SAT-$(2, 2, 2, 2)$ to obtain the first result is the last step. Here, we are not able to use the existing quantified enforcer Q^1 given in [8, p. 9], since it introduces mixed clauses. For this reason, we adapt the quantified enforcer Q^3 as follows

$$\begin{aligned} Q^1_{\text{mon}} = &\{u, r, a\}, \{\overline{u}, \overline{b}, \overline{a}\}, \{v, q, b\}, \{\overline{v}, \overline{r}, \overline{a}\}, \{w, a, b\}, \{\overline{w}, \overline{q}, \overline{b}\}, \\ &\{u, r, c\}, \{\overline{u}, \overline{d}, \overline{c}\}, \{v, q, d\}, \{\overline{v}, \overline{r}, \overline{c}\}, \{w, c, d\}, \{\overline{w}, \overline{q}, \overline{d}\}, \end{aligned}$$

where u, v, w, q, r are universal variables and a, b, c, d are existential variables. Intuitively, we use two instances of Q^3 on the same universal variables but with different existential variables. Consider an arbitrary truth assignment β for the universal variables. Since Q^3 is a yes-instance we can find truth values $\beta(a)$ and $\beta(b)$ such that the top eight clauses in Q^1_{mon} are satisfied. Hence, for $\beta(c) = \beta(a)$ and $\beta(d) = \beta(b)$ we can satisfy all clauses in Q^1_{mon}. In other words, Q^1_{mon} is a yes-instance of $\forall\exists$ 3-SAT that introduces 5 universal variables but only 4 existential variables (each of which appears exactly twice unnegated and exactly twice negated). Now, we can use multiple instances (each with new variables) of Q^1_{mon} to obtain a formula with the same number of existential and universal variables. Thus, we get the following theorem.

Theorem 4. Balanced Monotone $\forall\exists$ 3-SAT-$(2, 2, 2, 2)$ *is* Π_2^P*-complete.*

References

[1] K. Appel, and W. Haken (1989). Every Planar Map is Four Colorable. *In* Contemporary Mathematics, vol. 98, American Mathematical Soc.

[2] P. Berman, M. Karpinski, and A. D. Scott (2003). Approximation hardness of short symmetric instances of MAX-3SAT. *Electronic Colloquium on Computational Complexity*, Report No. 49.

[3] A. Biere (2016). Splatz, Lingeling, Plingeling, Treengeling, YalSAT entering the SAT competition 2016. *In* Proceedings of SAT Competition 2016, pp. 44–45.

[4] R. L. Brooks (1941). On colouring the nodes of a network. *Mathematical Proceedings of the Cambridge Philosophical Society*, 37(2).

[5] S. A. Cook (1971). The complexity of theorem-proving procedures. *In* Proceedings of the Third Annual ACM Symposium on Theory of Computing, pp. 151–158.

[6] A. Darmann and J. Döcker (2019). On simplified NP-complete variants of Not-All-Equal 3-SAT and 3-SAT. arXiv preprint arXiv:1908.04198.

[7] A. Darmann, J. Döcker, B. Dorn (2018). The Monotone Satisfiability Problem with Bounded Variable Appearances. *International Journal of Foundations of Computer Science*, 29(6):979–993.

[8] J. Döcker, B. Dorn, S. Linz, C. Semple (2019). Placing quantified variants of 3-SAT and Not-All-Equal 3-SAT in the polynomial hierarchy. arXiv preprint arXiv:1908.05361.

[9] M. R. Garey and D. S. Johnson (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Company.

[10] M. J. H. Heule, W. A. Hunt, N. Wetzler (2013). Trimming while checking clausal proofs. *In* 2013 Formal Methods in Computer-Aided Design. IEEE (2013), pp. 181–188.

[11] A. Ignatiev, A. Morgado, J. Marques-Silva (2018). PySAT: A Python Toolkit for Prototyping with SAT Oracles. *In* O. Beyersdorff, C. Wintersteiger (eds) Theory and Applications of Satisfiability Testing – SAT 2018. SAT 2018. Lecture Notes in Computer Science, vol. 10929, Springer, pp. 428–437.

[12] P. Jain (2010). On a variant of Monotone NAE-3SAT and the Triangle-Free Cut problem. arXiv preprint arXiv:1003.3704.

[13] A. Pilz. Planar 3-SAT with a clause/variable cycle. *Discrete Mathematics & Theoretical Computer Science*, 21(3).

[14] S. Porschen, B. Randerath, E. Speckenmeyer (2004). Linear time algorithms for some not-all-equal satisfiability problems. *In* E. Giunchiglia,

A. Tacchella (eds) Theory and Applications of Satisfiability Testing. SAT 2003. Lecture Notes in Computer Science, vol. 2919, Springer, pp. 172–187.

[15] M. Schaefer and C. Umans (2002). Completeness in the polynomial-time hierarchy: A compendium. *SIGACT News*, 33:32–49.

[16] L. J. Stockmeyer (1976). The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22.

[17] C. A. Tovey (1984). A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8:85–89.

A Proofs

We used the PySAT Toolkit [11] in the proofs of Lemmas 1 and 2 to obtain a DRUP proof [10] which is a certificate of unsatisfiablity. Here, we use the solver Lingeling [3] included in the PySAT Toolkit since it is one of the solvers that provide the option to return such a certificate of unsatisfiability.

Lemma 1. *The following set of clauses over variables Y_1^9 is unsatisfiable.*

1. $\{y_1, y_2\}$	*5.* $\{y_1, y_4, y_7\}$	*9.* $\{\overline{y_1}, \overline{y_6}, \overline{y_9}\}$	*13.* $\{\overline{y_4}, \overline{y_7}, \overline{y_9}\}$
2. $\{y_3, y_4\}$	*6.* $\{y_2, y_5, y_9\}$	*10.* $\{\overline{y_2}, \overline{y_3}, \overline{y_6}\}$	
3. $\{y_5, y_6\}$	*7.* $\{y_3, y_8, y_9\}$	*11.* $\{\overline{y_2}, \overline{y_4}, \overline{y_8}\}$	
4. $\{y_7, y_8\}$	*8.* $\{\overline{y_1}, \overline{y_5}, \overline{y_8}\}$	*12.* $\{\overline{y_3}, \overline{y_5}, \overline{y_7}\}$	

Proof. We can use the following Python code to obtain a DRUP proof.

```
from pysat.solvers import Lingeling
cnf = [[1, 2], [3, 4], [5, 6], [7, 8], [1, 4, 7], [2, 5, 9],
       [3, 8, 9], [-1, -5, -8], [-1, -6, -9], [-2, -3, -6],
       [-2, -4, -8], [-3, -5, -7], [-4, -7, -9]]
solver = Lingeling(bootstrap_with=cnf, with_proof=True)
print solver.solve()
print solver.get_proof()
solver.delete()
```

Output of the program:

```
False
['-8 -7 -5 0', '-8 9 5 0', '-5 -8 0', 'd -1 -5 -8 0', '-8 9 0',
    'd 5 -8 9 0', '9 0', '-4 -2 0', 'd -8 -4 -2 0', '-5 -3 0', 'd
     -7 -5 -3 0', '-3 -2 0', 'd -6 -3 -2 0', '-2 0', '1 0', '-6
    0', '5 0', '-8 0', '-3 0', '7 0', '4 0', '0']
```

□

Lemma 2. *The following set of clauses over variables Z_1^{15} is unsatisfiable.*

1. $\{\overline{z_1}, \overline{z_2}\}$
2. $\{\overline{z_3}, \overline{z_4}\}$
3. $\{\overline{z_5}, \overline{z_6}\}$
4. $\{\overline{z_7}, \overline{z_8}\}$
5. $\{z_7, z_{15}\}$
6. $\{z_1, z_6, z_8\}$
7. $\{z_1, z_{11}, z_{12}\}$
8. $\{z_2, z_6, z_8\}$
9. $\{z_2, z_{11}, z_{12}\}$
10. $\{z_3, z_5, z_9\}$
11. $\{z_3, z_{13}, z_{14}\}$
12. $\{z_4, z_5, z_{14}\}$
13. $\{z_4, z_9, z_{10}\}$
14. $\{z_7, z_{10}, z_{13}\}$
15. $\{\overline{z_5}, \overline{z_8}, \overline{z_{15}}\}$
16. $\{\overline{z_6}, \overline{z_7}, \overline{z_9}\}$
17. $\{\overline{z_9}, \overline{z_{11}}, \overline{z_{13}}\}$
18. $\{\overline{z_{10}}, \overline{z_{11}}, \overline{z_{14}}\}$
19. $\{\overline{z_{10}}, \overline{z_{12}}, \overline{z_{14}}\}$
20. $\{\overline{z_{12}}, \overline{z_{13}}, \overline{z_{15}}\}$

Proof. We can use the following Python code to obtain a DRUP proof.

```
from pysat.solvers import Lingeling
cnf = [[-1, -2], [-3, -4], [-5, -6], [-7, -8], [7, 15],
       [1, 6, 8], [1, 11, 12], [2, 6, 8], [2, 11, 12],
       [3, 5, 9], [3, 13, 14], [4, 5, 14], [4, 9, 10],
       [7, 10, 13], [-5, -8, -15], [-6, -7, -9],
       [-9, -11, -13], [-10, -11, -14], [-10, -12, -14],
       [-12, -13, -15]]
solver = Lingeling(bootstrap_with=cnf, with_proof=True)
print solver.solve()
print solver.get_proof()
solver.delete()
```

Output of the program:

```
False
['6 8 0', 'd 1 6 8 0', '14 10 13 0', '11 12 0', 'd 1 11 12 0',
    '-8 14 -13 0', '14 -13 0', 'd -8 14 -13 0', '14 10 0', 'd 13
    14 10 0', '-9 10 7 0', '-9 10 0', 'd 7 -9 10 0', '-5 0', '10
    9 0', 'd 4 10 9 0', '-13 -15 0', 'd -12 -13 -15 0', '-14 -10
    0', 'd -11 -14 -10 0', '14 13 0', 'd 3 14 13 0', '13 7 0', 'd
     10 13 7 0', '7 0', '-8 0', '6 0', '-9 0', '3 0', '10 0', '-4
     0', '-14 0', '0']
```

□

B Enforcer $\mathcal{M}^{(i)}(u_1, \overline{u_2}, \overline{u_3})$

The set of clauses $\mathcal{M}$ constructed in Section 3 is the basis for the enforcer $\mathcal{M}^{(i)}(u_1, \overline{u_2}, \overline{u_3})$ and thus, for several results presented in this article. To facilitate verification of our results, we provide the set $\mathcal{M}$ as a Python list:

[[1, 2], [-2, -3], [-2, -4], [-3, -5, -6], [-4, -5, -6], [5, 7, 8], [6, 7, 8], [-7, -18, -19], [-7, -20, -21], [-8, -18, -19], [-8, -20, -21], [3, 9, 10], [3, 11, 12], [4, 13, 14], [4, 15, 16], [9, 12, 15], [10, 13, 17], [11, 16, 17], [-9, -13, -16], [-9, -14, -17], [-10, -11, -14], [-10, -12, -16], [-11, -13, -15], [-12, -15, -17], [2, 24, 32], [18, 23, 25], [18, 28, 29], [19, 23, 25], [19, 28, 29], [20, 22, 26], [20, 30, 31], [21, 22, 31], [21, 26,

27], [24, 27, 30], [-1, -22, -23], [-1, -24, -25], [-22, -25, -32], [-23, -24, -26], [-26, -28, -30], [-27, -28, -31], [-27, -29, -31], [-29, -30, -32]]

C Unsatisfiable instance of Monotone 3-Sat-$(2, 2)$

The unsatisfiable instance constructed in Section 3 as a Python list:

[[-193, -196, -198], [194, 196, 197], [1, 2, 197], [-2, -3, -194], [-2, -4, -194], [-3, -5, -6], [-4, -5, -6], [5, 7, 8], [6, 7, 8], [-7, -18, -19], [-7, -20, -21], [-8, -18, -19], [-8, -20, -21], [3, 9, 10], [3, 11, 12], [4, 13, 14], [4, 15, 16], [9, 12, 15], [10, 13, 17], [11, 16, 17], [-9, -13, -16], [-9, -14, -17], [-10, -11, -14], [-10, -12, -16], [-11, -13, -15], [-12, -15, -17], [2, 24, 32], [18, 23, 25], [18, 28, 29], [19, 23, 25], [19, 28, 29], [20, 22, 26], [20, 30, 31], [21, 22, 31], [21, 26, 27], [24, 27, 30], [-1, -22, -23], [-1, -24, -25], [-22, -25, -32], [-23, -24, -26], [-26, -28, -30], [-27, -28, -31], [-27, -29, -31], [-29, -30, -32], [33, 34, 196], [-34, -35, -195], [-34, -36, -198], [-35, -37, -38], [-36, -37, -38], [37, 39, 40], [38, 39, 40], [-39, -50, -51], [-39, -52, -53], [-40, -50, -51], [-40, -52, -53], [35, 41, 42], [35, 43, 44], [36, 45, 46], [36, 47, 48], [41, 44, 47], [42, 45, 49], [43, 48, 49], [-41, -45, -48], [-41, -46, -49], [-42, -43, -46], [-42, -44, -48], [-43, -45, -47], [-44, -47, -49], [34, 56, 64], [50, 55, 57], [50, 60, 61], [51, 55, 57], [51, 60, 61], [52, 54, 58], [52, 62, 63], [53, 54, 63], [53, 58, 59], [56, 59, 62], [-33, -54, -55], [-33, -56, -57], [-54, -57, -64], [-55, -56, -58], [-58, -60, -62], [-59, -60, -63], [-59, -61, -63], [-61, -62, -64], [65, 66, 193], [-66, -67, -195], [-66, -68, -197], [-67, -69, -70], [-68, -69, -70], [69, 71, 72], [70, 71, 72], [-71, -82, -83], [-71, -84, -85], [-72, -82, -83], [-72, -84, -85], [67, 73, 74], [67, 75, 76], [68, 77, 78], [68, 79, 80], [73, 76, 79], [74, 77, 81], [75, 80, 81], [-73, -77, -80], [-73, -78, -81], [-74, -75, -78], [-74, -76, -80], [-75, -77, -79], [-76, -79, -81], [66, 88, 96], [82, 87, 89], [82, 92, 93], [83, 87, 89], [83, 92, 93], [84, 86, 90], [84, 94, 95], [85, 86, 95], [85, 90, 91], [88, 91, 94], [-65, -86, -87], [-65, -88, -89], [-86, -89, -96], [-87, -88, -90], [-90, -92, -94], [-91, -92, -95], [-91, -93, -95], [-93, -94, -96], [-97, -98, -197], [98, 99, 195], [98, 100, 195], [99, 101, 102], [100, 101, 102], [-101, -103, -104], [-102, -103, -104], [103, 114, 115], [103, 116, 117], [104, 114, 115], [104, 116, 117], [-99, -105, -106], [-99, -107, -108], [-100, -109, -110], [-100, -111, -112], [-105, -108, -111], [-106, -109, -113], [-107, -112, -113], [105, 109, 112], [105, 110, 113], [106, 107, 110], [106, 108, 112], [107, 109, 111], [108, 111, 113], [-98, -120, -128], [-114, -119, -121], [-114, -124, -125], [-115, -119, -121], [-115, -124, -125], [-116, -118, -122], [-116, -126, -127], [-117, -118, -127], [-117, -122, -123], [-120, -123, -126], [97, 118, 119], [97, 120, 121], [118, 121, 128], [119, 120, 122], [122, 124, 126], [123, 124, 127], [123, 125, 127], [125, 126, 128], [-129, -130, -196], [130, 131, 193], [130, 132, 194], [131, 133, 134], [132, 133, 134], [-133, -135, -136], [-134, -135, -136], [135, 146, 147], [135, 148, 149], [136, 146, 147], [136, 148, 149], [-131, -137, -138], [-131, -139, -140], [-132, -141, -142], [-132, -143, -144], [-137, -140, -143], [-138, -141, -145], [-139, -144, -145], [137, 141, 144], [137, 142, 145], [138, 139, 142], [138, 140, 144], [139, 141, 143], [140, 143, 145], [-130, -152, -160], [-146, -151, -153], [-146, -156, -157], [-147, -151, -153], [-147, -156, -157], [-148, -150, -154], [-148, -158, -159], [-149, -150, -159], [-149, -154, -155], [-152, -155, -158], [129, 150, 151], [129, 152, 153], [150, 153, 160], [151, 152, 154], [154, 156, 158], [155, 156, 159], [155, 157, 159], [157, 158, 160], [-161, -162, -193], [162, 163, 198], [162, 164, 198], [163, 165, 166], [164, 165, 166], [-165, -167, -168], [-166, -167, -168], [167, 178, 179], [167, 180, 181], [168, 178, 179], [168, 180, 181], [-163, -169, -170], [-163, -171, -172], [-164, -173, -174], [-164, -175, -176], [-169, -172, -175], [-170, -173, -177], [-171, -176, -177], [169, 173, 176], [169, 174, 177], [170, 171,

174], [170, 172, 176], [171, 173, 175], [172, 175, 177], [-162, -184, -192], [-178, -183, -185], [-178, -188, -189], [-179, -183, -185], [-179, -188, -189], [-180, -182, -186], [-180, -190, -191], [-181, -182, -191], [-181, -186, -187], [-184, -187, -190], [161, 182, 183], [161, 184, 185], [182, 185, 192], [183, 184, 186], [186, 188, 190], [187, 188, 191], [187, 189, 191], [189, 190, 192], [1, 33, 65], [-97, -129, -161], [5, 37, 69], [-101, -133, -165], [6, 38, 70], [-102, -134, -166], [14, 46, 78], [-110, -142, -174], [32, 64, 96], [-128, -160, -192]]