Structural and Relational Data Mining for Systems Biology Applications

Dissertation
der Fakultät für Informations- und Kognitionswissenschaften
der Eberhard-Karls-Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Dipl.-Bioinf. Elisabeth Georgii
aus Bielefeld

Tübingen
2010

Tag der mündlichen Qualifikation:	15.12.2010
Dekan:	Prof. Dr.-Ing. Oliver Kohlbacher
1. Berichterstatter:	Prof. Dr. Daniel Huson
2. Berichterstatter:	PD Dr. Peer Kröger

A self does not amount to much, but no self is an island; each exists in a fabric of relations [...].

Jean-François Lyotard

Abstract

Due to the enormous accumulation of experimental data and the increasing need for combining heterogeneous data sources, the field of systems biology yields novel and very interesting problems in data analysis.

The development of high-throughput technologies has opened the possibility to study the behavior of many cellular components simultaneously. Therefore, there is an increasing interest and effort in not only understanding the functions of single isolated components, but also revealing the interactions and functional relationships between different components. Often, the outcome of large-scale measurements is conveniently represented in a structured form; prominent examples are protein-protein interaction networks, coexpression networks for genes, and bipartite graphs of associations between experimental conditions and regulated genes. This thesis presents different methods that aim at finding interesting patterns in such data. The main contributions are as follows.

First, an exact enumerative approach to dense cluster detection is proposed. Given a weighted interaction network and a default weight for missing edges, the density of a node set is defined as the average pairwise interaction weight. The described method finds all patterns that satisfy a user-defined minimum density threshold. Conceptually, this task is a generalization of clique search; however, the standard techniques to solve that problem are not appropriate for the generalized question. Fortunately, an efficient enumeration strategy can be achieved by adopting the reverse search paradigm. Remarkably, the same algorithmic framework is applicable to discover cluster patterns in other types of structured data, like asymmetric binary relations and multipartite graphs, as well as hypergraphs, n-ary relations, and tensors.

Second, our approach integrates additional constraints in order to focus the search on clusters that are relevant for the specific application at hand. For example, if each node in a network has an annotation profile attached to it, we can identify dense clusters where the nodes share a common subprofile. The principal idea is that the user provides the datasets of interest and defines desired properties of patterns with respect to them, and the method yields all solutions that match these criteria. This allows to jointly explore network data and background information in a systematic way.

Third, we devise dense cluster detection approaches that sacrifice completeness of the solution set in favor of efficiency. Here, two different directions are pursued. On the one hand, we use the search strategy of the enumeration methods and introduce heuristic pruning rules to speed up the procedure. On the other hand, we propose generalizations of agglomerative hierarchical clustering for bipartite data. They detect dense clusters by successive "greedy" merging of instance sets. Consequently, this strategy and the complete enumeration approach can be seen as opposite extremes of dense cluster detection algorithms for structured data. However, both methods are very transparent with respect to the properties of the discovered set of patterns and thereby facilitate the interpretation of results.

The presented algorithmic approaches are illustrated with a number of real-world applications in systems biology. They involve multiple types of genomic datasets and relate to different representative organisms, primarily yeast, human, and the plant *A. thaliana*. One scenario is protein complex prediction from experimental interaction data, with optional constraints from background data; the latter allow to discover context-dependent variants of complexes. Another application is the joint analysis of multiple biological networks that describe different kinds of relationships between genes, in our case transcriptional coregulation under different cellular conditions. Beyond that, we consider the detection of bicluster patterns from gene expression measurements. Finally, we show a small-scale case study on discovering associations between genomic sequence variation and transcription of genes.

Zusammenfassung

Aufgrund der enormen Fülle an experimentellen Daten und des steigenden Bedarfs an Methoden, die heterogene Datenquellen integrieren, stellt die Systembiologie-Forschung das Gebiet der Datenanalyse vor neue und sehr interessante Aufgaben.

Die Entwicklung von Hochdurchsatz-Messmethoden hat die Möglichkeit eröffnet, das Verhalten zahlreicher zellulärer Komponenten gleichzeitig zu analysieren. Neben der Aufklärung der Funktion einzelner isolierter Komponenten gilt daher auch den Interaktionen und funktionellen Beziehungen zwischen verschiedenen Komponenten wachsendes Interesse. Zur zusammenfassenden Darstellung von experimentellen Daten großen Maßstabs sind Graphstrukturen oftmals sehr geeignet, beispielsweise Interaktionsnetzwerke für Proteine, Coexpressionsnetzwerke für Gene und bipartite Graphen von Assoziationen zwischen experimentellen Bedingungen und regulierten Genen. Diese Arbeit stellt verschiedene Methoden vor, die darauf abzielen, interessante Muster in solchen Daten zu finden. Die wesentlichen Beiträge sind im Folgenden zusammengefasst.

Zunächst wird ein exakter enumerativer Ansatz zum Auffinden von dichten Clustern vorgeschlagen. Für ein gegebenes gewichtetes Interaktionsnetzwerk und ein Standardgewicht für fehlende Kanten definieren wir die Dichte einer Knotenmenge als das durchschnittliche paarweise Interaktionsgewicht. Die beschriebene Methode zählt alle Muster auf, die einen benutzerdefinierten Dichte-Schwellwert überschreiten. Konzeptionell kann man dieses Problem als eine Verallgemeinerung der Cliquen-Suche betrachten; entsprechende Standardtechniken eignen sich allerdings nicht zur Lösung der allgemeineren Fragestellung. Jedoch kann man durch Anwendung des Paradigmas der reversen Suche eine effiziente Enumerationsstrategie erhalten. Bemerkenswerterweise lässt sich dasselbe algorithmische Framework auch zur Clustersuche in anderen Formen strukturierter Daten anwenden; Beispiele dafür sind asymmetrische binäre Relationen und multipartite Graphen sowie Hypergraphen, n-äre Relationen und Tensoren.

Zum zweiten integriert unser Ansatz zusätzliche Constraints, um die Suche auf Clusterstrukturen zu beschränken, die für die spezifische vorliegende Anwendung

relevant sind. Falls zum Beispiel jeder Knoten eines Netzwerks mit einem Annotationsprofil versehen ist, können wir dichte Cluster identifizieren, deren Knoten hinsichtlich eines Teilprofils übereinstimmen. Die grundsätzliche Idee dabei ist, dass der Benutzer die Datensätze von Interesse vorgibt und gewünschte Mustereigenschaften in Bezug auf die einzelnen Datensätze festlegt; die Methode liefert dann alle Lösungen, die diese Kriterien erfüllen. Dieser Ansatz ermöglicht es dem Benutzer, Netzwerkdaten und Hintergrundinformation gemeinsam und auf systematische Art und Weise zu untersuchen.

Drittens werden Methoden zum Auffinden von dichten Clustern ausgearbeitet, die zugunsten der Effizienz auf die Vollständigkeit der Lösungsmenge verzichten. Hierbei werden zwei unterschiedliche Richtungen verfolgt. Einerseits verwenden wir die enumerative Strategie und führen zusätzlich heuristische Pruningregeln ein, um die Suche zu beschleunigen. Andererseits schlagen wir Verallgemeinerungen des agglomerativen hierarchischen Clusterings in bipartiten Daten vor, welche durch fortgesetztes “greedy” Verschmelzen von Instanzmengen dichte Muster entdecken. Letzterer Ansatz und die vollständige Musteraufzählung stellen gewissermaßen entgegengesetzte Extreme für Algorithmen zur Clustersuche dar. Jedoch sind beide Methoden äußerst transparent im Hinblick auf die Eigenschaften der zurückgegebenen Mustermenge und erleichtern somit die Interpretation der Ergebnisse.

Die vorgestellten algorithmischen Ansätze werden anhand einer Reihe von praktischen Anwendungen aus dem Bereich der Systembiologie illustriert. Diese beinhalten unterschiedliche Arten von genomischen Datensätzen und beziehen sich auf verschiedene repräsentative Organismen, in erster Linie auf Hefe, Mensch und die Pflanze *A. thaliana*. Ein Szenario ist die Vorhersage von Proteinkomplexen auf der Basis von experimentellen Interaktionsdaten, mit optionalen Constraints bezüglich verschiedener Arten von Hintergrundinformation; letztere ermöglichen die Entdeckung kontextabhängiger Komplexvarianten. Eine weitere Anwendung ist die gemeinsame Analyse mehrerer biologischer Netzwerke, welche unterschiedliche Arten von Beziehungen zwischen Genen beschreiben, in unserem Fall transkriptionelle Coregulation unter verschiedenen zellulären Bedingungen. Darüber hinaus betrachten wir die Suche nach Bicluster-Mustern in Genexpressionsdaten. Schließlich zeigen wir eine kleine Fallstudie, die Assoziationen zwischen der Variation genomischer Sequenzen und der Transkription von Genen untersucht.

Acknowledgements

I would like to express my deep gratitude to everyone who supported my scientific and personal development during the past years, even if I can only list a few names here.

First of all, I would like to give my sincere thanks to my first advisor Dr. Koji Tsuda, who guided this work with his experience and open-mindedness; he always encouraged me and broadened my horizon in our numerous discussions. I am also very grateful to my second advisor Dr. Gunnar Rätsch, who gave invaluable advice and supported my work and scientific education in many ways.

Furthermore, I would like to thank Prof. Dr. Bernhard Schölkopf for providing an inspiring working environment and for taking an interest in my research. I have also benefited a lot from discussions with Dr. Karsten Borgwardt, for which I would like to thank him.

I am very grateful to Prof. Dr. Daniel Huson for his interest in my work; he willingly took part in my PhD committee and acts as the first referee. I would also like to thank PD Dr. Peer Kröger, who kindly agreed to examine the thesis as the second referee.

With gratefulness, I would also like to mention the many other people who had an impact on my scientific development, to name in particular Prof. Dr. Thomas Hofmann and Prof. Dr. Stefan Kramer as well as their research groups. I would like to thank Prof. Dr. Samuel Kaski and Dr. Jarkko Salojärvi for their patience, encouragement, and support while I was finalizing work related to this thesis.

The main part of this work was performed under a PhD grant by the Max Planck Institute for Biological Cybernetics and the Friedrich Miescher Laboratory of the Max Planck Society; furthermore, it was supported by funding from TEKES (Project 40141/07) and the IST Programme of the European Community, under the PASCAL2 Network of Excellence, IST-2007-216886.

Many aspects of this work have been initiated and inspired by insightful discussions with our collaborators. I am particularly grateful to Dr. Sabine Dietmann, Prof. Dr. Takeaki Uno, Prof. Dr. Bertram Müller-Myhsok, Michael Specht, Dr. Philipp Pagel, Prof. Dr. Sridhar Hannenhalli, and Dr. Mitch Levesque.

Finally, I cordially thank my colleagues, my friends, and my family for helping me, supporting me, and making my life a happy one.

Contents

I INTRODUCTION 1

1 Motivation for Structured Data Mining 3

1.1 Data Mining 3

1.2 Systems Biology 4

1.3 Challenges 5

1.4 Outline of the Thesis 6

2 Common Systems Biology Resources 9

2.1 Protein Interaction Data 9

2.2 Gene Expression Data 11

2.3 Gene Ontology 12

3 Structured Data Representations and Formalisms 15

3.1 Graphs and Networks 15

3.2 Relations 17

4 Review on Unsupervised Analysis of Structured Data 19

4.1 Graph Mining 19

4.2 Optimal Subgraph Search 20

4.3 Graph Clustering 21

4.4 Constrained Cluster Detection and Data Integration 22

4.5 Bicluster Analysis 22

4.6 Itemset Mining 24

4.7 Relational Data Mining and Higher-Order Analysis 25

II SET ENUMERATION BASED ON INTERACTION DENSITY – A REVERSE SEARCH APPROACH 27

5 Module Mining in Weighted Interaction Networks 29
5.1 Motivation . . . 29
5.2 Definitions . . . 32
5.3 Enumeration Algorithm . . . 33
  5.3.1 Search Space . . . 33
  5.3.2 Reduction Scheme . . . 36
  5.3.3 Search Procedure . . . 38
  5.3.4 Implementation Details . . . 39
  5.3.5 Complexity . . . 41
  5.3.6 Excursus: Reverse Search Applications . . . 43
5.4 Output Representation . . . 43
  5.4.1 Locally Maximal Modules and Leaf Modules . . . 45
  5.4.2 Module Ranking . . . 46
5.5 Degree-Based Module Criteria . . . 48
  5.5.1 Minimum Degree Criterion . . . 48
  5.5.2 Minimum Relative Degree and Quasi-Cliques . . . 49
  5.5.3 Previous Work on Quasi-Clique Mining . . . 51
  5.5.4 Discussion . . . 52
5.6 Integration of Node Weights . . . 53
  5.6.1 Definitions . . . 53
  5.6.2 Algorithm . . . 54
  5.6.3 Remarks . . . 55
5.7 Constraint Integration . . . 56
  5.7.1 Constraints from External Data Sources . . . 56
  5.7.2 Connectivity Constraints . . . 59
  5.7.3 Cardinality and Branching Restrictions . . . 60

6 Multi-Way Cluster Mining in Higher-Order Association Data 63
6.1 Motivation . . . 63
6.2 Problem Definition . . . 66
6.3 Enumeration Approach . . . 68
  6.3.1 Global Index Representation . . . 68
  6.3.2 Search Space . . . 69
  6.3.3 Reduction Scheme . . . 69
  6.3.4 Search Algorithm . . . 72

6.3.5 Implementation Details 73
6.3.6 Complexity 75
6.4 Extensions 76
6.4.1 Output Filtering and Balance Criteria 76
6.4.2 Cluster Ranking 77
6.4.3 Isolation-Based Pruning 77
6.4.4 Other Restrictions 78
6.5 Symmetry Adaptations 79
6.5.1 Motivation 79
6.5.2 Definitions 80
6.5.3 Reduction Scheme 81
6.5.4 Details 82
6.6 Experimental Studies 83
6.6.1 Scalability 84
6.6.2 Performance of Branching-Restricted Search 86
6.6.3 Efficiency of Reverse Search 89
6.6.4 Email Traffic Analysis 92
6.7 Discussion 93

III HIERARCHICAL DETECTION OF ASSOCIATION PATTERNS 95

7 Hierarchical Biclustering 97
7.1 Motivation 97
7.2 Review of Hierarchical Clustering 98
7.3 Agglomerative Biclustering Algorithm 101
7.3.1 General Scheme 101
7.3.2 Correctness 103
7.3.3 Complexity 104

8 Extensions of Hierarchical Biclustering 107
8.1 Alternative Association Criteria 107
8.2 Alternative Clusterings 108
8.3 Hierarchical Higher-Order Clustering 109

IV BIOLOGICAL APPLICATIONS 111

9 Module Discovery in Protein Interaction Networks 113
9.1 Data Collection and Preprocessing 114
9.2 Comparative Analysis on the Yeast Interaction Network 115
9.2.1 Precision-Recall Analysis 116
9.2.2 Further Result Statistics 117
9.3 Comparative Analysis on the Human Interaction Network 120
9.4 Phenotype-Associated Modules in Yeast 123
9.5 Evolutionary Conserved Modules in Yeast 124
9.6 Tissue-Specific Modules in Human 126
9.7 Disease-Related Module Analysis 128
9.8 Discussion and Outlook 129

10 Module Detection from Multiple Coexpression Networks 131
10.1 Data 131
10.2 Related Approaches 132
10.3 Experimental Set-Up 132
10.4 Evaluation Measures 133
10.5 Results 135

11 Biclustering of Gene Expression Data 137
11.1 Data 137
11.2 Sample-Based Evaluation 139
11.3 Gene Function Analysis 141

12 SNP-Transcript Association Discovery 145
12.1 Motivation 145
12.2 Data and Preprocessing 147
12.3 Experimental Approach 147
12.4 Results 148

V CONCLUSION 153

13 Summary 155

14 Discussion and Future Work 157

Bibliography **161**

List of Figures **177**

List of Tables **179**

List of Algorithms **181**

Publications **185**

Curriculum Vitae (English) **187**

Lebenslauf (German) **189**

Part I

Introduction

1 Motivation for Structured Data Mining

Due to the enormous advances in high-throughput measurement techniques and the accumulation of functional annotation for genes and their products, computer-assisted data analysis has become crucial in molecular systems biology studies. In this chapter, we introduce basic concepts of data mining in systems biology and outline the contents of this thesis.

1.1 Data Mining

Data mining is a discipline that comprises various kinds of methods for the automated extraction of patterns from a data collection [54, 78, 79, 84]. By that, it contributes to gaining novel insights about concepts and principles underlying the observations, a process that is commonly referred to as knowledge discovery [58]. Traditionally, one distinguishes two main subfields of data mining:

- *Descriptive methods* aim at revealing inherent properties of the data by searching for underlying distributions, similarity relationships between data objects, dependencies between variables, or characteristic patterns. Many approaches build global models of the whole data space, whereas others discover local patterns that describe only parts of the data. The most prominent descriptive data mining tasks are cluster analysis (i.e., identification of groups in the data) and frequent pattern mining.

- *Predictive methods* focus on the task of predicting the values of specific target variables, which are typically discrete labels or continuous values. In the former case, we obtain a classification problem, in the latter case a regression problem. The target values are available for a subset of data instances, and the goal is to exploit this information to make predictions for other data instances. As the methods know the desired outcome for the given training examples, this field is also called *supervised* learning.

Beyond that, many additional aspects play a role in data mining. For instance, visualization of high-dimensional datasets is an important tool for exploratory data analysis. Furthermore, retrieval settings become increasingly popular; there, the basic task is to find objects in a database that are similar to a given query object (e.g., a text document or an image). Other interesting data mining applications are outlier detection and the analysis of dynamic (i.e., time-dependent) behavior. Today, activities regarding data acquisition, storage, and analysis are crucial in almost every field and influence scientific, economic, and sociological decision processes. Therefore, data mining technology has to face widening and constantly changing demands and has to deal with more and more complex data types as well as with ever-growing databases. One area where this development can be clearly observed is molecular systems biology.

1.2 Systems Biology

We first give a very brief introduction to molecular biology. For more information, we refer to standard textbooks on that topic (e.g., [7, 138, 147]). Living organisms consist of cells. The genetic information of a cell is stored on large DNA molecules, which are called chromosomes. Before a cell divides, the DNA is replicated so that each daughter cell obtains a copy of the full genetic information. Genes are DNA segments that encode basic functional units of the cell; they essentially can be represented as a sequence of four different types of nucleotides, the building blocks of a DNA molecule. According to the central dogma of molecular biology (Francis Crick, 1958), the information flow from DNA to functional molecules consists of the following steps:

- DNA is transcribed into RNA, which also consists of four nucleotides. In eucaryotic cells, the primary RNA transcript is further processed and the resulting messenger RNA (mRNA) is transported from the nucleus to the cytoplasm; by a process called alternative splicing, one gene transcript may be transformed into different types of mRNAs; in prokaryotic cells, which do not have a nucleus, the primary RNA transcript directly acts as mRNA.

- The mRNA is translated into a protein. The amino acid sequence of the protein is uniquely determined by the mRNA sequence. One mRNA molecule can be used multiple times as a template for protein synthesis. Proteins are the main macromolecules to carry out cellular functions; they act for instance as enzymes for metabolic reactions, as transporter or signal transduction molecule, or as regulator of the gene transcription process.

There are many possibilities to control the availability of proteins in the cell. First of all, each substep in the described process of information transfer is regulated separately, starting from the transcription initiation – via mRNA synthesis, processing, export, localization, and degradation – to the initiation of translation and protein degradation. Beyond that, the activity of proteins can be tuned by their subcellular localization and by post-translational modifications.

Yet, a living cell is much more than the sum of its proteins. Proteins (and other functional molecules) are involved in a complex network of inter-relationships, and most cellular processes depend on functional modules rather than isolated components [83]. The field of systems biology aims at understanding how properties of living systems emerge from the functional interplay of molecules [6, 146]. This may refer to different levels of biological organization, ranging from the analysis of protein complexes in a single cell to the characterization of cell ensembles and tissues, or even models regarding the behavior of a whole organism, e.g., with respect to certain diseases and therapies.

1.3 Challenges

Systems biology is a highly interdisciplinary field. The biological problems under study can also involve aspects from chemistry, physics, mathematical modeling, and data analysis. Here, we point out some of the major data mining challenges that arise in systems biology [87]:

- Large-scale and high-throughput experiments produce enormous amounts of data; public data repositories grow rapidly.
- Measurements are noisy, and the number of samples is much smaller than the number of genes or proteins, rendering many tools of classical statistics inapplicable.
- Integrative analysis of heterogeneous datasets is very often required. Potentially relevant types of information are for instance genomic sequences and their variation (e.g., single nucleotide polymorphisms (SNPs) or copy number variations), transcriptomic and proteomic measurements, functional annotation,

metabolic pathway information, and interactomics data, also across multiple species.

- Structured and relational data representations become more and more prevalent, therefore methods dealing with data embeddings in Euclidean space are not sufficient anymore.
- Interpretability of data mining results is crucial for the generation of biological hypotheses, which then can be tested experimentally.

This work focuses on unsupervised methods for structured data analysis and data integration, also touching aspects of interpretability and scalability.

1.4 Outline of the Thesis

Graph-structured data are ubiquitous. For instance, graphs offer a convenient way to represent pairwise similarity relationships between objects. Beyond that, graph representations can formalize a multitude of associative relationships, such as physical interaction, spatial proximity, coocurrence, communication, and regulation. One of the central topics in unsupervised data mining with graphs is the detection of groups of related objects, also called clusters. This thesis presents an enumerative mining approach to find clusters of densely interacting nodes in large graphs. The search is based on an explicitly defined density threshold for solution patterns and allows to integrate additional constraints, also properties with respect to external data sources. Beside ordinary graphs, we look at asymmetric and multi-partite structures as well as higher-order associations involving more than two objects at the same time. Furthermore, to deal with large bipartite datasets, the exact enumerative pattern mining approach is complemented with a hierarchical cluster detection approach. Although the proposed methods are suitable for general use, we mainly show applications in systems biology. The remaining content of this thesis is organized as follows.

- *Chapter 2* introduces the major systems biology data types that will be used in our experiments, in particular protein interactions, gene expression measurements, and Gene Ontology annotation.
- *Chapter 3* describes structured data representations, ranging from graphs and networks to multi-relational and higher-order settings.

- *Chapter 4* gives a survey of related work on analyzing structured and relational data. This includes graph mining, optimal subgraph search, graph clustering, integrative and constrained clustering, biclustering, itemset mining as well as relational data mining and higher-order data analysis. Biological applications are mentioned where appropriate.

Part II deals with the enumerative density-based cluster detection approach for structured data. It is based on a general algorithmic framework called reverse search.

- *Chapter 5* considers the basic case where the data are represented as a weighted interaction network. After explaining the enumeration algorithm, various extensions are discussed.
- *Chapter 6* addresses the generalization of the method to multi-way association data. Again, several variants of the basic scheme are proposed, and the behavior of the algorithm is demonstrated in empirical studies.

Part III presents agglomerative hierarchical clustering strategies for structured data.

- *Chapter 7* first reviews the classical agglomerative hierarchical clustering, which can also be used for cluster detection in networks. Then it describes a generalized approach performing direct hierarchical biclustering of bipartite data.
- *Chapter 8* discusses several extensions including higher-order analysis.

Part IV shows results from systems biology applications using the described methods. All of the studies aim at uncovering functional relationships between different molecular components of a biological cell, based on experimental data and computational analysis.

- *Chapter 9* treats the problem of protein complex prediction from protein interaction networks.
- *Chapter 10* considers the joint analysis of multiple coexpression networks.
- *Chapter 11* handles an example of bicluster analysis in gene expression data.
- *Chapter 12* illustrates the use of bicluster detection in the context of SNP association studies.

Part V concludes the thesis.

- *Chapter 13* summarizes the different parts.
- *Chapter 14* discusses merits and limitations of the proposed approaches and gives some hints for future work.

2 Common Systems Biology Resources

In this chapter, we give some background information on the major systems biology resources used in this work, including protein interaction and gene expression data as well as Gene Ontology annotation. Additional data types will be introduced in the experimental sections (Part IV) where appropriate.

2.1 Protein Interaction Data

Proteins specifically interact with each other in cellular processes. The linear amino acid chain of a protein folds into a particular three-dimensional structure consisting of one or several domains, and several proteins with identical or different amino acid sequences can physically aggregate to build so-called complexes. In the last decade, a number of experimental techniques to identify such protein interactions have been developed [194], some of which are suitable for high-throughput application.

The *yeast two-hybrid method (Y2H)* is very popular for large-scale in-vivo interaction measurements. The principal idea of the approach is to split a transcription activator (most often Gal4) into its two domains, the DNA-binding domain (BD) and the activation domain (AD). In order to activate transcription, both domains must be physically associated; either of them alone is not sufficient for activation. The BD and AD sequences are fused with a protein-coding gene, respectively, and then inserted into separate plasmids, which are transfected into yeast cells. If the expressed proteins interact with each other, they bring the BD and AD domains in close proximity and thereby activate the transcription of a specific reporter gene; many experimental systems use for that a gene encoding an enzyme that turns a certain substrate into a dye. To analyze the entire interactome of an organism, one can either systematically test all pairs of proteins (by mating two yeast strains containing the corresponding plasmids) or screen particular strains agains undefined libraries, which are sequenced in case of success. Problems of the Y2H method are false positives arising from unspecific interactions as well as potential disruption or changes of protein folding by the fusion constructs. Also, the processes of protein folding and post-translational modification in yeast can differ from other organisms.

Another well-established method is *TAP-MS*, *tandem affinity purification* with subsequent *mass spectrometry analysis*. It investigates protein interactions in vitro and is based on a sensitive multi-step purification process that preserves protein complexes, which are characterized using polyacrylamide gel electrophoresis and mass spectrometry. First, a gene of interest is fused with the so-called TAP-tag and expressed, e.g., in yeast; then, cellular extracts are analyzed for complexes containing the protein of interest; the purification is greatly facilitated by the specific structure of the expressed TAP-tag, which is composed as follows:

(Target protein) – CBP – TEV site – Protein A

Protein A binds tightly to an IgG matrix, which is used in the first purification step. After washing away unbound proteins, the TEV (tobacco etch virus) protease cleaves the tag at the TEV site; the eluted material is exposed to calmodulin-coated beads, which bind CBP (calmodulin binding peptide), together with the target protein and the potentially associated protein complex. To identify the complex components, the purified material is separated by SDS polyacrylamide gel electrophoresis and further analyzed by mass spectrometry. In contrast to Y2H, TAP-MS has the advantage that it looks directly at protein complexes, i.e., higher-order interactions. However, being an in-vitro technique, it might miss transient interactions. As in the Y2H method, the tag construct might interfere with protein folding or complex formation, thereby producing false negatives as well as false positives. The quality of protein interaction predictions from high-throughput experiments is considerably improved by taking evidence from multiple data sources into account.

Other approaches that are suitable for high-throughput usage are protein microarrays and phage display. Beyond that, indirect methods can be used to screen for potential protein interactions, e.g., gene expression measurements, which are described in the next section. For a more detailed characterization of specific protein interactions, various experimental methods are available, for instance chemical cross-linking, calorimetry, ultracentrifugation, fluorescence resonance energy transfer (FRET), surface plasmon resonance, and atomic force microscopy; to reach an atomic-level resolution of protein structures and protein complexes, X-ray crystallography or NMR spectroscopy is required, both of which are extremely laborious.

Experimentally determined pairwise interactions and complexes of proteins are stored in huge public databases. The most widely used data repositories are DIP [230], BIND [15], MPact/MIPS [76], MINT [35], IntAct [85], BioGRID [26], and HPRD [174]. Also, computational prediction of protein interactions becomes an important source of information; the STRING database [102] exploits for instance homology relationships to transfer protein interactions across species and currently covers more than two million proteins from 630 organisms. In addition, interactions can be predicted de novo based on genomic context, phylogenetic profiles, domain fusion, and sequence coevolution [195]. So far, protein interactions are most commonly represented as static networks. However, it is widely recognized that interaction and complex formation in the living cell are context-specific and highly dynamic [8, 118].

2.2 Gene Expression Data

A very popular tool to study the function of genes is gene expression analysis [70, 147, 202], often referred to as transcriptomics. The experimental part essentially consists in measurements of the mRNA abundance for all genes under varying conditions. This can be achieved by several different techniques, among which microarray technologies constitute the most common approach. They are based on the principle of hybridization, which works as follows. A set of gene probes is immobilized on a solid surface (the "chip"). Each probe is a single DNA strand that is complementary to the mRNA of a certain gene (or to some part of the mRNA); here, complementarity means that the two nucleic acid strands can pair with each other by hydrogen bonds between corresponding nucleotides; DNA that is generated as a complementary copy of mRNA is called cDNA. Then, an mRNA sample is taken from a cell population of interest and the mRNA molecules or corresponding cDNA molecules are labeled with a fluorescent dye. The labeled sample is put onto the array for hybridization, and fluorescence of bound material is detected with a laser. Each spot of the microarray contains many identical probes, so the intensity of the signal at the spot depends on the amount of a particular mRNA in the sample.

There exist two main types of microarrays: cDNA microarrays and oligonucleotide arrays. For cDNA microarrays, preamplified cDNAs are attached to the chip; the hybridization experiment is done with a pair of differently labeled samples, which competitively bind to the probes; this results in ratio data that describe the differential expression between the two samples. In contrast, oligonucleotide arrays (e.g., Affymetrix) can be used to measure intensity values for a single sample; the probes are short sequences of about 25 nucleotides length, which are synthesized directly on the slide.

Each microarray experiment yields measurements for thousands of genes simultaneously. A set of measurements taken under different conditions or cellular perturbations can be used to monitor changes in the transcriptional behavior of genes; the comparison of the resulting expression profiles can reveal relationships between conditions as well as relationships between genes. The outcome of a particular study is typically represented in the form of a data matrix, with genes as rows and conditions as columns; the matrix cells contain condition-specific expression levels of specific genes, which are often given relative to a reference sample. This is the starting point for data analysis methods, with the aim to reach conclusions for specific biological questions. Inferring gene relationships from the data is non-trivial and complicated by the fact that the number of genes typically exceeds by far the number of samples. In addition, special care has to be taken with respect to data normalization in order to achieve comparability between different experiments and genes [93, 228].

Beside using hybridization-based microarrays, gene expression can be quantified by sequencing approaches like SAGE (serial analysis of gene expression) and the recently introduced and rapidly developing RNA-Seq methods, which are based on next-generation sequencing technologies [223].

2.3 Gene Ontology

Functional genomics studies do not have to start from scratch. Over the past decades, a multitude of genes have been functionally characterized; with the advent of the World Wide Web, this information is made conveniently accessible to the whole research community. In an effort to systematize functional annotation of genes, also across multiple species, different categorization schemes have been developed. Nowadays, very popular resources are Gene Ontology (GO) [13] and Functional Catalogue (FunCat) [183], both of which provide hierarchical classification systems to describe the function of genes and proteins. Another very frequently used database is the Kyoto Encyclopedia of Genes and Genomes (KEGG) [164], which contains biochemical pathway descriptions.

These sources of information can either be used to evaluate methods for data-driven prediction of functional gene groups or they can be exploited to restrict the analysis and focus on new biological findings. In this work, we often use GO to assess the biological significance of predicted gene clusters; therefore, we briefly introduce its main concepts. The notion of function is quite vague and can have many different meanings, depending on the context. For this reason, GO offers three independent ontologies:

- *Biological process* describes the cellular function by a defined biological objective, for instance "translation".
- *Molecular function* refers to the biochemical activity of gene products, without considering in which biological context the corresponding reaction takes place. A prominent example are enzyme classes.
- *Cellular component* specifies in which compartment or location of an eukaryotic cell the active gene product can be found.

Each ontology consists of a hierarchy of defined terms; a single term may have multiple parent terms, so the hierarchical structure is technically not a tree, but a directed acyclic graph (see next chapter). Each term comprises a set of genes, which can be further divided into functional subcategories represented by own terms. The other way round, a specific gene is assigned to multiple terms from different hierarchy levels; terms at the bottom of the hierarchy are more specialized than terms at the top. The terms provide a unifying framework for functional classification of genes across different organisms.

One common usage of GO is functional enrichment analysis for computationally predicted gene sets. Given a predicted gene set S, the enrichment with respect to a GO term corresponding to a gene set T is typically computed using a p-value based on the hypergeometric distribution [179]:

$$1 - \sum_{i=0}^{|S \cap T|-1} \left[\binom{|T|}{i} \binom{n-|T|}{|S|-i} \bigg/ \binom{n}{|S|} \right] \tag{2.1}$$

Here, n is the total number of genes, and $|S \cap T|$ denotes the number of genes in the overlap. The expression corresponds to the probability of obtaining by chance an overlap of at least that size. Very low values indicate that S is significantly enriched with genes that share a certain function.

3 Structured Data Representations and Formalisms

Biological data are often represented in an abstract form as networks or relations. Here, we describe common structural representations and some basic properties.

3.1 Graphs and Networks

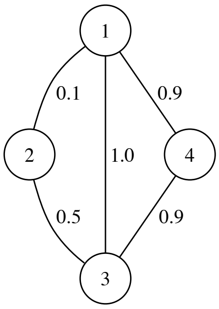
A graph or network consists of a set of *nodes* (*vertices*) with pairwise connections called *edges* [44, 161]. Figure 3.1 (a) shows an example graph with four nodes. If the edges are labeled with weights (as in the example), we say that the graph is *weighted*, and *unweighted* otherwise. Nodes that are connected by an edge are called *adjacent* ("neighboring"). An *induced subgraph* is defined as the restricted graph we obtain by considering a specific subset of nodes and the edges connecting them with each other. The size of a subgraph corresponds to the number of its nodes; to explicitly indicate the size of a subgraph, often the notation "k-node subgraph" is used. The number of edges connecting a node-induced subgraph with the remainder of the graph (or, in weighted graphs, the sum of the corresponding edge weights) are referred to as the *cut*. A graph can alternatively be represented in form of an adjacency matrix for node pairs that contains 0-entries for missing edges (see Figure 3.1 (b)). Here, the matrix is symmetric because the edges are *undirected*. In the case of *directed* edges, there can be different weights for forward and backward connections.

A *tree* is a special type of a directed graph where the nodes are organized in a hierarchy such that there is one node at the top (the *root node*) and each other node has a unique parent node and an arbitrary number of child nodes;[1] see Figure 3.2 for an example. Parent and child nodes are connected by edges that are oriented along the hierarchy, typically pointing from parents to children. Nodes without children

---

[1] This definition of tree refers to the typical usage in data mining; in graph theory, the concept is more general.

(a) Graph representation



(b) Matrix representation

	1	2	3	4
1	0	0.1	1.0	0.9
2	0.1	0	0.5	0
3	1.0	0.5	0	0.9
4	0.9	0	0.9	0

Figure 3.1: A weighted graph with four nodes.

are called *leaf nodes*. A *path* is a sequence of nodes such that subsequent nodes are connected by an edge; the length of a path is defined as the corresponding number of edges. Any non-root node in the tree is reachable by a specific path from the root. By definition, trees are *acyclic*, that means, there does not exist a path that starts and ends with the same node.

An edge that connects a node with itself is called *loop*. Throughout this thesis, we only consider simple graphs, which have the following properties: a) they do not contain loops; b) there cannot be more than one edge pointing from one particular node to another specific node. Both nodes and edges can carry labels, which are either discrete categories or numerical weights. In most parts of this work, we will consider undirected graphs with edge weights; furthermore, we will assume that each node is labeled with a unique identifier (such as a gene or protein name). If the node set can be split into disjoint subsets such that edges exist only between subsets and not within them, we say that the graph is *multipartite*. The case of two partitions occurs quite frequently; such graphs are called *bipartite*.

If there exists a path between each pair of nodes, the graph is said to be *connected*. Otherwise, it consists of several *connected components*, which can be determined by graph traversal algorithms starting from specific nodes. Two common traversal strategies are *depth-first search* and *breadth-first search*. In breadth-first search, first the neighbors of the current node are visited, and then each of them is further investigated (in the same way, but ignoring already visited nodes). In contrast, depth-first search recursively explores all descendants of the first neighbor before going to the next neighbor. Performing a depth-first search through the tree shown in Figure 3.2, starting from the root node, would yield the nodes in their numbered order.

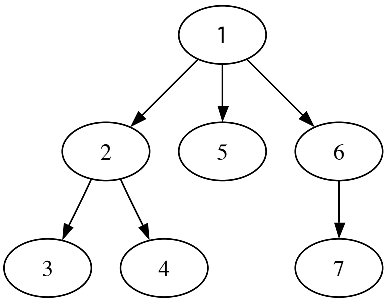


Figure 3.2: A tree structure.

(a) Graph relation

First entity	Second entity	Weight
1	2	0.1
2	1	0.1
1	3	1.0
3	1	1.0
1	4	0.9
4	1	0.9
2	3	0.5
3	2	0.5
3	4	0.9
4	3	0.9

(b) Node annotation relation

Entity	Description	Expression in colon	Cancer-related
1	Protein1	3.5	Yes
2	Protein2	1.1	No
3	Protein3	2.7	Yes
4	Protein4	3.8	Unknown

Figure 3.3: Example relations; (a) is an equivalent representation of the graph in Figure 3.1 (a).

3.2 Relations

Graphs can be viewed as special cases of relational data [51, 54]. Mathematically, a *relation* R is a subset of the cartesian product of a set of domains $D_1, \ldots, D_k$:

$$R \subset D_1 \times \ldots \times D_k \tag{3.1}$$

The domain tuple $(D_1, \ldots, D_k)$ is called relation schema. According to the above definition, a relation R consists of a set of k-tuple observations $(d_1, \ldots, d_k)$, $d_i \in D_i$ for $i = 1, \ldots, k$, which can be conveniently represented in a *table* with k columns (also called *attributes*). The minimal set of attributes that is needed to uniquely identify any possible tuple is called the *key* of the relation schema. Figure 3.3 (a) shows an example of a relation; the key of the corresponding schema consists of the first two attributes. In this case, both attribute domains correspond to the same entity set (i.e., a set of distinct objects or instances, here represented by numeric identifiers), and the third attribute domain is the set of real numbers from 0 to 1. The relation is equivalent to the graph in Figure 3.1 (a). However, the tabular representation immediately allows to describe more general types of data.

For instance, by adding more columns, one could consider higher-order relations, where each observation tuple associates not only two different entities (nodes), but n of them. Again, the tuples can contain additional information, such as a weight of the association or discrete labels, although most higher-order relational data mining approaches do not consider this case [31, 100, 103]. We refer to this kind of table as an *n-ary relation*, emphasizing the number of key attributes rather than the total number of columns in the table. From a graph-theoretic perspective, an n-ary relation corresponds to a *hypergraph* where each edge involves n different nodes. In analogy to the matrix representation of graphs shown in Figure 3.1, the adjacency structure of this hypergraph can be represented as an n-dimensional array, also known as *tensor*. In the absence of edge weights, the array is binary-valued, i.e., entries representing observed tuples of the relation are marked with 1's, and all other entries are 0. More details about n-ary relations will be given in Section 4.7 and Chapter 6.

Moreover, multiple relations can be regarded simultaneously in data analyses, provided that they share some set of attributes. This is illustrated with a table in Figure 3.3 (b), which yields auxiliary information for the entities in Figure 3.3 (a). Another very common scenario are star-structured arrangements of relation schemata [60, 148, 205], where one central entity type connects relations that otherwise have disjoint attribute sets; essentially, this yields a multipartite graph of relationships between entities of different types.

4 Review on Unsupervised Analysis of Structured Data

Unsupervised data mining in the context of structural and relational data is a very broad and active field of research. In this chapter, we review central approaches that are related to our work. This includes mining for interesting subgraph patterns (Section 4.1), search for optimal subgraphs (Section 4.2), and graph clustering (Section 4.3). Furthermore, we describe the tasks of integrative cluster detection (Section 4.4) and biclustering (Section 4.5), both of which are heavily driven by computational biology applications. Finally, we survey important topics in itemset mining (Section 4.6) and relational data mining (Section 4.7).

4.1 Graph Mining

Graph mining refers to the search for subgraph patterns with predefined characteristics, in a database of one or multiple graphs. In many cases, it is possible to design algorithms that yield the complete set of solutions; such approaches are called *enumerative*. In the following presentation, we focus on the most common tasks. We start with the classical problem of frequent subgraph mining [78, 133, 225, 233, 234].

Definition 1 (Frequent Subgraph Mining from Multiple Graphs)**.** *Let D be a database of labeled graphs $G_1, \ldots, G_l$. Find all connected subgraphs that occur in at least m graphs, where m is a positive integer referred to as the minimum support threshold.*

Regarding the generation of subgraph patterns, there exist two main strategies: depth-first search [233] and breadth-first or level-wise search [133]. As the same node label may appear multiple times in each graph of the database (e.g., atom names in a database of molecule graphs [129]), these methods generally have to deal with the problem of subgraph isomorphism. In biological networks considering gene or protein relationships, the node labels within a graph are typically unique.

However, as the graphs are large, the frequency criterion is typically combined with other criteria like interaction density or cut thresholds, in order to restrict the size of the output [91, 236].

Other mining approaches search for substructure patterns in a single graph. While a subgraph frequency criterion can be applied to graphs with non-unique node labeling [106, 134], a popular analysis tool for uniquely labeled graphs is clique finding [5]:

Definition 2 (Clique)**.** *Given a graph G with node set V, a clique is defined as a subset of nodes $U \subset V$ that induces a complete subgraph, i.e., all pairs of nodes are connected by an edge. A clique is maximal if it is not contained in any other clique.*

Only for specific classes of graphs (e.g., chordal graphs), all maximal cliques can be discovered in polynomial time; in general, clique search is NP-complete [25, 110, 168]. Nevertheless, it is frequently used in practical applications [166, 198]. Also, less strict pattern definitions have been considered, e.g., quasi-cliques [104, 145, 172, 242] and pseudo-cliques [214], which will be discussed in more detail in Chapter 5. They correspond to dense subgraphs rather than complete subgraphs.

4.2 Optimal Subgraph Search

In addition to subgraph enumeration algorithms, there exist multiple approaches to search for (approximately) optimal subgraphs. With respect to the criterion of subgraph density, a number of problems have been studied. First of all, it has been shown that finding a k-node subgraph with the maximum number of edges is NP-hard [11]. Tight approximation bounds have been derived for a simple greedy optimization scheme [12]; the same approximation scheme has been used for directed graphs [34]. Equivalently, the problem of finding a k-node subgraph with the maximum average number of edges per node is NP-hard [59, 114]. On the other hand, a subgraph with the maximum average number of edges per node (i.e., without a size constraint) can be found in polynomial time by flow-based techniques [59, 65].

Moreover, local search approaches have been used to discover dense subgraphs around seed cliques [16, 55]. Recently, linear integer programming has been successfully applied for finding connected subgraphs with the maximum sum of node weights, an NP-complete problem [49]. Yet another concept of a subgraph pattern is the so-called connection subgraph or reliable subgraph [56, 86, 126]. There, the aim is to maximize the connectivity between a set of given nodes while removing a large portion of the graph.

4.3 Graph Clustering

Graph clustering is the task of assigning the nodes of a graph into distinct groups ("clusters") such that there are many edges within a group and few edges between different groups. This topic has been studied extensively, see [186] for a review. One seminal work in this area is the Kernighan-Lin algorithm [113], which is a heuristic strategy to divide a graph into components with fixed maximum size. If neither the (maximum) size nor the number of partitions is known beforehand, a popular choice are hierarchical clustering methods, which yield a hierarchy of clusters instead of a single partitioning. Hierarchical methods can be divided into two classes: agglomerative and divisive. Agglomerative strategies build the hierarchy bottom-up, starting from single-node clusters and iteratively merging the "closest" pair into a common cluster, e.g., see [41, 90, 180, 231]. For that purpose, one has to define a distance measure between the nodes in a graph; a simple choice would be the length of the shortest path [180]; another possibility are diffusion kernels [125].

Divisive hierarchical strategies, in contrast, work in a top-down manner, starting with the entire graph and iteratively dividing it into smaller parts. An obvious splitting criterion are graph cuts [82]. Girvan and Newman [71] use the so-called edge betweenness measure, which is the number of node pairs with the shortest path passing through a specific edge; edges bridging between clusters are expected to have large betweenness values, so edges are removed from the graph in decreasing order of betweenness; the same technique has also been applied to weighted graphs [37]. Luo *et al.* combine this method with an agglomerative approach [149]; furthermore, some variants of the betweenness measure have been investigated [175]. As an alternative measure to assess bipartitionings of graphs, the modularity criterion has been introduced [160]; it compares the actual number of edges within the two parts with the expected connectivity in a random network. Finally, for the sake of efficiency, divisive clustering has been integrated with graph coarsening procedures [111].

Other methods directly partition the graph into a set of clusters, without using hierarchical decomposition steps. The most basic approach is to extract the connected components [153]. An increasingly popular method is spectral clustering; a tutorial on that topic can be found in [150]. Technically, it is based on eigen-decomposition of graph Laplacians, and has interpretations related to graph cuts and random walks. Markov clustering [53, 173, 216] is also motivated by random walks; it performs two alternating matrix operations; thus, in contrast to spectral clustering, one does not fix the number of clusters beforehand, but has to choose the parameters of the matrix operations. Furthermore, probabilistic latent variable models have been used for graph clustering [189], which often also allow cluster overlaps, i.e., the same node may belong to different clusters [39, 170].

4.4 Constrained Cluster Detection and Data Integration

In many bioinformatics applications, graph clustering and dense subgraph mining methods are augmented by integrating multiple data sources, the most common scenario being the combination of protein-protein interactions and gene expression data. One straightforward strategy is to build a new network where protein interaction links and coexpression links are simply pooled [170], or where the edge weights are determined as a function of multiple data sources [80]. Tanay *et al.* [205] also create one single network to analyze multiple genomic data at once; however, they use a bipartite network where each edge corresponds to one data type only. In both cases, edge weights of different datasets have to be normalized appropriately in order to be comparable in the integrated setting.

In contrast to that, other approaches keep the data sources separate and define individual constraints for each of them. Consequently, arbitrarily many datasets can be jointly analyzed without the need to take care of appropriate scaling or normalization. Within this class of approaches, there exist two main strategies to deal with profile data like gene expression measurements. In the first case, global similarity of profiles is considered. For that purpose, one possibility is to transform the profile information into a gene similarity network, where the strength of a link between two genes represents the profile similarity [172, 212, 213]; another approach is to learn cluster-specific models of gene expression profiles [170, 189]. In the second case, the cluster analysis is based on local profile similarities, i.e., context-specific patterns can be revealed [92, 95, 235].

4.5 Bicluster Analysis

In addition to homogeneous interaction graphs like protein interaction networks, bipartite graphs occur very frequently in biological data analysis. Similarly as in the previous sections, one central problem arising in that context is dense subgraph detection. A pattern of interest would then consist of a pair of node subsets, one from each partition, such that each node is connected to a large fraction of nodes from the other set. This can be seen as a special instance of the biclustering problem, which is very prominent in gene expression analysis [139, 151, 176, 207], but has also applications in text mining [47] and collaborative filtering [89]: given a data matrix (e.g., the adjacency weights of a bipartite graph), the goal is to extract subsets of rows that are similar with respect to subsets of columns; a particular pair of a row subset and a column subset (defining a submatrix) is called bicluster. This framework, which is also known as co-clustering or two-mode clustering [217], contrasts with traditional clustering approaches, which cluster either the rows according to their similarity across all columns, or the columns according to their similarity across all rows [105].

A multitude of bicluster detection methods has been developed during the last decade, and they can be grouped into similar categories as the subgraph discovery approaches described in Sections 4.1 to 4.3. First of all, one basic idea is to partition the bipartite graph that corresponds to the data matrix into distinct biclusters; using spectral clustering [150], this essentially amounts to singular value decomposition of the matrix, resulting in a set of block-diagonal patterns [47]. Kluger *et al.* [120] also use singular value decomposition, but treat rows and columns separately in the postprocessing, proposing a checkerboard structure of the matrix. Other approaches allow for a more flexible arrangement of biclusters, including bicluster overlap, while still optimizing a global objective function taking the whole matrix into account [135, 222, 227].

In contrast, enumerative approaches use local criteria based solely on individual biclusters. Many of them are motivated by the (weighted) bipartite graph formalism and refer to some density property of the subgraph. The widely used SAMBA method [205, 206] finds around each node the k heaviest subgraphs under additional connectivity restrictions. Sim and coauthors [196] fix the maximum number of missing edges tolerated per node as well as minimum size constraints. In [232], all maximal bicliques are detected and then further extended. Another work [22] searches for all bicluster patterns that satisfy homogeneity constraints with respect to the weight entries. Moreover, various other methods define specific bicluster criteria and solve the problem in a non-exhaustive way, by greedy strategies or approximation techniques, e.g., [28, 38, 88, 92, 127, 140, 151, 156, 239].

A related field is subspace clustering (introduced in [3]), see [130] for a recent review on that topic. Rather than looking at subgraph or submatrix patterns, these approaches are motivated by spatial considerations; that is, the rows of the data matrix are represented as data points in the feature space defined by the columns, typically Euclidean space. The goal is to extract axis-parallel subspaces with clusters in the form of dense clouds of data points; this resembles homogeneity criteria for biclusters, but it usually allows more flexibility in the spatial appearance (shape) of the clusters. Furthermore, the concept has been generalized to cluster detection in arbitrarily oriented subspaces. There also exist subspace clustering approaches for categorical data, e.g., [241]. These can be seen as variants of relational data mining, which will be discussed in Section 4.7. The next section describes a fundamental subtype of relational data mining.

4.6 Itemset Mining

For binary-valued data matrices, the simplest bicluster pattern of interest is a submatrix that purely consists of 1-entries. Such patterns can be exhaustively enumerated using itemset mining. This approach has been developed in the context of market basket analysis [4]. There, the data represent a set of transactions, where each transaction consists in a list of products (called items) that were purchased together. The task of frequent itemset mining is to find all sets of items that cooccur in more than m transactions. The frequent itemsets can be used to derive association rules of the following kind: “if a customer bought products A and B, he or she will also buy product C.” This information can, for instance, assist in improving shop layouts. Itemset mining has also been applied in the biological domain, e.g., gene expression analysis [45, 167].

The principal algorithmic idea behind itemset mining methods is based on the observation that any subset of a frequent itemset is frequent as well. The originally proposed Apriori algorithm [4] implements this in a level-wise search strategy, where the frequent sets on one level determine the candidate sets on the next level. Since then, there has been quite active research on improving the efficiency by introducing additional pruning rules and investigating alternative strategies to traverse the search space [215].

4.7 Relational Data Mining and Higher-Order Analysis

Itemset mining is only suitable for analyzing binary relations like the transaction-item association data described in the previous section. A natural extension is to consider higher-order relations, which involve more than two (key) attributes. For example, one could consider relations between purchased items, regions, and weeks of customer transactions. The generalized mining task can be formulated as follows [31, 32, 100, 103]:

Definition 3 (Relational Set Mining)**.** *Given an n-ary relation $R \subset D_1 \times \ldots \times D_n$, find all n-set patterns $(S_1, \ldots, S_n)$ such that $S_i \subset D_i$ for all $i = 1, \ldots, n$ and $S_1 \times \ldots \times S_n \subset R$.*

Generalizing the frequency criterion from itemset mining (see previous section), one can specify minimum size thresholds for $S_1, \ldots, S_n$. In the above definition, all domains D_i are assumed to be finite (i.e., categorical); their cardinality is denoted by $|D_i|$. An equivalent representation for such data is a $|D_1| \times \ldots \times |D_n|$ array A (also called data cube or tensor), where $A(d_1, \ldots, d_n) = 1$ if $(d_1, \ldots, d_n) \in R$, and $A(d_1, \ldots, d_n) = 0$ otherwise ($d_i \in \{1, \ldots, |D_i|\}$, $i = 1, \ldots, n$). Then, an n-set corresponds to a subarray that contains only 1-entries.

More generally, one can drop the constraint of binary values and consider tensors with arbitrary weight entries. Such higher-order datasets occur in different application fields like sales analysis [31], web mining [2, 100, 123], neuroscience [20], and computational biology [1, 18, 103, 243]. Therefore, methods that deal with multi-way arrays receive increasing attention in the data mining community. One of the most prominent topics is tensor decomposition (see [122] for a review), which can serve as a basis for clustering or anomaly detection [124]; furthermore, decomposition approaches can assist in analyzing dynamic changes in tensors [203].

The goal of tensor clustering is to partition each dimension of the tensor into a predefined number of clusters such that the resulting multi-way clusters are as homogeneous as possible [17]. This can be approximated by combining the results from clustering individual dimensions separately [101]. Zhao and Zaki [243] also mine for homogeneous clusters, but instead of specifying the number of clusters, they fix thresholds regarding the homogeneity of values along each dimension and detect overlapping cluster patterns (in the three-way case). Relational models [112] focus on binary-valued tensors, aiming at partitioning them into blocks that contain either mostly ones or mostly zeros. Finally, there exist approaches that deal with multiple relations or tensors at the same time, searching for clusters (communities) [17, 143] or association rules [51].

Part II

Set Enumeration based on Interaction Density – a Reverse Search Approach

5 Module Mining in Weighted Interaction Networks

This chapter presents an enumerative approach to identify cluster patterns in the most basic type of structured data, namely graphs or networks. As described in Section 3.1, a network consists of a set of nodes and a set of pairwise interactions represented by edges. Here, we focus on undirected interactions, but allow for interaction weights. In this context, a cluster is defined as a set of densely interacting nodes. After motivating the concept of enumeration in cluster detection tasks [66], we describe an algorithm to extract clusters from a given input network. It follows a general framework called reverse search, which has been introduced by Avis and Fukuda [14]. Several extensions are proposed, in particular the integration of constraints from other data sources, which are exploited using ideas from itemset mining (Section 4.6). The most central technical results of this chapter are covered in an earlier publication by the author [67].

5.1 Motivation

The problem of identifying clusters (also called *modules* or *communities*) in large graphs has been extensively studied, see [186] for a survey. One fundamental approach is clique discovery, i.e., the enumeration of fully connected subgraphs from an unweighted input graph (e.g., see [198]). As real-world datasets are usually incomplete, various methods relax this criterion by tolerating missing edges to a certain extent [81, 166, 214, 242]. In our approach, we relax the search criterion even further by explicitly considering edge weights. This allows for a more fine-tuned cluster analysis than a threshold-based preselection of edges [57, 166]. Edge weights naturally arise in systems biology tasks. For instance, in the context of protein interaction networks, they are often used to indicate the experimental evidence for a specific interaction [99], which helps to reduce false positive predictions in the analysis.

Beside these enumerative approaches to cluster detection, there exist a number

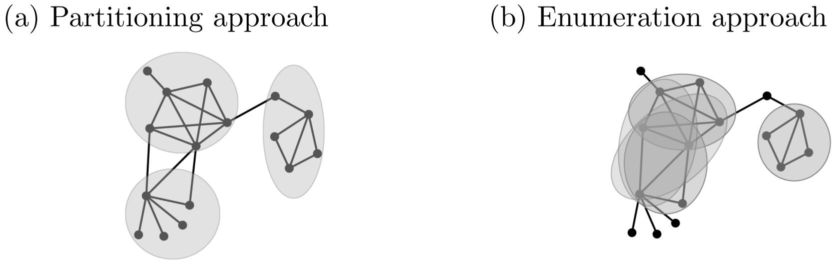


Figure 5.1: Different cluster finding concepts: graph partitioning versus cluster enumeration. While partitioning methods return one clustering of the graph, enumeration methods discover all clusters that satisfy a certain density criterion.

of methods that employ local search techniques starting from a set of seed clusters [16, 55]. Finally, the most commonly used class of methods is based on graph partitioning [37, 160, 186, 216]. In contrast to the other approaches, which respect explicit criteria regarding individual clusters, partitioning methods consider global characteristics of the graph in order to divide it into a set of mutually exclusive clusters.[1] Figure 5.1 illustrates the conceptual difference between partitioning and enumerative strategies. The partitioning approach is very suitable to obtain an overview of the structure in the data; nodes are grouped into clusters, and by replacing each cluster with a representative node, one can get a condensed representation of the input graph. However, cluster overlaps cannot be captured, and true clusters might be hidden by a large number of "satellite" nodes that are assigned to the same partition. These problems are avoided by enumeration techniques, which directly control the properties of clusters via user-defined parameters and do not miss any solution that satisfies these properties; in particular, these methods naturally allow for overlapping clusters. On the other hand, the number of solutions can get very large; furthermore, it might be difficult to come up with universal requirements for clusters, because the characteristics of true clusters can vary quite much. But this kind of problem arises in any cluster detection approach: a specific choice of criteria results in a certain trade-off between the reliability of predicted clusters and the coverage of true clusters.

From a biological point of view, an enumerative cluster finding approach is appealing because it naturally allows for cluster overlaps and a systematic consideration of cluster constraints. For motivation, let us consider the problem of predicting protein complexes from interaction networks. There are two typical overlap scenar-

---

[1] Some methods, e.g., Markov Clustering [216], allow marginal overlaps between different partitions. However, this usually concerns only negligibly few nodes in total, compared with the overlaps produced by enumerative methods.

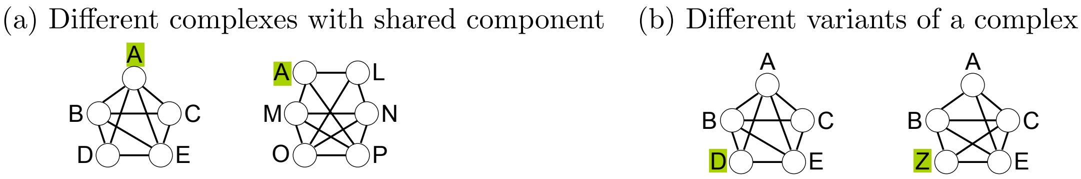

(a) Different complexes with shared component
(b) Different variants of a complex
A
B
C
D
E
L
M
N
O
P
Z

Figure 5.2: Schematic view of typical overlap scenarios in protein complex analysis: (a) the same protein can appear in different functional complexes, and (b) one complex can appear in different variants that share the same core, but have different extensions.

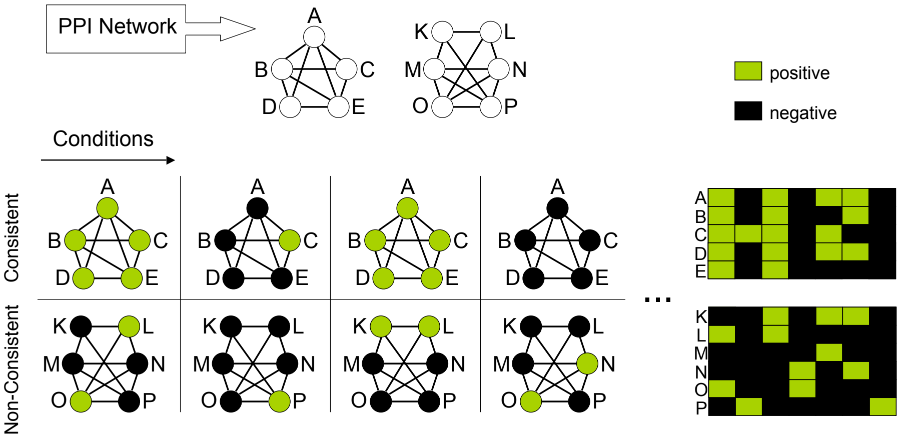


Figure 5.3: Integration of profile data. The combination of protein-protein interaction (PPI) and external profile data for proteins across different conditions allows to focus on clusters with consistent behavior of all nodes in a subset of conditions.

ios. First, it is a known fact that the same component may belong to different functional complexes [83] (Figure 5.2 (a)). Second, one complex can appear in several slightly different variants that share the same core part (Figure 5.2 (b)). In particular, the composition of complexes can change in dependence of the organism, the cell type, the environmental conditions, and the developmental stage [63]. Therefore, it is promising to integrate additional data sources during the network analysis, such as gene expression profiles, evolutionary conservation, subcellular localization, or phenotypic properties. This information helps to focus the search on clusters that are biologically relevant. For illustration, let us consider binary-valued profile data, indicating the state of each protein (e.g., present or absent) across multiple conditions. Then, dense modules are more likely to appear as a complex in the living cell if all member proteins consistently have the same state in a subset of conditions (see Figure 5.3). Moreover, context-specific changes can be revealed.

Our method allows to search for modules in the protein interaction network that have consistent profiles with respect to a subset of conditions. In contrast to previous integrative network mining methods (Section 4.4), it systematically identifies all modules satisfying a density criterion and optional consistency constraints. Before we describe the algorithmic framework, we introduce some definitions to formalize the problem of dense module enumeration.

5.2 Definitions

Let us consider an undirected weighted graph with node set V. For notational convenience, we assume that V is a set of consecutive indices starting from 1, i.e.,

$$V = \{1, \ldots, I\}\,, \tag{5.1}$$

where I is a positive natural number. Further, we denote by $|V|$ the size or *cardinality* of the node set, i.e., the number of nodes (here, $|V| = I$). The $|V| \times |V|$ *interaction weight matrix* is written as

$$W = (w_{ij})_{i,j \in V}\,. \tag{5.2}$$

It contains for each pair of nodes an entry, which corresponds to the weight of the connecting edge, if existent, and has a default value of zero otherwise.[2] In the following, we assume that the weights are given relative to their maximum possible value, so the normalized weights are bounded by 1:

$$w_{ij} \leq 1 \tag{5.3}$$

Negative weights are generally possible, but non-negative input matrices allow for additional speed-up techniques during the search and facilitate the elimination of redundant results (see Sections 5.3.4 and 5.4). Unweighted input graphs are translated into binary weight matrices with 1-entries for existing edges and 0-entries for missing edges.

A cluster or *module* is defined as a non-empty subset of nodes $U \subset V$, $|U| \geq 1$. The induced subgraph corresponding to a specific module U is represented by the following interaction matrix:

$$W|_U = (w_{ij})_{i,j \in U}\,. \tag{5.4}$$

---

[2] Other default values may be specified as well.

The average pairwise interaction weight within a module is referred to as the *module density*:

Definition 4 (Module Density)**.** *For a node set V with interaction weight matrix W and a module $U \subset V$, the density of U with respect to W is defined as*

$$\rho_W(U) = \frac{\sum\limits_{i,j \in U, i<j} w_{ij}}{|U|(|U|-1)/2}\,. \tag{5.5}$$

Here, self-interactions of nodes are not taken into account.[3] Because of the weight normalization, the largest possible density value is 1, conveniently expressed as 100%. For $|U| \leq 1$, we define $\rho_W(U)$ =100%.

Now we formulate the module mining problem we are interested in.

Definition 5 (Dense Module Enumeration)**.** *Given a graph with node set V and interaction weight matrix W, and a minimum density threshold $\theta > 0$, find all modules $U \subset V$ such that $\rho_W(U) \geq \theta$.*

The next section introduces an exact method to solve this problem. For unweighted input graphs, the problem is equivalent to pseudo-clique enumeration [214] (for $\theta < 100\%$) or clique search [110] (for $\theta = 100\%$).

5.3 Enumeration Algorithm

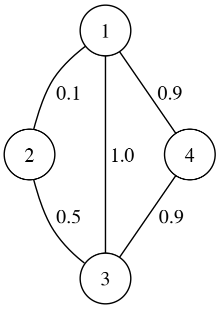
The problem of dense module enumeration can be solved efficiently by *reverse search*, a general algorithmic framework that has been published by Avis and Fukuda [14]. In the following, we first describe the basic search strategy of the dense module enumeration algorithm, which generalizes the unweighted graph approach described in [214]. The section continues with details regarding the implementation and is concluded with a complexity analysis.

5.3.1 Search Space

The heart of any enumeration algorithm is the definition of a search space structure that allows for efficient traversal and pruning. A canonical search scheme for set

---

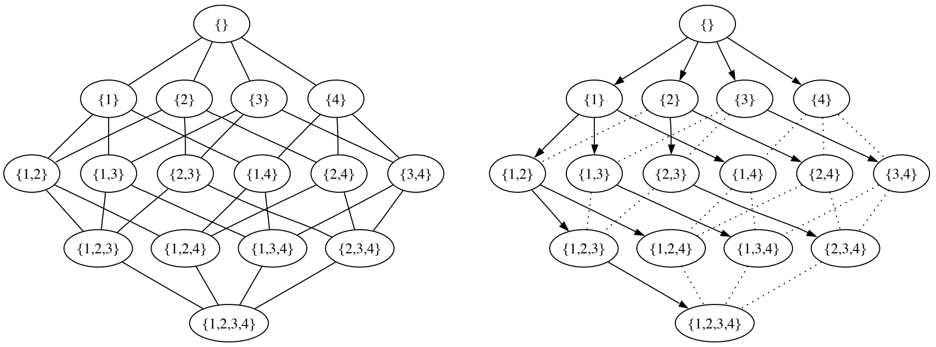
[3] However, they can be integrated in a similar way as node weights (see Section 5.6).



(a) Example input graph

(b) Corresponding weight matrix

	1	2	3	4
1	0	0.1	1.0	0.9
2	0.1	0	0.5	0
3	1.0	0.5	0	0.9
4	0.9	0	0.9	0

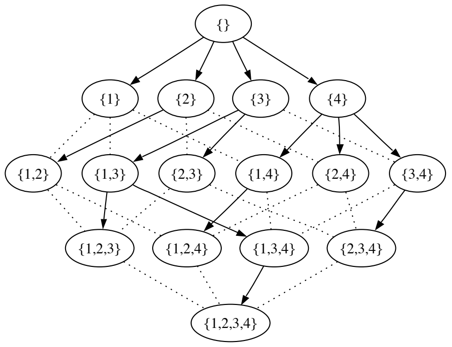


(c) Graph-shaped search space

(d) Lexicographical tree

(e) Densities of example modules

Module	Density
{1,2}	0.10
{1,2,3}	0.53
{1,2,4}	0.33
{1,3}	1.00
{1,3,4}	0.93



(f) Reverse search tree

Figure 5.4: Motivating example for module enumeration strategy. While a lexicographical traversal of the search space does not yield density guarantees, the module density is monotonically decreasing along each path of the reverse search tree.

enumeration tasks is to start with the empty set and then iteratively form larger sets by adding one element at a time. This defines a search space that is organized in multiple levels, having the empty set as its root; each time one moves a level downwards, the set obtains an additional member, i.e., the set cardinality increases by 1. Figure 5.4 (c) illustrates the search space of node sets for the example input graph with four nodes shown in Figure 5.4 (a). For an efficient search, it is crucial to avoid recomputations, i.e., the same set should not be visited several times. This is usually achieved by defining a tree structure that spans the original graph-shaped search space. In pattern mining approaches it is very common to use lexicographical set enumeration trees [19, 184, 240]. That means, a predefined order on the elements is exploited such that a set can only be extended by elements that are greater than any of the current member elements (see Figure 5.4 (d) for an example).

As the size of the complete set enumeration tree is exponential in the number of input elements, the practical applicability of a search procedure strongly depends on the definition of effective pruning rules, which prevent the exploration of irrelevant subtrees. To motivate our search algorithm for the dense module enumeration task, let us first consider the special problem of clique finding. In that case, the pruning is straightforward: if the current module is not a clique, we know that all supersets are non-cliques as well. More generally, this property is described as *downward closure* or *anti-monotonicity* [4]:

Definition 6 (Anti-monotonicity). *A function* $f : 2^V \to \mathbb{R}$ *is anti-monotonic if* $f(U') \geq f(U)$ *for all* U' *and* U *with* $U' \subset U \subset V$.

Here, 2^V denotes the power set of V, so the function f assigns a score to any subset of nodes in the input graph. For clique search, we define $f(U) = 1$ if U is a clique, and $f(U) = 0$ otherwise. If the current module has a score of 0, all descendants will have that score; hence, we can prune the search tree as soon as the clique criterion is violated.

While the lexicographical search tree can be used for clique search, it is not suitable for solving the general dense module enumeration problem because the density criterion is in general not anti-monotonic. For instance, while the density decreases when we go from $\{1, 3\}$ to $\{1, 3, 4\}$, it increases when stepping from $\{1, 2\}$ to $\{1, 2, 3\}$ (see Figure 5.4 (e)). Thus, the lexicographical structure does not provide guarantees regarding the maximum module density in subtrees, which makes it impossible to define effective pruning rules. The key idea for our dense module enumeration approach consists in the definition of a specific search tree where the density is monotonically decreasing on each path from the root to a leaf. We call this structure an *anti-monotonic set enumeration tree.*

Definition 7 (Anti-monotonic Set Enumeration Tree). *A set enumeration tree is anti-monotonic with respect to a function* $f : 2^V \to \mathbb{R}$ *if* $f(U') \geq f(U)$ *for all* U' *and* U *such that* $U' \subset U$ *is the parent of* U.

Note that in this definition, the anti-monotonicity depends on a specific parent-child relationship between sets, whereas Definition 6 considers general subset-superset relations. Figure 5.4 (f) shows a search tree that is anti-monotonic with respect to the module density (for the weighted example graph given in Figure 5.4 (a)). Regarding the clique criterion, both this search tree and the lexicographical tree (Figure 5.4 (d)) are anti-monotonic because the anti-monotonicity holds for any subset-superset relation.

5.3.2 Reduction Scheme

Next, we explain how to construct a module search tree that enforces anti-monotonicity of the module density. For that purpose, we need the definition of *degree* in weighted graphs.

Definition 8 (Degree). *Given a node $u \in U \subset V$, the (weighted) degree of u with respect to the module U is defined as*

$$\deg_U(u) = \sum_{j \in U, j \neq u} w_{uj} \,. \tag{5.6}$$

The degree obviously depends on the given weight matrix W. As W remains fixed during the whole algorithm, we omit an explicit reference to W in the notation.

The following lemma states a fundamental property of the module density.

Lemma 1. *Let $v \in U$ be a node with minimum degree in U, i.e., for all $u \in U$: $\deg_U(u) \geq \deg_U(v)$. Then, $\rho_W(U \setminus \{v\}) \geq \rho_W(U)$.*

Proof. Using the formulae for module density and degree from Definition 4 and Definition 8, respectively, we can rewrite the following expression:

$$\begin{aligned}
&\rho_W(U \setminus \{v\}) - \rho_W(U) \\
&= \frac{\sum\limits_{i,j \in U \setminus \{v\}, i<j} w_{ij}}{(|U|-1)(|U|-2)/2} - \frac{\sum\limits_{i,j \in U, i<j} w_{ij}}{|U|(|U|-1)/2} \\
&= \frac{\Big(\sum\limits_{i,j \in U, i<j} w_{ij}\Big) - \Big(\sum\limits_{j \in U} w_{vj}\Big)}{(|U|-1)(|U|-2)/2} - \frac{\sum\limits_{i,j \in U, i<j} w_{ij}}{|U|(|U|-1)/2}
\end{aligned}$$

$$
\begin{aligned}
&= \frac{\Big(\sum\limits_{i,j\in U, i<j} w_{ij}\Big)\left(1-\dfrac{|U|-2}{|U|}\right) - \deg_U(v)}{(|U|-1)(|U|-2)/2} \\
&= \frac{\Big(\frac{1}{2}\sum\limits_{u\in U}\deg_U(u)\Big)\dfrac{2}{|U|} - \deg_U(v)}{(|U|-1)(|U|-2)/2} \\
&= \frac{\frac{1}{|U|}\sum\limits_{u\in U}\deg_U(u) - \deg_U(v)}{(|U|-1)(|U|-2)/2} \\
&\geq 0
\end{aligned}
$$

The inequality holds because of the assumption that v is a minimum degree node in U. In summary, it follows that $\rho(U \setminus \{v\}) \geq \rho(U)$.

□

For unweighted graphs, this property is also known as “weak anti-monotonicity” and has for instance been exploited as auxiliary pruning criterion in graph pattern mining [245]. Furthermore, iterative removal of minimum degree instances has been used to approximate dense subgraphs [12].

In the context of dense module enumeration, the lemma yields the key for defining an anti-monotonic search tree, similarly to the pseudo-clique enumeration approach in [214]. Namely, we define the parent of a certain module as the module that is obtained by removing a minimum degree node. If there exist several minimum degree nodes, we apply an arbitrary predefined rule to select one among them, in order to ensure the uniqueness of the parent. For that purpose, we exploit a specific order on the nodes; as the nodes are represented by a set of indices (see Equation 5.1), we here simply use the order of natural numbers. With this, we define the parent-child relationship between modules as follows.

Definition 9 (Module Parent)**.** *Given a module U, let $v \in U$ be the node with the smallest index among the minimum degree nodes, i.e.,*

$$\forall u \in U \setminus \{v\} : \quad [\deg_U(v) < \deg_U(u)] \vee [\deg_U(v) = \deg_U(u) \wedge v < u] \ .$$

Then, $U \setminus \{v\}$ is the parent of U.

As the parent is a subset of the original module, this parent construction rule is also called *reduction scheme.* It defines a tree structure on the search space of modules, which has two important properties: anti-monotonicity and *completeness (module reachability).* Lemma 1 implies that parents have at least the same density as their children, so the tree is guaranteed to be anti-monotonic. The second property refers to the fact that the tree covers the whole search space: by iterative application of the reduction scheme, any module is transformed into the empty set, hence it is reachable on a path from the root.

5.3.3 Search Procedure

Given these properties, the enumeration procedure is quite straightforward. We traverse the module tree by depth-first search, starting from the empty set and recursively generating children on demand as long as the density threshold is satisfied. However, there remains one difficulty: while the reduction scheme allows to go from a child to its parent (i.e., bottom-up), it is not possible to directly derive the children of a given module in a top-down search. Therefore, we have to generate all direct supersets of the current module and check whether they belong to this parent or not. This paradigm is known as *reverse search principle* [14]. For the sake of clarity, we reformulate the conditions of the parent-child relationship from a top-down perspective, i.e., as it is used during the search process.

Definition 10 (Module Child)**.** *Let U be a module and $v \in V \setminus U$. The extended module $U^* := U \cup \{v\}$ is a child of U if and only if*

$$\forall u \in U : \quad [\,\deg_{U^*}(v) < \deg_{U^*}(u)\,] \;\vee\; [\,\deg_{U^*}(v) = \deg_{U^*}(u) \;\wedge\; v < u\,]\ .$$

With this, the dense module enumeration method boils down to the simple pseudocode shown in Algorithm 1. In the beginning, the module U is set to the empty set. In each iteration of the algorithm, we build an extended module candidate with every node that is not yet contained in U. If it satisfies the density criterion and is indeed a child of U, the search is continued recursively. The correctness of this algorithm follows directly from the anti-monotonicity and completeness properties discussed above; i.e., it finds all modules that satisfy the given density threshold.

Aside from the presented algorithm, several variants of the search scheme are conceivable. For instance, we can traverse the search tree in a breadth-first manner. That means, we first visit all solutions of a certain cardinality n before we go to the next level, which corresponds to cardinality $n+1$. In many applications, a depth-first

Algorithm 1 Dense module enumeration (DME) for node set V, interaction weight matrix W, and minimum density threshold θ. U represents the current module; in the initial method call, U is the empty set.

```
DME $(V, W, \theta, U)$ :
  for each $v \in V \setminus U$ do
    if $\rho_W(U \cup \{v\}) \geq \theta$ and $U \cup \{v\}$ is child of $U$ then
      DME $(V, W, \theta, U \cup \{v\})$
    end if
  end for
  output $U$
```

traversal is preferable to the breadth-first search because it requires only the storage of the current path through the search tree[4], whereas the latter has to keep track of all the solutions in the previous level. Another option is to perform a top-k search: instead of defining a density threshold, one specifies the desired number of solutions, k, and the method yields a set of k solutions with the largest density values. For this, we maintain a module list that is sorted according to the density criterion. In each step, the top module is expanded into its children, and the list is updated accordingly. This approach is particularly useful in conjunction with minimum size constraints (Section 5.7.3). Beyond that, it is possible to split the search into several phases with step-wise decreasing density thresholds. This works as follows: in the beginning, we set a stringent density threshold and perform the usual dense module enumeration algorithm; now, if we keep track of the border in the search tree where pruning has taken place, we can later resume the search at those sites to find modules with respect to a lower density threshold. This stop-and-continue strategy can be repeated as often as desired. Finally, one could in principle design search schemes that start from the full set instead of the empty set, and iteratively build candidate modules of reduced set size rather than letting the modules grow. However, we do not further pursue this idea because in our application scenarios, the expected module size is small in comparison with the total network size.

5.3.4 Implementation Details

The central step of a reverse search algorithm is the generation of children for the current solution. Therefore, engineering of this process is important for the efficiency of the method. Here, we describe some additional details for the implementation of the dense module enumeration approach.

---

[4] In fact, it would even be sufficient to store the previous solution and the current candidate, as the ancestor solutions and their next candidates can be recomputed.

First of all, the density of a module candidate and the degree values of its nodes can be calculated incrementally. For that purpose, we maintain an array d of length $|V|$ where we store the degree of each node with respect to the current module U; more precisely, it contains for the nodes in U the degree value with respect to U, and for all nodes $v \in V \setminus U$ the degree value looking one potential extension step ahead, i.e.,

$$d_U(v) = \begin{cases} \deg_U(v) & \text{if } v \in U \ , \\ \deg_{U \cup \{v\}}(v) & \text{if } v \in V \setminus U \ . \end{cases} \tag{5.7}$$

In addition, we keep track of the *total weight* of the current module, which is equivalent to the following expressions:

$$\text{totalWeight}(U) = \sum_{i,j \in U, i<j} w_{ij} = \rho_W(U)\,|U|(|U|-1)/2 = \tfrac{1}{2}\sum_{u \in U} \deg_U(u) \tag{5.8}$$

This allows to check in constant time whether the candidate $U \cup \{v\}$ satisfies the density criterion:

$$\rho_W(U \cup \{v\}) \geq \theta \iff \text{totalWeight}(U) + d_U(v) \geq \theta\,|U|(|U|+1)/2 \tag{5.9}$$

For each successful candidate $U \cup \{v\}$ we further investigate whether it actually is a child of the module U. For this, we have to test the conditions given in Definition 10, which requires $O(|U|)$ operations (assuming constant access to the entries in W): we have to determine the values of $\deg_{U \cup \{v\}}(u) = d_U(u) + w_{uv}$ for all $u \in U$ and compare them with $\deg_{U \cup \{v\}}(v) = d_U(v)$. In several cases, however, it is possible to skip these computations and decide in constant time whether $U \cup \{v\}$ is a child or not. The following two lemmata describe such speed-up rules.

Lemma 2. *Given a module U with respect to the interaction weight matrix W, let $u^* \in U$ be the previously added node. Let us consider a node $v \in V \setminus U$. If W is non-negative, the following rule holds:*

$$[d_U(v) < d_U(u^*)] \vee [d_U(v) = d_U(u^*) \wedge v < u^*] \implies U \cup \{v\} \textit{ is a child of } U$$

Proof. By definition, u^* is the smallest among the minimum degree nodes in U. As W contains only non-negative entries, we obtain for $u \in U$ $\deg_{U \cup \{v\}}(u) = d_U(u) + w_{uv} \geq d_U(u) \geq d_U(u^*)$. Further, $\deg_{U \cup \{v\}}(v)$ is equal to $d_U(v)$ by definition. In the case of $d_U(v) < d_U(u^*)$, it follows that $\deg_{U \cup \{v\}}(v) = d_U(v) < d_U(u^*) \leq \deg_{U \cup \{v\}}(u)$ for $u \in U$, so v is the unique node with minimum degree in $U \cup \{v\}$. In the second case, an analogous derivation shows that v is the node with smallest index in the set of minimum degree nodes in $U \cup \{v\}$. □

Lemma 3. *Given a (normalized) weight matrix* W *and a cluster* U*, let* $u^* \in U$ *be the previously added node. For* $v \in V \setminus U$*, the following rule holds:*

$$\begin{aligned}&[d_U(v) > d_U(u^*) + 1] \vee [d_U(v) = d_U(u^*) + 1 \wedge v > u^*]\\ &\quad\Longrightarrow\ U \cup \{v\} \textit{ is not a child of } U\end{aligned}$$

Proof. By assumption, $w_{ij} \leq 1$ for all $i, j \in V$, so $\deg_{U\cup\{v\}}(u^*) = d_U(u^*) + w_{u^*v} \leq d_U(u^*) + 1$. If $d_U(v) > d_U(u^*) + 1$, it follows directly that $\deg_{U\cup\{v\}}(v) = d_U(v) > \deg_{U\cup\{v\}}(u^*)$, so v cannot be a minimum degree node. For the case that $d_U(v) = d_U(u^*) + 1 \wedge v > u^*$, a similar argumentation shows that $U \cup \{v\}$ cannot be a child. □

In the case of binary-valued interaction matrices, further efficiency improvements are possible (see [214]). In particular, it has been proposed to organize the set of remaining candidate nodes (i.e., $V \setminus U$) into buckets, according to their (integer) degree values with respect to the current module. This allows to process the module candidates in the order of decreasing density, and buckets can be totally ignored if the corresponding degree value is too low to satisfy the density threshold. Similarly, we could maintain for weighted interaction matrices a priority queue where the candidate nodes are sorted by decreasing degree. This can be particularly useful in sparse data, where each module extension step requires only few updates of the data structure. For simplicity, we stick in our analysis to the implementation based on degree arrays.

Whenever a candidate $U \cup \{v\}$ turns out to be a true child, we have to update the total weight and the degree array before we can search for its own children:

$$\begin{aligned}\text{totalWeight}(U \cup \{v\}) &= \text{totalWeight}(U) + d_U(v)\\ d_{U\cup\{v\}}(j) &= d_U(j) + w_{vj} \quad \text{for all } j \in V,\ j \neq v\end{aligned}$$

This requires $O(1)$ and $O(|V|)$ operations, respectively.

5.3.5 Complexity

In combinatorial enumeration problems, the output size (i.e., the number of solution patterns) can be exponential in the input size. As an extreme example, a fully connected input graph with node set V contains $2^{|V|}$ cliques; in other words, each subset of nodes appears in the output. Therefore, rather than the total running time, a conventional complexity measure for enumeration methods is the time between two consecutive solution patterns, which is called *delay* [72, 73, 178]. In the following, we show that the reverse search algorithm for dense module enumeration has polynomial delay.

Using the degree array implementation described in the previous section, each recursion step of Algorithm 1 needs $O(|V| + |V \setminus U| \cdot |U|)$ operations, for updating the degree array and generating the children of the current module U. In the worst case, we perform for each candidate node a temporary update for the degree values of the nodes in U and compare them to its own degree value. However, in practice, the number of operations is typically much smaller because many candidates already fail at the density check, and the rules from Lemma 2 and Lemma 3 often allow to circumvent the temporary update step. To estimate the delay to the subsequent solution pattern, we consider a small modification of the algorithm: if the recursion depth is odd, we output the module before the recursive calls, and otherwise afterwards. Thereby, any three consecutive iterations of the code yield at least one output.[5] This computational trick is known as the odd-even output method [158, 214]. For illustration, we show in Figure 5.5 an example execution of the algorithm. Now, the delay has the same complexity as the execution of one recursion step, i.e., $O(|V| + |V \setminus U| \cdot |U|)$, where U is the current solution. More generally, let the $U_{\max}$ be the largest solution pattern; then, the delay is bounded by $O(|V| \cdot |U_{\max}|)$, so it is at most quadratic in the number of nodes in the network (in practice $|U_{\max}| \ll |V|$). We summarize our (worst-case) analysis in the following theorem.

Theorem 1. *The dense module enumeration problem can be solved by a reverse search algorithm with polynomial delay. In particular, for an input graph with node set V, the delay has a complexity of $O(|V|^2)$.*

In contrast to that, straightforward branch-and-bound strategies might need exponential time between two outputs because they have to solve an NP-complete problem in each recursive step [214]. Moreover, it is worth mentioning that the reverse search approach is directly compatible with distributed computation because different branches of the search tree can be investigated in parallel. Finally, the memory requirements of the recursive implementation are also polynomial in the

---

[5] Note that without this modification, the algorithm would output solutions only after having reached leaf modules of the search tree (i.e., after a sequence of recursive calls).

input size. For each recursive call, we store the current module U and the degree array of length $|V|$. Hence, the space complexity depends on the maximum recursion depth, $|U_{\max}|$, and is given by $O(|U_{\max}| \cdot |V|)$ plus the space needed for the input matrix. Using a simple implementation with a full matrix representation, the total space complexity of the algorithm amounts to $O(|V|^2)$, but improvements for sparse settings are conceivable. Note that it is not necessary to keep all previous solutions in memory.

5.3.6 Excursus: Reverse Search Applications

The technique of reverse search provides a feasible solution strategy for various types of enumeration problems. To define a reverse search method for a particular application, one has to specify

- a multi-level search space and
- a reduction scheme that guarantees anti-monotonicity and completeness.

The most prominent application field of reverse search are graph-related enumeration problems. Beside the problem of dense module enumeration discussed above, the framework can be used to enumerate all spanning trees and all connected modules of a graph as well as to derive all topological orderings of a directed acyclic graph [14]. For instance, in the case of connected module enumeration, the search space is the same as for dense module enumeration; one possible reduction scheme is to select the node with the smallest index among all nodes that are not *articulation points*, where an articulation point or cut vertex is a node that is essential for the connectivity of the induced subgraph (i.e., removal of an articulation point produces a disconnected module). Also, string problems can be solved using the reverse search paradigm, e.g., enumeration of maximal motifs in a sequence [9, 10]. Finally, reverse search is very useful in computational geometry applications. Example tasks are vertex enumeration in a convex polyhedron, cell enumeration in a hyperplane arrangement, and enumeration of triangulations of a set of points in the plane [14].

5.4 Output Representation

As enumerative approaches potentially return a large solution set, we discuss in this section how to obtain a user-friendly representation of the dense module enumeration output. In particular, direct submodules of other solutions can be efficiently

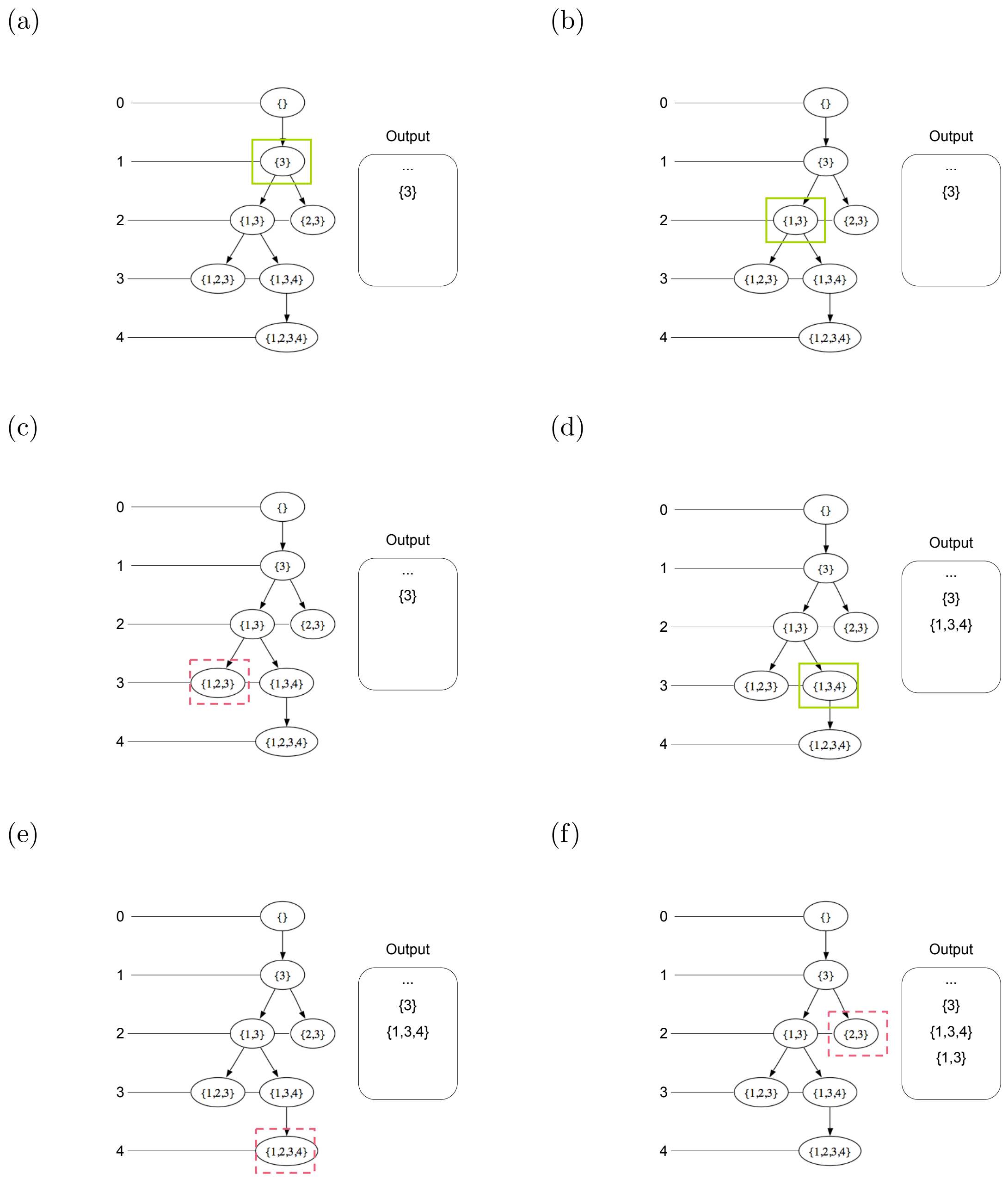


Figure 5.5: Illustration of reverse search with the odd-even output method. Sequence of traversal steps for a subtree of the example in Figure 5.4. The boxes indicate the module that is currently investigated. The green, solid boxes correspond to solution modules; the red, dashed boxes correspond to candidates that are pruned. For that, we assume a minimum density threshold of 0.9. Note that each solution module gives rise to a new recursive call. The numbered lines indicate the levels of recursion depth. After three recursive calls, two of which are completely executed ($\{1,3\}$ and $\{1,3,4\}$), the output contains three solution patterns.

eliminated, and the remaining modules are ranked according to their statistical significance.

5.4.1 Locally Maximal Modules and Leaf Modules

Usually, the user is not so much interested in modules that are subsets of other module solutions; rather, the most comprehensive modules are most relevant for further analyses. Therefore, the concept of *maximality* is widely used in pattern mining approaches [5, 19, 74, 77, 141]. In the context of dense module mining, it can be formulated as follows.

Definition 11 (Maximal Dense Module)**.** *A dense module is called maximal if it is not contained in any other dense module.*

A straightforward approach to obtain the set of maximal solutions would be to go for each newly detected module through all previous solutions, checking for inclusions. However, the structure of our reverse search algorithm allows us to reduce the number of solutions in the output in a meaningful way without any additional costs. We simply set a flag that indicates whether there exists a direct supermodule (i.e., a module with one additional node) that also satisfies the minimum density threshold (see Algorithm 2). If that is the case, we do not output the current module, otherwise we do. This yields us the set of all *locally maximal* dense modules.

Definition 12 (Locally Maximal Dense Module)**.** *A dense module U is called locally maximal if for all $v \in V \setminus U$, $U \cup \{v\}$ does not satisfy the minimum density threshold.*

As this definition looks only one step ahead, a module with this property can in principle violate the (stricter) maximality criterion. However, in practice the local maximality criterion successfully eliminates most non-maximal modules. If the density threshold θ is equal to 1, a locally maximal module is always maximal. Looking several extension steps ahead is in our search procedure only possible with respect to the own descendants. Alternatively to listing locally maximal modules, we could report all modules that do not have any dense descendants, i.e., all leaf nodes of the pruned search tree. A pseudocode for this is shown in Algorithm 3. It is easy to see that the set of leaf modules includes the set of locally maximal modules: by the definition of local maximality, any local maximal module cannot have descendants that are solutions. The other way round, a leaf module is not necessarily locally maximal. Therefore, we usually prefer the local maximality criterion to achieve a more compact result set. But if additional module filtering criteria are applied, the

Algorithm 2 Enumeration of locally maximal dense modules for node set V, interaction weight matrix W, and minimum density threshold θ. U represents the current module; in the initial method call, U is the empty set.

```
DME_lmax $(V, W, \theta, U)$ :
  locallyMaximal = true
  for each $v \in V \setminus U$ do
    if $\rho_W(U \cup \{v\}) \geq \theta$ then
      locallyMaximal = false
      if $U \cup \{v\}$ is child of $U$ then
        DME_lmax $(V, W, \theta, U \cup \{v\})$
      end if
    end if
  end for
  if locallyMaximal then
    output $U$
  end if
```

Algorithm 3 Enumeration of dense leaf modules for node set V, interaction weight matrix W, and minimum density threshold θ. U represents the current module; in the initial method call, U is the empty set.

```
DME_leaf $(V, W, \theta, U)$ :
  isLeaf = true
  for each $v \in V \setminus U$ do
    if $\rho_W(U \cup \{v\}) \geq \theta$ and $U \cup \{v\}$ is child of $U$ then
      isLeaf = false
      DME_leaf $(V, W, \theta, U \cup \{v\})$
    end if
  end for
  if isLeaf then
    output $U$
  end if
```

leaf criterion can sometimes be checked much more efficiently than the local maximality, as we discuss in Section 5.5. In any case, one could perform a straightforward postprocessing step to remove all non-maximal modules or select results according to other criteria of interest, but this is greatly facilitated if as many non-informative modules as possible are already discarded during the search.

5.4.2 Module Ranking

Even after filtering the results for local maximality or other predefined criteria, the solution set might be large. Therefore, it is important to provide a meaningful ranking criterion for the discovered modules. A widely used concept to measure the uncommonness or statistical significance of patterns are p-values. In general, the p-value of a certain pattern is defined as the probability that a randomly selected pattern of equal size is at least as "good" as the given pattern [21]. In our case, the statistics to measure the quality of an outcome is its density. Regarding the mechanism for random selection, different choices are conceivable. Given a module U, we here simply assume that a set of $|U|$ different nodes is randomly selected from the network at hand. The probability that this produces a module with at least the same density as the given pattern U is calculated by the following expression:

$$p_W(U) = \left|\{U' \subset V : |U'| = |U| \wedge \rho_W(U') \geq \rho_W(U)\}\right| \Big/ \binom{|V|}{|U|} \tag{5.10}$$

The completeness of our dense module enumeration approach enables us to determine the numerator exactly, i.e., we can compute for each detected module an *exact p-value* [21]. For that purpose, we have to keep track of the densities and the sizes of all solutions we encounter during the search. If we maintain for each module size a sorted list of densities, one pass is sufficient to obtain the p-values for all output module of that size. Lower p-values indicate more remarkable or surprising patterns, so the results are sorted according to their p-values in increasing order. This ranking scheme captures the intuition that the importance of a module should increase with its size and density; still, from a theoretical point of view, it is more principled than the ranking criterion used in [16], which is the product of size and density. Furthermore, it specifically refers to the network at hand, in contrast to significance measures that are based on reference network models [128]. This can be advantageous because real networks might violate the assumptions of network models.

Other module finding approaches calculate p-values by assessing the number of interactions within the module relative to the number of interactions between module nodes and the remaining network [137]; in that case, interaction weights are ignored. Finally, it is very common to estimate *empirical p-values* by generating multiple random networks with the same degree distribution as the given input network [191]. Note that the (exact) p-value criterion formulated above does not take the degree into account. More sophisticated approaches are conceivable; for instance, one could consider random modules that do not only have the same size as the pattern of interest, but also satisfy some constraints regarding the node degrees with respect to the whole network. This refined analysis could reveal significance differences among modules with similar size and similar density.

5.5 Degree-Based Module Criteria

So far, our primary criterion of interest has been the density of interactions across the whole module. Here, we discuss more specific criteria that refer to individual module nodes.

5.5.1 Minimum Degree Criterion

The module density criterion is very flexible, allowing for missing or weak edges to a certain extent; in particular, for a fix density threshold, the flexibility increases with growing module size. While this property is advantageous in many situations (e.g., for finding modules supported by a large number of weak edges as well as modules containing few, but very strong edges), there can also occur undesired artifacts: a large dense module might tolerate the addition of several loosely connected nodes without violating the density threshold. This effect can blow up the number of solutions considerably, even after selection for (local) maximality, because the same core module can appear in many different variants, each time augmented by a few loosely attached, possibly irrelevant nodes. Of course, an obvious remedy would be to choose a stricter density threshold, but this could lead to the loss of other interesting solutions. Therefore, we introduce an optional filtering criterion for modules, which fixes a minimum degree threshold for module nodes.

Definition 13 (Minimum Degree Threshold)**.** *A module U satisfies the minimum degree threshold t if* $\deg_U(u) > t$ *for all* $u \in U$*.*

By default, we set $t = 0$, i.e., modules containing isolated nodes or nodes with negative degree are not considered as solutions. If the interaction matrix W contains only non-negative entries, it is easy to search for modules that are locally maximal with respect to the new combined criterion, consisting of a minimum density threshold and a minimum degree threshold; here, a solution module U is locally maximal if and only if $\forall v \in V \setminus U : \rho_W(U \cup \{v\}) < \theta \vee \min_{u \in U \cup \{v\}} \deg_{U \cup \{v\}}(u) \leq 0$. For this, we replace line 5 in Algorithm 2 with the following statement: **if** $d_U(v) > t$ **then** locallyMaximal = false **endif**. Here, $d_U(\cdot)$ denotes the corresponding entry of the degree array for the current module U (see Section 5.3.4). The node v is not necessarily a minimum degree node with respect to $U \cup \{v\}$; thus, the module $U \cup \{v\}$ might still violate the minimum degree criterion. However, in that case the module U does not satisfy it either (because of the non-negativity of W, the degree values can only grow by the addition of v), so this check does not discard any true solution. To ensure that the output contains only valid solutions, we further have to add in line 11 a condition that checks the minimum degree criterion. We can either store with each module U the result of the check preceding its generation as a child or check again the degree value of the previously added node u^* (which is by definition a minimum degree node): $d_U(u^*) > t$.

For mixed-sign input data, the situation is more complicated because the degree values do not monotonically increase with the extension of the module. So it can happen that U is a valid pattern, but $U \cup \{v\}$ not, although $d_U(v)$ is above the threshold. As an example, let us consider the following three-node graph: $V = \{v_1, v_2, v_3\}$, $w_{v_1v_2} = 0.5$, $w_{v_1v_3} = 1$, $w_{v_2v_3} = -0.6$. For $\theta = 0.3$ and $t = 0$, $\{v_1, v_2\}$ is a solution and $\{v_1, v_2, v_3\}$ not; however, $\rho_W(\{v_1, v_2, v_3\}) \geq \theta$ and $d_{\{v_1,v_2\}}(v_3) > t$. So we need a different strategy to determine local maximality. One solution is to check the degree values of all nodes for each module candidate $U \cup \{v\}$ that satisfies the density criterion, which requires a temporary update of the degree array (see Section 5.3.4). In that case, we cannot make use of the speed-up rule described in Lemma 3. Another possibility of output filtering is to keep all solutions where no descendant satisfies both the density and the minimum degree criterion, i.e., the leaf solutions. As the minimum degree is not anti-monotonic, we have to adjust Algorithm 3 such that it returns the existence of solutions from (multiple) recursive calls to the outer procedure and prevents it from producing an output (*multi-step look-ahead*). This strategy is more efficient because the minimum degree criterion must only be checked for true children, which are updated anyway, not for all dense candidates. In contrast to the setting where we consider only the module density criterion, here it is not clear a priori which strategy will reduce the output set more effectively.

5.5.2 Minimum Relative Degree and Quasi-Cliques

A drawback of the minimum degree criterion is that it specifies the threshold irrespective of the module size. That means, for low thresholds we will get undesired weakly connected extensions of large modules, as described above, whereas high thresholds a priori exclude small modules. Therefore, it seems to be promising to replace the combination of density and minimum degree criteria by the following *minimum relative degree criterion.*

Definition 14 (Minimum Relative Degree Threshold)**.** *The minimum relative degree threshold* γ *is satisfied for a module* U *if* $\deg_U(u)/(|U| - 1) \geq \gamma$ *for all* $u \in U$.

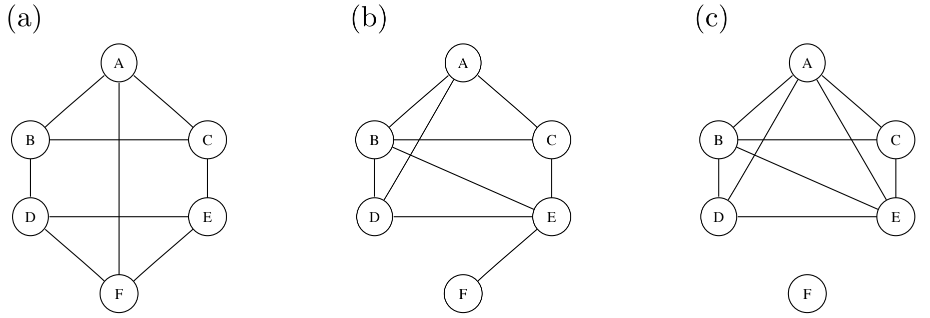


Figure 5.6: Minimum relative degree versus density. While the modules (a)-(c) all have the same density (3/5), the minimum relative degree varies: (a) 3/5, (b) 1/5, (c) 0.

This condition considers the density with respect to each individual module node and thereby guarantees a certain balance in the distribution of the weight among the nodes in a module, which is a reasonable requirement in many applications. For unweighted graphs, this kind of pattern is known as γ-*quasi-clique* [104, 145, 172, 242]:

Definition 15 (γ-Quasi-Clique)**.** *A node set* U *is a* γ*-quasi-clique if each node has edges to at least* $\lceil \gamma(|U| - 1) \rceil$ *other nodes in* U.

It is easy to verify that a module satisfying the minimum relative degree threshold γ has a density of at least γ. On the other hand, for a module with density $\geq \gamma$ the minimum relative degree is not necessarily greater than or equal to γ. Figure 5.6 illustrates this relationship between module density and minimum relative degree. Unfortunately, we cannot mine directly for modules that satisfy the minimum relative degree criterion. The reason is that it is inherently impossible to define an anti-monotonic reduction scheme. For instance, let us consider the module in Figure 5.6 (a). The minimum relative degree is 3/5. However, no matter which node we remove, the resulting module will have a minimum relative degree of 2/4. Therefore, we have to use a more general criterion during the search and perform a filtering step to select the actual solutions. One possible approach is to enumerate by reverse search all modules with density $\geq \gamma$ and use similar strategies as with minimum degree thresholds for mixed-sign data to filter the modules. For unweighted input graphs, alternative approaches have been developed, which are described in the following section.

5.5.3 Previous Work on Quasi-Clique Mining

We briefly review the major techniques for γ-quasi-clique enumeration used in existing work [104, 145, 172, 242]. The basic search strategy is depth-first search in a lexicographical set enumeration tree. As it lacks an anti-monotonicity property with respect to the quasi-clique criterion, the pruning is based on other characteristics. The first rule is connected with the diameter of γ-quasi-cliques. In general, the diameter of a subgraph is defined as the maximum shortest path length between any pair of nodes. For example, the graph in Figure 5.6 (a) has diameter 2. It turns out that the diameter of a γ-quasi-clique U can be bounded in dependence of the minimum relative degree threshold γ and the number of nodes $|U|$:

Lemma 4. *Let U be a γ-quasi-clique ($|U| > 1$). Further, let $\operatorname{diam}(U)$ denote the diameter of U. Then,*

$$\operatorname{diam}(U) \begin{cases} = 1 & \textit{if } 1 \geq \gamma > \frac{|U|-2}{|U|-1} \\ \leq 2 & \textit{if } \frac{|U|-2}{|U|-1} \geq \gamma \geq \frac{1}{2} \end{cases} .$$

Proof. In the first case, each node is directly connected to more than $|U| - 2$ other nodes in U, hence U is a clique. In the second case, each node $u \in U$ has edges to at least $\lceil (|U| - 1)/2 \rceil$ other nodes in U (according to the definition of γ-quasi-clique). We call these nodes the neighbor set of u with respect to the node set U, denoted by $N_U(u)$. Now let us consider a pair of nodes u_1, u_2. If there exists an edge between u_1 and u_2, the corresponding shortest path length is 1. Otherwise, $|N_U(u_1) \cup N_U(u_2)| \leq |U| - 2$ because u_1 and u_2 are excluded and there are $|U|$ nodes in total. Further, it follows from the assumption that $|N_U(u_1)| + |N_U(u_2)| \geq |U| - 1$. Putting both statements together, we obtain $|N_U(u_1) \cap N_U(u_2)| \geq 1$, i.e., u_1 and u_2 have at least one common neighbor. Consequently, there exists a path of length 2 from u_1 to u_2. □

For upper bounds of the diameter for smaller γ, we refer to [104]. With this lemma, we can restrict the set of candidate nodes for extending a given set of nodes:

Lemma 5. *Let $U \subset V$ be the current set of nodes and $u \in U$. Further, we denote by $N_V^b(u)$ the b-step neighborhood of u with respect to V, i.e., all nodes in V that are reachable from u by a path of length $\leq b$. If there exists a γ-quasi-clique Q with $U \subset Q \subset V$, each node $v \in Q \setminus U$ satisfies*

$$v \in \bigcap_{u \in U} N_V^b(u) \,,$$

where b is the upper bound of the diameter of a γ-quasi-clique.

Consequently, we can discard candidate nodes that are not included in the intersection of the b-neighborhoods of the current nodes. Moreover, one can derive degree-based pruning rules. Given a minimum size threshold m for γ-quasi-cliques, one can a priori remove all nodes v with $deg_V(v) < \lceil \gamma \cdot (m-1) \rceil$. Further degree-related pruning is possible during the search. If

$$\deg_U(v) + \deg_{V \setminus U}(v) < \lceil \gamma \cdot (|U| + \deg_{V \setminus U}(v)) \rceil$$

for a node set U and a node $v \in V \setminus U$, then v can be excluded from the extension candidates, because v does not satisfy the minimum relative degree criterion for any superset of $U \cup \{v\}$ [242]. As the exclusion of one node can influence the degrees of other nodes, this pruning procedure is performed iteratively. Beyond that, many additional refinements of γ-quasi-clique mining have been proposed, including degree-dependent bounds, look-ahead strategies, and candidate sorting (see [104, 145, 242] for details).

5.5.4 Discussion

One problem of the existing quasi-clique mining approaches is that the underlying search tree is not anti-monotonic. Consequently, indirect criteria have to be used to decide where pruning is possible. Often, one single criterion is not sufficient to achieve an efficient search. For instance, the diameter-based pruning is problematic for input graphs that contain highly connected nodes ("hubs") because the b-step neighborhoods of nodes and their intersection can be very large. Similarly, degree-based pruning techniques fail if there are too many nodes for which the degree exceeds a critical threshold. Therefore, it is crucial for the feasibility of the approach to combine several such criteria, as suggested in the literature [104, 145, 172, 242].

As the module density criterion is a direct generalization of the γ-quasi-clique criterion and allows to construct an anti-monotonic search tree, an interesting alternative would be to exploit the dense module enumeration strategy for quasi-clique mining. This is particularly promising considering the fact that the pruning rules from previous quasi-clique mining methods can also be integrated into the framework, thereby combining the advantageous properties of both approaches to achieve a high pruning potential. Furthermore, it would be interesting to extend the quasi-clique pruning concepts to the minimum relative degree criterion for weighted input graphs. Finally, a combination of topology-based and weight-based criteria could also be fruitful for cluster detection in weighted graphs. Here, we focused on search criteria that consider exclusively the interactions within a module. However, depending on the application scenario it can be desirable to take outgoing edges into account as well. In the literature, a number of different approaches to combine these two aspects have been studied (see, e.g., [149, 160, 162, 181]).

5.6 Integration of Node Weights

So far, the density criterion for modules takes only interaction weights into account. However, node weights play a role in many biological applications; they are commonly used to indicate the (measured or estimated) relevance of a node for the biological problem at hand. In this section, we discuss how to integrate node weights into the module mining process. In contrast to module finding approaches that use node weights to preprocess the input graph (i.e., remove low-weight nodes) [191], we directly incorporate them into the search criterion.

5.6.1 Definitions

Let us consider again an input network with node set V. We assume that each node $i \in V$ has an assigned *node weight*, denoted by o_i. With this, we define the *node density* of a module.

Definition 16 (Node Density)**.** *Given a module $U \subset V$ and node weights $o = (o_i)_{i \in V}$, the node density is defined as*

$$\rho_o(U) = \frac{1}{|U|} \sum_{i \in U} o_i \,. \tag{5.11}$$

That means, the node density corresponds to the average node weight within a module. For clarity, we use the term *interaction density* to refer to the previous definition of module density, the average pairwise interaction weight (see Definition 4). Now we introduce a *combined density* criterion, which includes interaction weights, node weights, and a calibration parameter α.

Definition 17 (Combined Density)**.** *Given a module $U \subset V$, the combined density is defined as*

$$\rho_{\alpha,W,o}(U) = \tfrac{1}{1+\alpha} \left(\rho_W(U) + \alpha \, \rho_o(U) \right) \,, \tag{5.12}$$

where $\alpha \geq 0$.

In words, the combined density is a weighted sum of the interaction density and the node density, where the node contribution obtains the α-fold weight of the interaction contribution. For convenience, interaction weights and node weights are normalized independently such that the maximum value is 1, respectively; i.e., $o_i \leq 1$ for all $i \in V$ and $w_{ij} \leq 1$ for all $i, j \in V$, $i \neq j$. Thus, both the maximum interaction density and the maximum node density are 1. Then, the maximum combined density is 1, since the scaling factors in the weighted sum in (5.12) are non-negative and sum up to 1.

By the definition of the combined density criterion, we can formulate the following generalized dense module enumeration problem.

Definition 18 (Generalized Dense Module Enumeration)**.** *Given a set of nodes* V*, an interaction weight matrix* W*, a node weight vector* o*, a calibration parameter* $\alpha \geq 0$*, and a density threshold* $\theta > 0$*, find all modules* $U \subset V$ *such that* $\rho_{\alpha,W,o}(U) \geq \theta$*.*

5.6.2 Algorithm

Remarkably, the generalized dense module enumeration problem can be reduced to the basic problem involving only interaction weights. More precisely, using the formulae of interaction density (5.5) and node density (5.11), we can rewrite the combined density (5.12) as follows:

$$\begin{aligned}\rho_{\alpha,W,o}(U) &= \frac{1}{1+\alpha}\left(\frac{\sum\limits_{i,j \in U, i<j} w_{ij}}{|U|(|U|-1)/2} + \alpha\,\frac{\sum\limits_{i \in U} o_i}{|U|}\right)\\ &= \frac{\sum\limits_{i,j \in U, i<j} \frac{1}{1+\alpha}\left(w_{ij} + \alpha\,\frac{(o_i + o_j)}{2}\right)}{|U|(|U|-1)/2}\end{aligned}$$

This directly suggests the following enumeration procedure:

1. Construct a transformed interaction weight matrix W^{new}:

$$w_{ij}^{\text{new}} = \tfrac{1}{1+\alpha}\left(w_{ij} + \alpha\tfrac{(o_i+o_j)}{2}\right) \tag{5.13}$$

2. Use the standard DME algorithm to enumerate all modules U that satisfy the following criterion:

$$\rho_{W^{\text{new}}}(U) \geq \theta \tag{5.14}$$

By the transformation, most interactions that formerly had a weight of zero obtain non-zero values, so the transformed interaction weight matrix is usually non-sparse.

5.6.3 Remarks

The previous sections explained how to enumerate modules with respect to a criterion that combines interaction density and node density. The user may specify a minimum threshold for the combined density as well as a parameter α to calibrate the two components. Another possible setting would be to consider individual thresholds θ_W and θ_o for interaction density and node density, respectively. Clearly, one can formulate independent module finding tasks for the two criteria. Each of them can be solved with an anti-monotonic search tree. For the task that exclusively considers the node density, this is simply achieved by the following definition of parent: the node set that is obtained by the removal of one minimum weight node. As the node weights do not depend on the other nodes in a module, we do not need the framework of reverse search, but can directly generate children by respecting a weight-based ordering of nodes.

For a given module, there does not necessarily exist a parent such that the anti-monotonicity is satisfied with respect to both criteria. Consequently, it is not possible to construct a combined search tree or to perform cross-tree pruning, i.e., if a module fails to satisfy one criterion, one still has to consider its descendants with respect to the other criterion. A straightforward solution strategy, which has been used in [181] for a similar problem, is to perform the search with respect to one density criterion, and filter the results with respect to the other one. Alternatively, we can search with the combined density criterion from Definition 17, setting the minimum density threshold to $\frac{1}{1+\alpha}(\theta_W + \alpha\, \theta_o)$. If the individual density criteria are satisfied, the combined density exceeds the threshold, so we do not miss any solution. However, note that the interaction density and the node density might balance each other; therefore, the solutions still have to be checked with respect to the interaction density and node density, respectively. The parameter α allows to trade off the importance of the two criteria a priori. In particular, if one criterion is much more likely to be satisfied, it is recommendable to focus on the stricter one during the search.

5.7 Constraint Integration

Often, it is desirable to restrict the dense module search by some additional criteria. Our enumeration framework allows to incorporate and systematically exploit various types of constraints defining further module characteristics or involving external datasets. In the following subsections, we exemplarily present some constraints that actively contribute to pruning during the search; constraint-based output filtering has been discussed in Section 5.5.

5.7.1 Constraints from External Data Sources

In many systems biology applications, the integration of background knowledge or other data sources is crucial to enhance the biological relevance of patterns. One useful criterion are consistency constraints with respect to external profile data, as illustrated in Figure 5.3. Given a discrete-valued profile for each node, we call a module consistent if all its nodes share a common subprofile. In the context of protein interaction data, we could for instance consider subcellular localization profiles, which indicate presence of proteins in each cellular compartment. With such data, the search for consistent patterns aims at the discovery of modules that are "realizable" in the living cell and therefore more plausible than a module without protein cooccurrence. Formally, we define the concept of consistency as follows.

Definition 19 (Consistency). *Let V be the node set of the input network and $U \subset V$ a module. Let $X = (x_{ij})_{i \in V, j \in C}$ be an auxiliary profile matrix, where C denotes the set of columns (representing different conditions) and the entries x_{ij} take values from a set of discrete states S. Given a state $s \in S$, a particular column $c \in C$ is said to be s-consistent with respect to U if*

$$x_{uc} = s \text{ for all } u \in U \, .$$

The number of s-consistent columns with respect to U is denoted by $f_s(U)$. The module U is consistent with respect to s if

$$f_s(U) \geq n_s \, ,$$

where n_s is a prespecified (non-negative) integer.

To control the consistency of module profiles, we fix minimum thresholds n_s for the frequencies $f_s(U)$ of the different states s. Then, we can formulate the constrained dense module enumeration task:

Definition 20 (Dense Module Enumeration with Consistency Constraints)**.** *Given a graph with node set* V *and weight matrix* W*, a density threshold* $\theta > 0$*, an auxiliary profile matrix* $(x_{ij})_{i \in V, j \in C}$ *with* $x_{ij} \in S$*, and integers* n_s *for all* $s \in S$*, find all modules* $U \subset V$ *such that* $\rho_W(U) \geq \theta$ *and* $f_s(U) \geq n_s$ *for all* $s \in S$*.*

If we are only interested in consistency with respect to one specific state, the thresholds for the other states are simply set to 0. For example, one might look for protein modules with consistent presence in a cellular compartment, ignoring patterns of coordinated absence. The consistency constraints help to concentrate on patterns that are of interest for the study at hand. In addition, they can lead to a considerable speed-up of the search procedure. Namely, we can exploit pruning techniques from traditional frequent itemset mining [4], based on the observation of anti-monotonicity (see Definition 6).

Lemma 6. *The consistency criterion satisfies the anti-monotonicity property, i.e.,*

$$U' \subset U \implies f_s(U') \geq f_s(U)\,.$$

Proof. Each column that is s-consistent with respect to U is also s-consistent with respect to U'. □

In other words, module extension cannot increase the frequency of consistent columns, see Figure 5.7 for illustration. When adding node F to the module {A,B,C,D,E}, the number of consistent 1-columns shrinks, whereas the number of consistent 0-columns is left unchanged. To determine the frequencies of consistent columns for {A,B,C,D,E,F}, only the columns that are consistent with respect to {A,B,C,D,E} need to be considered. The lemma implies that if a module itself does not satisfy the frequency requirements, no extension can satisfy them. Thus, they can be directly used as additional pruning criteria and thereby avoid that irrelevant parts of the search space are visited. That means, we simply check one further condition before doing the recursive call in line 4 of Algorithm 1. To perform local maximality filtering in the context of consistency constraints, we just have to make sure that the consistency is checked before the child conditions. More precisely, we add the consistency conditions to line 4 in Algorithm 2. These are then automatically taken into account when checking for local maximality. Alternatively, the

Module U	$f_1(U)$	$f_0(U)$
{A,B,C,D,E}	2	2
{A,B,C,D,E,F}	1	2

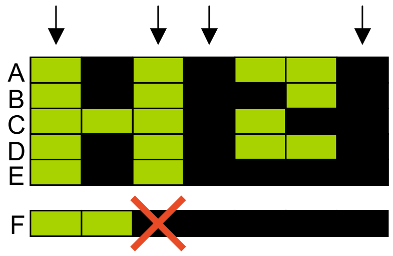


Figure 5.7: Anti-monotonicity of consistency constraints. Given the example profile on the right, the table shows the frequencies of consistent columns for the 1-state (green) and for the 0-state (black) before and after module extension.

output can be summarized according to the concept of closed patterns [171, 221]. There, the idea is to exclude subsets of other solutions only if they are supported by exactly the same set of consistent columns. The motivation behind this is that subsets of other solutions can still be informative if they have a different specificity profile (i.e., a larger set of consistent columns).

The described framework can be applied to incorporate any kind of anti-monotonic constraints. Furthermore, one can use arbitrarily many of those constraints at the same time, so it is possible to coanalyze several data sources, defining individual constraints for each of them. Beside profile consistency constraints, another prominent example for anti-monotonic constraints on external data are *node pair constraints*. Assume we have an additional weight matrix containing an entry for each pair of nodes. This could be a similarity or distance matrix that is based on another criterion or comes from another experiment. Then, constraints that define minimum or maximum weight thresholds for all node pairs in a module are anti-monotonic: if a module contains a pair that violates the threshold, all its supermodules will not satisfy the constraint. Finally, not all constraints that are potentially interesting have the anti-monotonicity property. For instance, the requirement that a certain percentage of module nodes fulfills some criterion is inherently non-anti-monotonic. Sometimes it might be possible to derive anti-monotonic bounds even if the criterion is not anti-monotonic. Otherwise, one can handle these constraints simply by output filtering (see Section 5.5), in which case they do not contribute to accelerating the search.

5.7.2 Connectivity Constraints

The density criterion for modules does not necessarily guarantee the connectivity of the corresponding subgraph. A subgraph is connected if every node is reachable from every other node via a path, i.e., a series of (non-zero) edges. Strictly speaking, it should even be a path of positively weighted edges, as negative edge weights usually indicate negative associations, so the corresponding nodes would be interpreted as antagonists rather than as members of the same group or community. For simplicity, we focus in the following discussion on interaction matrices with non-negative weights, although extensions to mixed-sign weights are possible. As already mentioned in Section 5.5, a module might tolerate a certain fraction of weakly connected or even disconnected nodes; also, it might fall apart into separate components. These problems arise in particular for large modules or low density thresholds.

However, in many applications it is meaningful to consider only connected modules as results. Unfortunately, we nevertheless have to visit disconnected modules during the search because the connectivity property is not anti-monotonic with respect to the density-based reverse search tree; i.e., connected modules may descend from disconnected ones, as the example in Figure 5.8 shows. In fact, there exists a reverse search tree that is anti-monotonic with respect to the connectivity criterion, as described in Section 5.3.6. But it is different from the density-based reverse search tree, and the combination of two search trees is difficult (see Section 5.6). Again, we could traverse the search tree for the density criterion and filter the results with respect to the connectivity criterion; an easy way to check whether a module is connected is a depth-first search traversal of the induced subgraph. However, the connectivity criterion can also contribute to the pruning of the density-based search tree:

Definition 21 (Isolated Node)**.** *Given a module U, a node $u \in U$ is called isolated if* $\deg_U(u) = 0$.

Lemma 7. *Let $W = (w_{ij})_{i,j \in V}$ be a weight matrix representing the input network such that $w_{ij} > 0$ if the nodes i and j are connected by an edge, and $w_{ij} = 0$ otherwise. Then, a module with two isolated nodes cannot have any connected descendant.*

Proof. Let us consider a connected module, on which we iteratively apply the reduction scheme, i.e., in each step we transform the current module into its direct ancestor in the search tree (Definition 9). During this process, we might encounter modules that include an isolated node (see Figure 5.8). Whenever this happens,

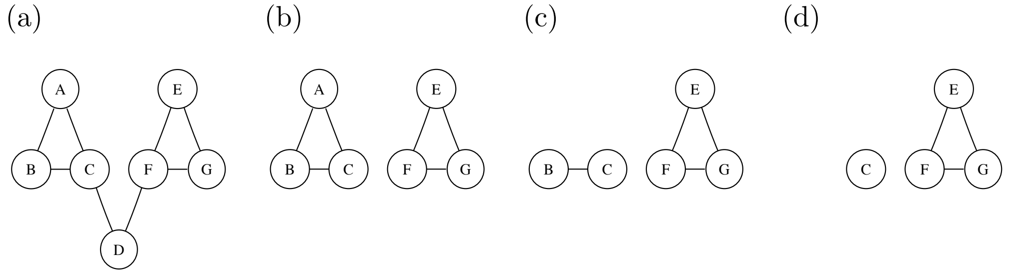


Figure 5.8: Example reduction of a module. It shows that a connected module (a) can have disconnected ancestors (b-d). In particular, one ancestor has an isolated node (d). For the uniqueness of parents, a lexicographical order on the nodes is assumed.

the isolated node(s) will be removed in the next reduction step(s) because they are the minimum degree nodes; as removals of isolated instances leave the degrees of other nodes unchanged, they cannot produce new isolated nodes. Hence, isolated nodes cannot be accumulated during the reduction. It remains to show that a single reduction step on a module without isolated nodes cannot produce two or more isolated nodes. Let us assume this would be possible. We denote by U the original module, i.e., $\deg_U(u) > 0$ for all $u \in U$. Further, let $u^* \in U$ be the node that is removed, and let u_1, $u_2 \in U \setminus \{u^*\}$ be the isolated nodes in the resulting module, i.e., $\deg_{U\setminus\{u^*\}}(u_1) = \deg_{U\setminus\{u^*\}}(u_2) = 0$. As $\deg_U(u_1) > 0$ and $\deg_U(u_2) > 0$, there have to exist edges $\{u_1, u^*\}$ and $\{u_2, u^*\}$ with $w_{u_1u^*} > 0$ and $w_{u_2u^*} > 0$. This implies $\deg_U(u^*) \geq w_{u_1u^*} + w_{u_2u^*} > w_{u_1u^*} = \deg_U(u_1)$ and analogously $\deg_U(u^*) > \deg_U(u_2)$, which is a contradiction to $U \setminus \{u^*\}$ being the parent of U. □

This implies that we can refrain from extending modules that have two isolated nodes. By additionally activating the minimum degree filtering step explained in Section 5.5, we can make sure that any modules with isolated nodes are eliminated from the output. This strategy does not guarantee yet that the modules are indeed connected, but in conjunction with an adjustment of the minimum degree threshold it is often sufficient in practice. Also, more sophisticated pruning rules exploiting the density threshold and size constraints are conceivable.

5.7.3 Cardinality and Branching Restrictions

Sometimes it is possible to specify in advance a size range for the modules of interest. As our search strategy extends the modules by exactly one node in each step, it can naturally respect thresholds for the maximum number of nodes in a module. Minimum cardinality constraints, on the contrary, are a popular means to eliminate insignificant results. Being non-anti-monotonic, they do not explicitly contribute to speed up the search procedure. However, in the case of additional minimum (relative) degree constraints (Section 5.5), we can exploit minimum cardinality constraints to a priori remove low-degree nodes from consideration.

Due to the completeness of the dense module enumeration, the method might visit a large number of overlapping modules, in particular it will consider all dense submodules of a large solution module. A simple way to control the generation of similar modules are *maximum branching constraints*. While maximum cardinality thresholds constrain the depth of a search tree, branching criteria constrain the width of subtrees by restricting the maximum number of children per module to k. Consequently, the number of modules sharing the same ancestor is limited, and it decreases with increasing cardinality of the ancestor. Of course, this heuristic strategy leads to the loss of the completeness guarantee. That means, in contrast to the other constraints mentioned in this section, branching constraints do not allow to provide a declarative description of the result set; in particular, we cannot determine anymore exact, analytical p-values (Section 5.4.2). But if we select the most "promising" children in each step, the method is likely to find a substantial fraction of the most significant solutions. This idea is related to the concept of beam search [23]. The success of the approach depends on the selection of children. We propose the following procedure for that purpose: among all nodes v that produce children $U \cup \{v\}$ of the current module U, we choose the k nodes with the largest degree within $U \cup \{v\}$ (leading to child modules with the largest density). The motivation behind this is that they are most likely to have dense descendants. Among nodes with equal degree, we prefer those with the largest indices because according to our reduction scheme, they are the last to be removed. In other words, we use an ordering of candidate nodes that is reverse to the ordering defined for the reduction scheme (Definition 9).

6 Multi-Way Cluster Mining in Higher-Order Association Data

Now we generalize the dense module enumeration approach from the previous chapter to structured input data of higher order, i.e., each associative relationship involves n partners, where n can be greater than 2. In essential parts, this work is published in [69]; an extended article will appear in [68].

6.1 Motivation

So far, we have been considering undirected weighted graphs as input data, which can be represented by symmetric weight matrices. A dense module is a subset of nodes that corresponds to a symmetric submatrix with large average weight (ignoring diagonal entries). In Figure 6.1, we illustrate this basic setting and possible generalizations. As a first extension, we consider two-way weight matrices, where we have different entity sets in the two dimensions, like genes and experimental conditions from gene expression data, for example. Given a two-way matrix, we can look for subsets of rows that are associated with certain subsets of columns. Such submatrix patterns are called biclusters (or bi-sets) (see Section 4.5). There exist many approaches to detect biclusters, and various criteria have been used in the literature to assess the interestingness of bicluster patterns [151]. For instance, some methods consider the homogeneity of values within a bicluster [22]. In contrast, other approaches focus on the strength or density of the association [156, 205, 206, 232]. In analogy to our module finding approach described in Chapter 5, we use the average weight value within a bicluster to determine valid solutions. This approach to biclustering can also be seen as extracting dense subgraphs from a weighted bipartite graph (see Figure 6.1).

Taking this idea one step further, we look at higher-order input data, which can be represented as n-way weight arrays, also known as tensors. A higher-order or n-way cluster is then a subtensor described by specifying a non-empty subset of

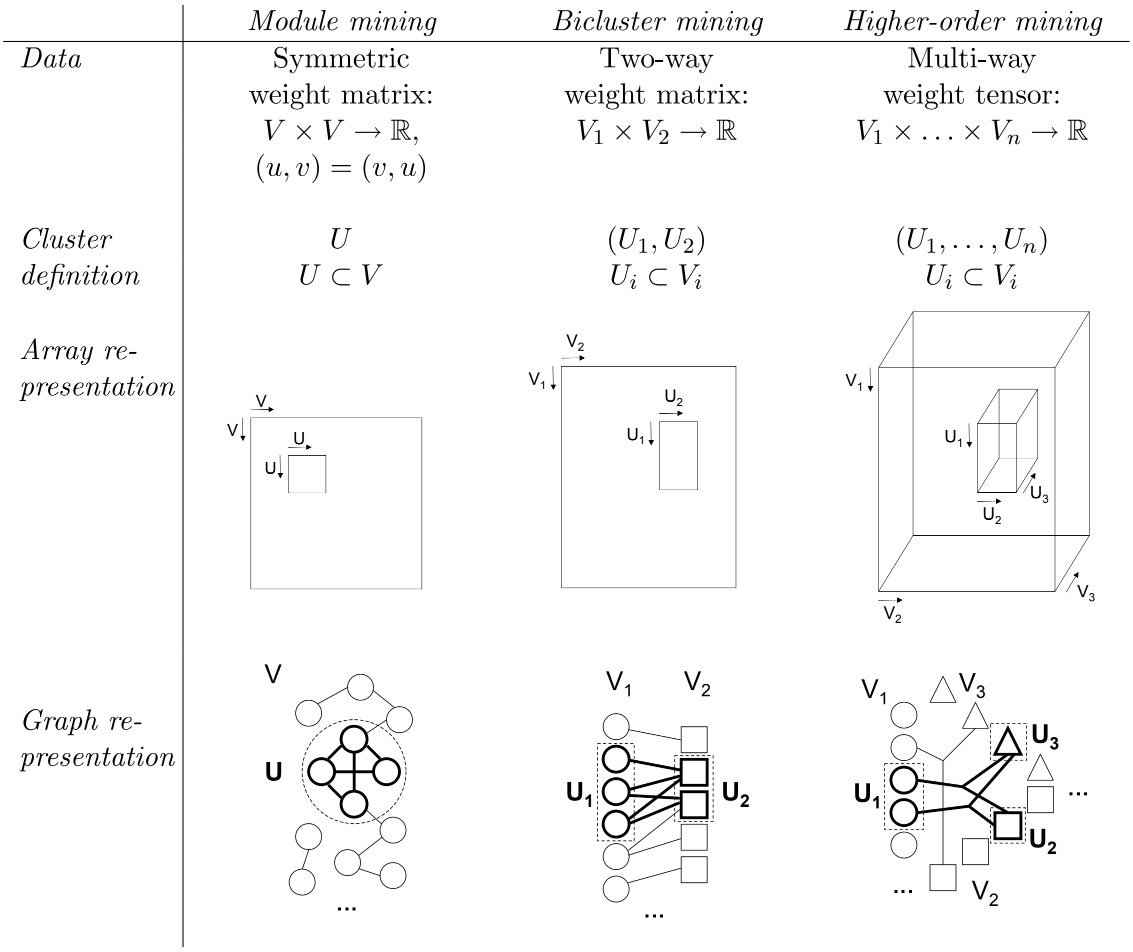


Figure 6.1: Generalization of module mining to two-way and higher-order data. For visualization purposes, the sets U and U_i are shown as coherent blocks; however, they can be arbitrary subsets.

indices in each dimension. From a graph-theoretic point of view, an n-way tensor corresponds to a weighted n-partite hypergraph, where each hyperedge connects n nodes, exactly one node from each partition. A popular approach to investigate such higher-order data is relational data mining (see Section 4.7). There, the focus is on binary-valued multi-way relationships, also called n-ary relations. They can be represented as an n-way tensor where an entry is 1 if the corresponding n-way relationship has been observed, and 0 otherwise. In that framework, relational mining is equivalent to extracting subtensors (clusters) that contain only 1-entries; different clusters may overlap. In the relational data mining terminology, these patterns are called n-sets, as a generalization of the concept of itemsets [4].

Our approach to higher-order cluster detection extends the definition of n-set to numerical data, that means, the tensor is not restricted to binary values, but may contain arbitrary weights. We consider a cluster or n-set pattern as a solution if the average value of the entries in the corresponding subtensor exceeds a given threshold; in particular, 0-entries are tolerated to a certain extent, whereas relational mining approaches require that all entries are 1 (see Figure 6.2). Our generalization can be advantageous in applications where data are sparse (i.e., it is likely that some observations are missing) or where observations have different weights (e.g., because their reliability or significance are subject to variation). Consequently, we can detect strong associations between sets of instances in noisy data; such higher-level patterns strengthen individual observations and can assist in making reliable predictions of missing values. There exist several previous methods for cluster discovery in multi-way weight tensors (see Section 4.7). Our approach differs from them in considering the association strength within a cluster instead of the homogeneity of weights. In fact, our average weight constraint can be seen as homogeneity criterion in the sense that most values in the cluster should be close to the maximum weight. However, we introduce an asymmetry in the problem, as blocks with homogeneous values close to zero are not considered as solutions (in analogy to relational n-set approaches). Also, note that our method detects overlapping clusters, while most previous techniques are based on partitioning.

Moreover, we treat cases where multi-way data contain inherent symmetries with respect to a subset of dimensions. For instance, multiple undirected networks, each represented by a symmetric weight matrix, can be stacked into a three-way tensor that is symmetric with respect to two dimensions. The coanalysis of multiple networks or graphs has received much attention, in particular in the context of computational biology and chemistry. While many approaches focus on the detection of frequently occurring subgraphs in large databases of small graphs [233], Example of an n-set for $n = 3$: $(\{a_1, a_2\}, \{b_1, b_2\}, \{c_1, c_2, c_3\})$

(a) Relational approach

Association	Value
(a_1, b_1, c_1)	1
(a_1, b_1, c_2)	1
(a_1, b_1, c_3)	1
(a_1, b_2, c_1)	1
(a_1, b_2, c_2)	1
(a_1, b_2, c_3)	1
(a_2, b_1, c_1)	1
(a_2, b_1, c_2)	1
(a_2, b_1, c_3)	1
(a_2, b_2, c_1)	1
(a_2, b_2, c_2)	1
(a_2, b_2, c_3)	1
average	$=1$

(b) Our approach

Association	Value
(a_1, b_1, c_1)	0.9
(a_1, b_1, c_2)	1
(a_1, b_1, c_3)	1
(a_1, b_2, c_1)	0.8
(a_1, b_2, c_2)	0.9
(a_1, b_2, c_3)	1
(a_2, b_1, c_1)	0.7
(a_2, b_1, c_2)	1
(a_2, b_1, c_3)	0.9
(a_2, b_2, c_1)	1
(a_2, b_2, c_2)	0
(a_2, b_2, c_3)	0.9
average	$\geq \theta$

Figure 6.2: Illustration of the n-set definition in the relational approach and in our dense cluster approach (here, $n = 3$). While the relational approach is based on binary values and requires that all associations within the n-set have value 1, we allow for arbitrary weights and require that the overall average across the n-set associations is larger than a threshold θ.

methods for larger networks often use density criteria combined with additional constraints [91, 104, 181, 236, 242]. Our approach extends the latter class of methods: it provides a dense pattern detection framework that allows to analyze tensor data with an arbitrary number of dimensions, can deal with binary and weighted values, and optionally includes symmetry constraints for one or several subsets of dimensions.

After formalizing the problem, we will present a multi-way extension of the dense module enumeration algorithm from the previous chapter.

6.2 Problem Definition

Our goal is to extract all dense clusters from a multi-dimensional data array (tensor). To formalize the problem, we first introduce some notation, generalizing the definitions from Section 5.2. Let $n > 0$ be the number of *dimensions* in the given data array (also called *ways* or *modes*). Then, we write the input in the following form:

$$W = (w_{k_1,\dots,k_n})_{k_i \in V_i,\ i=1,\dots,n} \tag{6.1}$$

The index k_i is used to access the i-th dimension and takes values from a finite index set $V_i = \{1, \dots I_i\}$, where I_i is a natural number that can differ from dimension to dimension. V_i is also called the *instance set* or *range* for the i-th dimension; the cardinality of the set is denoted by $|V_i|$ and equals I_i. The elements (entries) of W are real-valued weights indicating the association strength between the n instances. For convenience, we again normalize the array such that

$$w_{k_1,\dots,k_n} \leq 1 \quad \forall\, k_i \in V_i,\, i = 1, \dots, n\,. \tag{6.2}$$

An n-way *cluster* U is defined by specifying for each dimension a non-empty subset of the corresponding index set,

$$U = (U_1, \dots, U_n), \quad U_i \subset V_i,\, |U_i| \geq 1 \quad \forall i = 1, \dots, n\,. \tag{6.3}$$

The induced subarray is given by

$$W|_U = (w_{k_1,\dots,k_n})_{k_i \in U_i,\, i=1,\dots,n}. \tag{6.4}$$

Let us define the *cardinality* of a cluster as the sum of the cardinalities of the index subsets in all n dimensions, i.e., the total number of instances included in the cluster:

$$\text{card}(U) = \sum_{i=1}^{n} |U_i|. \tag{6.5}$$

This is not to be confused with the *cluster size*, which corresponds to the number of entries in the induced subarray,

$$\text{size}(U) = \prod_{i=1}^{n} |U_i|. \tag{6.6}$$

Our cluster definition implies that $\text{size}(U) \geq 1$. The *density* of the cluster U is defined as the average value of the weight entries in the induced subarray:

Definition 22 (Cluster Density)**.** *The density of an n-way cluster U with respect to the n-dimensional weight array W is given by*

$$\rho_W(U) = \frac{1}{\text{size}(U)} \sum_{k_i \in U_i} w_{k_1,\dots,k_n}\,. \tag{6.7}$$

Due to our normalization of the data array W, the largest possible cluster density is 1. Using the above definitions, we state the problem of dense cluster enumeration as follows.

Definition 23 (Dense Cluster Enumeration). *Given a weight-normalized n-dimensional data array W and a minimum density threshold θ with $0 < \theta \leq 1$, find all n-way clusters U such that $\rho_W(U) \geq \theta$.*

Note that different clusters are allowed to overlap. For $\theta = 1$, the problem is equivalent to n-set or hyperclique enumeration [31, 32, 100, 103].

6.3 Enumeration Approach

In order to solve the dense cluster enumeration problem, we again use a reverse search algorithm. In the following, we explain how we extend the module enumeration strategy from Chapter 5 to deal with this task. For that purpose, we first introduce an index mapping scheme that facilitates the algorithmic description; then, we specify the level-wise search space, establish a reduction scheme, and present the overall search procedure. Finally, we consider some implementation details and analyze the complexity of the method.

6.3.1 Global Index Representation

As defined in Equation (6.1), an n-way array is represented using dimension-specific index sets $V_1, \ldots, V_n$; each of them consists of successive indices starting from 1. Now we build a *global index* set across all dimensions:

$$\mathcal{V} = \{1, \ldots, \sum_{i=1}^{n} |V_i|\}\,. \tag{6.8}$$

The conversion of an element $v \in V_i$ to a global index is carried out according to the following scheme:

$$\mathcal{C}(v, i) = v + \sum_{j=1}^{i-1} |V_j| \tag{6.9}$$

For $i = 1$, the summation term is zero, i.e., $\mathcal{C}(v, 1) = v$. Accordingly, we determine the array dimension to which an element $v \in \mathcal{V}$ belongs as

$$\dim(v) = \max\{i = 1, \ldots, n : \sum_{j=1}^{i-1} |V_j| < v\}\,. \tag{6.10}$$

Then a cluster $U = (U_1, \dots, U_n)$ can also be represented as a subset of $\mathcal{V}$,

$$\mathcal{U} = \bigcup_{i=1}^{n} \bigcup_{u \in U_i} \{\mathcal{C}(u,i)\} . \tag{6.11}$$

Note that U and $\mathcal{U}$ are alternative representations of a uniquely determined cluster and can easily be transformed into each other. In the following, we will use the representation that is more convenient in the particular context.

6.3.2 Search Space

The search space for the dense cluster enumeration problem is the set of all possible n-way clusters. As in the module enumeration setting, it can be organized in the form of a lattice, i.e., in multiple levels. Here, the root level consists of all *trivial* clusters:

Definition 24 (Trivial Cluster)**.** *A cluster* $U = (U_1, \dots, U_n)$ *is called trivial if* $|U_i| = 1$ *for* $i = 1, \dots, n$*. Consequently,* $\mathrm{size}(U) = 1$ *and* $\mathrm{card}(U) = n$ *for each trivial cluster* U*.*

A trivial cluster corresponds to exactly one entry of the multi-dimensional array. By adding exactly one index to one particular set U_i, we obtain the clusters on the subsequent level of the search lattice. In this way, the clusters are iteratively expanded from level to level; at each level, the cluster index set $\mathcal{U}$ grows by one element and the cluster cardinality increases by 1. To traverse the search lattice in an efficient way, we define a search tree for each trivial cluster such that the resulting set of trees is a spanning forest of the search space. The next section describes a reduction scheme to construct search trees that allow for effective pruning based on the density criterion.

6.3.3 Reduction Scheme

The core component of a reverse search algorithm is the definition of a reduction scheme (i.e., the rule for parent construction) that guarantees anti-monotonicity (downward closure) and completeness of the search. For the reduction of dense multi-way clusters, we extend the parent definition for modules (Definition 9). First, we define the degree of an instance in a multi-dimensional array.

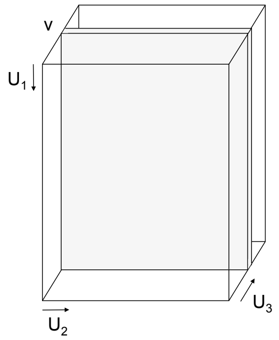


Figure 6.3: Visualization of a three-way cluster. In each reduction step, we remove one slice of the cluster, specified by a particular index element v.

Definition 25 (Degree)**.** *Given a cluster U, the degree of $v \in U_j$ with respect to U is defined as*

$$\deg_U(v, j) = \sum_{k_i \in U_i, i \neq j} w_{k_1,\ldots,k_{j-1},v,k_{j+1},\ldots,k_n} . \tag{6.12}$$

In the global index representation, there is no ambiguity for instances of different dimensions, so we simply write $\deg_{\mathcal{U}}(v)$ for $v \in \mathcal{U}$. Furthermore, W is not included as an explicit parameter of deg because our method deals with one given data array at a time.

With that, we specify the following reduction scheme.

Definition 26 (Reduction Scheme)**.** *Let $\mathcal{U}$ be a cluster. If v is the instance with the smallest index among the minimum degree elements in $\mathcal{U}$, the parent of $\mathcal{U}$ is given by $\mathcal{U} \setminus \{v\}$.*

Reducing the cluster $\mathcal{U}$ by one instance corresponds to removing a *slice* of the respective subarray, namely all entries involving the specified instance (see Figure 6.3). Here, we select an instance such that the sum of the entries in the corresponding slice (i.e., the degree) is minimal. It remains to show that this parent-child relationship satisfies the cluster reachability and anti-monotonicity requirements.

Lemma 8. *Let $\mathcal{U}$ be a non-trivial cluster and $\rho_W(\mathcal{U}) > 0$. Then, for all $v \in \mathcal{U}$ with minimum degree, i.e.,*

$$v \in \operatorname*{argmin}_{u \in \mathcal{U}} \deg_{\mathcal{U}}(u),$$

the following properties hold:

1. $\mathrm{size}(\mathcal{U} \setminus \{v\}) \geq 1$.

2. $\rho_W(\mathcal{U} \setminus \{v\}) \geq \rho_W(\mathcal{U})$.

Proof. We consider the case where the minimum degree instance belongs to the j-th dimension. For convenience, we use the subarray representation of clusters, $U = (U_1, \ldots, U_n)$, and denote by v the corresponding element in U_j. First, we show that $(U_1, \ldots, U_{j-1}, U_j \setminus \{v\}, U_{j+1}, \ldots, U_n)$ is a valid cluster where all index sets are non-empty. For that purpose, let us assume that $|U_j| = 1$. Then, $U_j = \{v\}$ and $\deg_U(v, j) = \sum_{k_i \in U_i} w_{k_1,\ldots,k_n} =: T$, that means the degree of v is equal to the sum of all elements in the subtensor induced by U. Furthermore, T is positive because $\rho_W(U) > 0$. As U is non-trivial, there exists a $j' \in \{1, \ldots, n\}$, $j' \neq j$, such that $|U_{j'}| > 1$. Let u' be an instance of minimum degree in $U_{j'}$, i.e., $u' \in \operatorname{argmin}_{u \in U_{j'}} \deg_U(u, j')$. Then, $T > \deg_U(u', j')$:

- For $\deg_U(u', j') > 0$: $T = \sum_{u \in U_{j'}} \deg_U(u, j') \geq |U_{j'}| \cdot \deg_U(u', j') > \deg_U(u', j')$
- For $\deg_U(u', j') \leq 0$: obvious because $T > 0$.

So we have found a cluster instance u' with $deg_U(u', j') < deg_U(v, j)$, which contradicts the assumption of the lemma. Consequently, $|U_j| > 1$ and therefore $|U_j \setminus \{v\}| > 0$. Thus, size$(U) \geq 1$.

The second part of the lemma is shown by simple algebra resembling the proof of Lemma 1:

$$\begin{aligned}
&\rho_W(U_1, \ldots, U_{j-1}, U_j \setminus \{v\}, U_{j+1}, \ldots, U_n) - \rho_W(U_1, \ldots, U_n) \\
&= \frac{\sum_{u \in U_j \setminus \{v\}} \deg_U(u, j)}{(|U_j| - 1) \cdot \prod_{i=1, i \neq j}^{n} |U_i|} - \frac{\sum_{u \in U_j} \deg_U(u, j)}{\prod_{i=1}^{n} |U_i|} \\
&= \frac{\frac{1}{|U_j|} \cdot \sum_{u \in U_j} \deg_U(u, j) - \deg_U(v, j)}{(|U_j| - 1) \cdot \prod_{i=1, i \neq j}^{n} |U_i|} \\
&\geq 0
\end{aligned}$$

The inequality holds due to the choice of v and the first part of the proof, which guarantees that the denominator is greater than zero. □

Algorithm 4 Pseudocode of DCE. W is the given n-dimensional data array with global index set $\mathcal{V}$ (and corresponding mapping $\mathcal{C}$), and θ denotes the minimum density threshold.

```
DCE ($\mathcal{V}$, $\mathcal{C}$, $W$, $\theta$) :
  for each $(k_1, \ldots, k_n)$ with $w_{k_1,\ldots,k_n} \geq \theta$ do
    DCE_Rec($\mathcal{V}, W, \theta, \bigcup_{i=1}^{n} \{\mathcal{C}(k_i, i)\}$)
  end for

DCE_Rec ($\mathcal{V}$, $W$, $\theta$, $\mathcal{U}$) :
  for each $v \in \mathcal{V} \setminus \mathcal{U}$ do
    if $\rho_W(\mathcal{U} \cup \{v\}) \geq \theta$ then
      if $\mathcal{U} \cup \{v\}$ is child of $\mathcal{U}$ then
        DCE_Rec ($\mathcal{V}, W, \theta, \mathcal{U} \cup \{v\}$)
      end if
    end if
  end for
  output $\mathcal{U}$
```

The first statement of the lemma refers to the cluster reachability; it ensures that, by iterative application of the reduction scheme in Definition 26, any cluster with positive density shrinks to a trivial cluster, i.e., a root of the search space; that means, there do not occur degenerate constructs where some dimensions-specific instance sets are empty. The second statement implies anti-monotonicity, i.e., a parent cluster is at least as dense as any child cluster.[1] Note that these properties hold for any minimum degree instance; however, to avoid duplicate investigation of subspaces, each cluster should have a unique parent, i.e., the reduction map has to specify which one of the minimum degree instances is selected (in our case, the instance with the smallest index). The defined reduction scheme directly suggests the following algorithm.

6.3.4 Search Algorithm

To enumerate all clusters in an n-dimensional data array W that satisfy a minimum density threshold θ, we perform the procedure shown in Algorithm 4, which is in the following referred to as DCE (dense cluster enumeration algorithm). The first step consists in finding all entries in the array that are greater than or equal to θ. These trivial clusters are then further expanded by a depth-first strategy producing descendants of increasing cardinality, in a similar way as in the module enumeration algorithm (Algorithm 1). That means, we generate in each step the

---

[1] However, the parent is not necessarily the densest subcluster of a given child cluster.

set of all possible candidates extending the current cluster by one instance and then select the actual children among those. According to the reduction scheme specified in Definition 26, we can characterize the children of a cluster as follows:

Definition 27 (Cluster Child)**.** *Let* $\mathcal{U}$ *be a cluster and* $v \in \mathcal{V} \setminus \mathcal{U}$*.* $\mathcal{U}^* = \mathcal{U} \cup \{v\}$ *is a child of* $\mathcal{U}$ *if and only if*

$$\forall u \in \mathcal{U} : \quad [\deg_{\mathcal{U}^*}(v) < \deg_{\mathcal{U}^*}(u)] \vee [(\deg_{\mathcal{U}^*}(v) = \deg_{\mathcal{U}^*}(u)) \wedge (v < u)] \,.$$

Only children that satisfy the density threshold are further investigated; otherwise, the current search tree is pruned. The correctness of the algorithm follows from Lemma 8 in the previous section. For efficiency reasons, the density condition is checked before the child condition; further implementation details are given in the next section.

6.3.5 Implementation Details

To be able to deal with an arbitrary number of dimensions, the input tensor is represented in a sparse format. For each non-zero entry, we create a data object that contains the n-dimensional index vector and the corresponding value. To facilitate the access to entries during the search, we generate for each $v \in \mathcal{V}$ a list of pointers to the objects containing v (also called *adjacency list* of v). In order to visit each data entry exactly once, it is sufficient to traverse the first $|V_1|$ adjacency lists because each entry has an index between 1 and $|V_1|$ in the first dimension. This procedure can be used for determining the root clusters. For efficient density checks, we employ the same strategy as in the module mining case (see Section 5.3.4), maintaining an array that contains for each element $v \in V$ its degree with respect to the current cluster $\mathcal{U}$, i.e., $d_{\mathcal{U}}(v) = \deg_{\mathcal{U}}(v)$ if $v \in \mathcal{U}$, and $d_{\mathcal{U}}(v) = \deg_{\mathcal{U} \cup \{v\}}(v)$ otherwise. To initialize this array for a trivial cluster $\mathcal{U}$, we fill in for each $v \in \mathcal{U}$ the value of the corresponding entry, whereas for each $v \in \mathcal{V} \setminus \mathcal{U}$, we fill in the value of the tensor element that shares with $\mathcal{U}$ all indices except v. After the addition of a new instance v to the current cluster, we traverse the adjacency list of v, filtering for entries containing at most one instance that is not member of $\mathcal{U} \cup \{v\}$, and updating the degree array accordingly. This requires at most $O(l_v n)$ operations, where l_v denotes the length of the adjacency list of v.

Again, this data structure for degree values allows for some shortcuts concerning the check of the child criterion. They are stated in the following lemmata.

Lemma 9. *Let $\mathcal{U}$ be a cluster in a non-negative tensor W, and let $d_{\mathcal{U}}$ be the degree array with respect to $\mathcal{U}$. Further, let u^* the previously added instance, and $v \in \mathcal{V} \backslash \mathcal{U}$. Then the following rule holds:*

$$[d_{\mathcal{U}}(v) < d_{\mathcal{U}}(u^*)] \vee [d_{\mathcal{U}}(v) = d_{\mathcal{U}}(u^*) \wedge v < u^*] \implies U \cup \{v\} \textit{ is a child of } U$$

Proof. Analogous to Lemma 2 in Section 5.3.4. □

For the next lemma, we need the notion of v-*slice*; it simply refers to the set of entries in the cluster subarray that include the instance v (see Figure 6.3).

Lemma 10. *Let W be a weight-normalized tensor such that the maximum entry is 1. Further, let $\mathcal{U}$ be a cluster with degree array $d_{\mathcal{U}}$, and let u^* be its most recently added instance. For $v \in \mathcal{V} \setminus \mathcal{U}$, we denote by $g_{\mathcal{U}}(u^*, v)$ the number of elements that the u^*-slice of the cluster $\mathcal{U}$ gains by the addition of v. Then the following rule holds:*

$$\begin{gathered}[d_{\mathcal{U}}(v) > d_{\mathcal{U}}(u^*) + g_{\mathcal{U}}(u^*, v)] \vee [d_{\mathcal{U}}(v) = d_{\mathcal{U}}(u^*) + g_{\mathcal{U}}(u^*, v) \wedge v > u^*] \\ \implies \mathcal{U} \cup \{v\} \textit{ is not a child of } \mathcal{U}\end{gathered}$$

Proof. The quantity $g_{\mathcal{U}}(u^*, v)$ is equivalent to the number of entries in the intersection of the u^*-slice and the v-slice of the cluster $\mathcal{U} \cup \{v\}$, which is easily computed as follows:

$$g_{\mathcal{U}}(u^*, v) = \begin{cases} 0 & \text{if } \dim(v) = \dim(u^*) \\ \frac{\text{size}(U)}{|U_{\dim(v)}| \cdot |U_{\dim(u^*)}|} & \text{otherwise}, \end{cases}$$

where $U = (U_1, \ldots, U_n)$ is the subarray representation corresponding to $\mathcal{U}$ and $\dim(v)$ is the dimension to which v belongs. Due to the normalization of W, $g_{\mathcal{U}}(u^*, v)$ corresponds to the maximum increase of the degree of u^* after addition of v. With that, the rule follows analogously to Lemma 2 in Section 5.3.4. □

If neither of these rules applies, we traverse the adjacency list of v to determine the values $\deg_{\mathcal{U} \cup \{v\}}(u)$ for $u \in \mathcal{U}$, which are needed to check the child conditions given in Definition 27.

6.3.6 Complexity

Like the network module enumeration procedure described in Section 5.3, DCE is a polynomial-delay algorithm. To see this, let us first consider a single iteration of the subroutine DCE_Rec (see Algorithm 4), which corresponds to finding the children of a given cluster. Using the implementation outlined above, this needs $O(|\mathcal{V}| \cdot (l \cdot n + |\mathcal{U}|))$ operations, where $\mathcal{V}$ is the global index set, l is the average length of an adjacency list, n is the number of dimensions, and $\mathcal{U}$ is the current cluster. In the worst case, we go for each $v \in \mathcal{V} \setminus \mathcal{U}$ through the whole adjacency list to determine the updated degree values for the members of $\mathcal{U}$, and then check all conditions in Definition 27. In practice, the density check will already discard many candidates; furthermore, we can avoid the traversal of the adjacency list in the cases where Lemma 9 or Lemma 10 can be applied. Finally, when going through an adjacency list, only objects that are relevant for the update have to be fully processed. For a complete input tensor $V_1 \times \ldots \times V_n$, the adjacency list for an instance in dimension j has a length of $(\prod_{i=1,\, i\neq j}^{n} |V_i|)$, i.e., roughly $O(|\mathcal{V}|^{n-1})$; in sparse settings, it can be considerably shorter. Assuming $|\mathcal{U}| \ll |\mathcal{V}|$, the traversal of the adjacency list will dominate the $|\mathcal{U}|$ term, so the complexity of one iteration can be expressed as $O(|\mathcal{V}| \cdot l \cdot n)$, i.e., it is linear in the input size, which corresponds to the adjacency list representation of the data array W.

Using data representations that allow for constant-time access to specific entries of W, the complexity of one iteration is given by

$$O\Big(\sum_{i=1}^{n} |V_i \setminus U_i| \prod_{j=1,\, j\neq i}^{n} |U_j|\Big) = O\Big(\prod_{i=1}^{n} |V_i|\Big),$$

which reflects the costs of traversing the entries of each slice that can be added to the cluster; here, V_i and U_i denote dimension-specific instance sets as introduced in Section 6.2. That means, also in this representation the complexity is linear in the size of the input tensor.

As in Section 5.3.5, we can apply the odd-even method for outputs in recursive calls. Then, the delay between two consecutive solutions within the same search tree has the same complexity as one iteration of DCE_Rec. Furthermore, the traversal of irrelevant entries between two successful search trees (i.e., the time between two calls of DCE_Rec from the routine DCE in Algorithm 4) is bounded by the size of the input tensor. Thus, we have the following theorem.

Theorem 2. *The dense cluster enumeration problem for a given n-way tensor can* *be solved by a reverse search algorithm that has linear delay with respect to the input size.*

Empirical results on the runtime behavior of the algorithm will be shown in Section 6.6. Again, the computation can be parallelized because different search trees as well as different branches of the same search tree can be explored independently of each other. The memory requirements of the algorithm mainly consist in the storage space needed for the input; as discussed above, either sparse or full data representations are conceivable. In addition to that, the implementation described in the previous section uses $O(|\mathcal{V}|)$ space for each recursive step; the maximum recursion depth is equal to $|\mathcal{U}_{\max}| - n + 1$, where $\mathcal{U}_{\max}$ is the solution cluster with the largest cardinality.

6.4 Extensions

To facilitate a carefully directed cluster analysis, we can employ similar techniques as proposed for module enumeration in networks (Chapter 5). These include output filtering steps as well as pruning strategies regarding additional criteria.

6.4.1 Output Filtering and Balance Criteria

As in the network module approach, we can define a *local maximality criterion* for clusters, which can be checked during the enumeration procedure without additional costs:

Definition 28 (Locally Maximal Dense Cluster)**.** *A dense cluster $\mathcal{U}$ is called locally maximal if for all $v \in \mathcal{V} \setminus \mathcal{U}$, $\mathcal{U} \cup \{v\}$ is not dense.*

Also, we can take again minimum degree thresholds into account in order to filter the results. For that, we can either use absolute or relative degree values; the relative degree value for an instance is equivalent to the average of the entries in the corresponding slice of the cluster (i.e., the density of the slice).

Definition 29 (Minimum Relative Degree Threshold)**.** *A cluster $U = (U_1, \ldots, U_n)$ satisfies the minimum relative degree threshold γ if $\deg_U(u)/(\mathrm{size}(U)/|U_{\dim(u)}|) \geq \gamma$ for all $u \in \mathcal{U}$.*

Clusters that satisfy the minimum relative degree threshold are also called *balanced clusters* later on. Beyond that, we can define balance constraints at finer granularity levels: if we consider fix indices not only for one, but for several dimensions, we can also check the density of lower-order slices or fibers of the cluster. Clearly, clusters that exceed a threshold γ for a density balance criterion also have a density greater than γ, whereas the other direction is not true. Different techniques to search for clusters with balance constraints and to combine these requirements with other criteria are discussed in Section 5.5; the strategies described for the reverse search approach of module enumeration are directly transferable to the higher-order setting.

6.4.2 Cluster Ranking

In analogy to the exact p-value criterion for module ranking introduced in Section 5.4.2, we compute for each solution cluster $U = (U_1, \ldots, U_n)$ the exact probability of obtaining by chance a cluster with at least the same density from the input tensor $W = (w_{k_1,\ldots,k_n})_{k_i \in V_i \; \forall i=1,\ldots,n}$:

$$\frac{|\{U' = (U'_1, \ldots, U'_n) : \rho_W(U') \geq \rho_W(U) \wedge \forall i \; U'_i \subset V_i \wedge |U'_i| = |U_i| \,\}|}{\prod_{i=1}^{n} \binom{|V_i|}{|U_i|}} \tag{6.13}$$

To determine the count in the numerator, the enumeration algorithm optionally stores density and size characteristics for all solutions. This probability value yields a simple, but well-defined criterion to rank the results. Moreover, it is conceivable to generalize the alternative ranking criteria mentioned in Section 5.4.2.

6.4.3 Isolation-Based Pruning

A cluster instance that has a degree of zero is called an isolated instance. By default, we only keep clusters without any isolated instance in the solution set. While this can be achieved by output filtering, we can exploit the non-isolation criterion already during the search for pruning. The following lemma generalizes the result from Lemma 7 in Section 5.7.2.

Lemma 11. *Let W be a tensor with non-negative entries. If a cluster contains only non-isolated instances, its ancestors have at most one isolated instance per dimension.*

Proof. Let $\mathcal{U}$ be a non-trivial cluster that contains only non-isolated instances. Its ancestors are obtained by iterative reduction until a trivial cluster is reached. First, we show that a single reduction step cannot turn $\mathcal{U}$ into a cluster with (at least) two isolated instances in a dimension. Assume that this would be possible. Let $u^* \in \mathcal{U}$ denote the instance that is removed in the reduction step; further, let $u_1, u_2 \in \mathcal{U} \setminus \{u^*\}$ be the isolated instances in the resulting cluster, i.e., $\deg_{\mathcal{U}\setminus\{u^*\}}(u_1) = \deg_{\mathcal{U}\setminus\{u^*\}}(u_2) = 0$. The instances u_1 and u_2 belong to the same dimension, u^* belongs to a different dimension. Together with the non-negativity assumption for tensor entries, it follows that $\deg_{\mathcal{U}}(u^*) \geq \deg_{\mathcal{U}}(u_1) + \deg_{\mathcal{U}}(u_2)$. By the definition of $\mathcal{U}$, $\deg_{\mathcal{U}}(u_1) > 0$ and $\deg_{\mathcal{U}}(u_1) > 0$. Thus, $\deg_{\mathcal{U}}(u^*) > \deg_{\mathcal{U}}(u_1)$ and $\deg_{\mathcal{U}}(u^*) > \deg_{\mathcal{U}}(u_2)$, contradicting the fact that u^* is a minimum degree node. Second, following the argumentation in the proof of Lemma 7, isolated instances cannot be accumulated during multiple reduction steps. $\square$

That means, a search tree can be pruned if the current cluster contains two isolated instances u_1 and u_2, $u_1 \neq u_2$, with $\dim(u_1) = \dim(u_2)$. Similarly, one can show that clusters with one zero-degree instance in each dimension cannot have a descendant that contains only non-isolated instances. In practice, accumulation of isolated instances can only occur for large clusters or low density thresholds; therefore it would also be possible to define heuristic rules that stop such degenerated extensions of clusters, at a very low risk of losing interesting solutions; in the worst case, the algorithm would then fail to find some very loosely connected clusters, but it still would yield their dense subclusters.

6.4.4 Other Restrictions

As in the case of module finding in networks, we can restrict the cluster cardinality and size by predefining minimum or maximum thresholds for the number of instances in individual dimensions. Also, it is possible to include constraints from external data sources for some instance sets, as described in Section 5.7.1. Furthermore, if there exists a natural order of instances in certain dimensions of the input tensor (like consecutive time intervals), it might be useful for some applications to consider only clusters with neighboring instances in specific dimensions, rather than exploring the full combinatorial space. It is not trivial to exploit this criterion for pruning the density-based reverse search tree because it is not anti-monotonic; i.e., the ancestors of a cluster with consecutive instances might have gaps in the sequence of instances. However, several approaches are possible if cardinality constraints have been specified with respect to that dimension. On the one hand, one can prune the search as soon as the index range (that is, the difference between the largest and the smallest occurring index in the cluster) gets too large. On the other hand, one can consider windows of a fixed size, and perform for each of them separately a reverse search with respect to the other dimensions. Note that, by aggregating over the values in the window, we can eliminate the corresponding dimension from the input tensor and obtain a standard dense cluster enumeration problem for the lower-dimensional tensor. Beyond these exact constraints, we can again heuristically control the complexity of the search; for instance, branching constraints reduce the generation of overlapping clusters; as in Section 5.7.3, we use the degree values to pick the k most promising instances for extension. Further possibilities to restrict the search include the selection of starting entries (i.e., roots of search trees) and the explicit control of instance reusage in different clusters. Finally, symmetry structure in the data can impose specific requirements on the cluster analysis, which are explained in detail in the next section.

6.5 Symmetry Adaptations

So far, we considered multi-way data with distinct instance sets in each dimension. Here, we discuss how to deal with partial symmetries in the input data and include cluster symmetry constraints. This extension of the dense cluster enumeration formalism will restore the network module mining task from Chapter 5 as a special case. As a motivating example for symmetries in the multi-way setting, let us consider a set of weighted undirected networks that share the same set of nodes.

6.5.1 Motivation

In many systems biology studies, there are multiple networks available that represent different kinds of relationships between the genes or proteins of a certain species. Let us focus here on undirected relationships. To jointly analyze these networks for dense patterns, they can be stacked into a three-way tensor where an entry w_{ijk} corresponds to the weight of the edge between the i-th and the j-th node in the k-th network. This tensor representation has the following characteristics: a) the first two dimensions contain identical instance sets; b) as the networks are undirected, the entries w_{ijk} and w_{jik} are equivalent; we say that the tensor is *symmetric* with respect to the first two dimensions; c) “diagonal” entries w_{iik} correspond to self-edges (loops). Now we are interested in finding subsets of nodes that induce dense subgraphs in a subset of networks. We can tackle this problem in the tensor framework by extracting dense three-way clusters that have identical instance sets in the first two dimensions; i.e., the clusters will be symmetric with respect to these dimensions, and, in analogy to Chapter 5, we will ignore self-edges for the density criterion. In the following section, we introduce definitions for multi-way cluster enumeration that respect these symmetry requirements.

6.5.2 Definitions

Our dense cluster enumeration framework is suitable to handle arbitrary symmetry relationships among dimensions; there may exist multiple symmetry groups of different size. For instance, a six-way tensor could be symmetric with respect to dimensions 1 and 2, and also symmetric with respect to dimensions 3, 4, and 6. To keep the notation simple, we illustrate the main concepts for the case where we have symmetry with respect to the first j dimensions ($j \leq n$) and all other dimensions are not involved in symmetry relationships. That means, given a tensor entry $w_{k_1,\ldots,k_n}$ with distinct indices $k_1, \ldots, k_j$, all entries that can be obtained by permutation of the first j indices are equivalent, so it is sufficient to store only one of the $j!$ possibilities. Then, a cluster $U = (U_1, \ldots, U_n)$ is called symmetric (with respect to the first j dimensions) if $U_1 = \ldots = U_j$. Its size is given by

$$\mathrm{size}_j(U) = \binom{|U_1|}{j} \prod_{i=j+1}^{n} |U_i| \,, \tag{6.14}$$

and its density is calculated as follows:

$$\rho_{W,j}(U) = \frac{1}{\mathrm{size}_j(U)} \sum_{k_i \in U_i, k_1 < \ldots < k_j} w_{k_1,\ldots,k_n} \tag{6.15}$$

Like in Section 5.2, we count equivalent entries only once and we do not take self-relationships into account (i.e., we only consider entries with distinct indices $k_1, \ldots, k_j$). This leads us to a modified definition for the degree of an instance $v \in U_l$:

$$\deg_U(v,l) = \begin{cases} \displaystyle\sum_{k_i \in U_i, k_l = v, k_1 < \ldots < k_j} w_{k_1,\ldots,k_n} & \text{if } l > j \\ \displaystyle\sum_{k_i \in U_i, v \in \{k_1,\ldots,k_j\}, k_1 < \ldots < k_j} w_{k_1,\ldots,k_n} & \text{if } l \leq j \end{cases} \tag{6.16}$$

In analogy to the setting without symmetry, we represent clusters as subsets of a global index set $\mathcal{V}$ (see Section 6.3.1). However, identical instances from the different dimensions involved in a symmetry relation correspond to one global index, i.e., an element $v \in \mathcal{V}$ may belong to several dimensions. For convenience, we define $\dim(v)$ to be the first dimension among them (respecting the order of dimensions in the input array); in our example case, assuming symmetry for the dimensions 1 to j, we set $\dim(v) = 1$ for the corresponding instances.

6.5.3 Reduction Scheme

With the new definition of degree, reduction is again performed by removing the minimum degree instance that has the smallest global index. In analogy to Lemma 8, it can be shown that this reduction map is valid, i.e., it yields anti-montonicity and cluster reachability:

Lemma 12. *Let $\mathcal{U}$ be a non-trivial cluster with $\rho_W(\mathcal{U}) > 0$ that is symmetric with respect to dimensions 1 to j. Then, for all $v \in \mathcal{U}$ with minimum degree, i.e.,*

$$v \in \operatorname*{argmin}_{u \in \mathcal{U}} \deg_{\mathcal{U}}(u)\,,$$

the following properties hold:

1. $\operatorname{size}(\mathcal{U} \setminus \{v\}) \geq 1$.
2. $\rho(\mathcal{U} \setminus \{v\}) \geq \rho(\mathcal{U})$.

Proof. Let U be the subtensor representation of $\mathcal{U}$. Because of the symmetry assumption we know that $U_1 = \ldots = U_j$. $\mathcal{U}$ is non-trivial, i.e., $\operatorname{size}(\mathcal{U}) > 1$. Now let us assume that $\operatorname{size}(\mathcal{U} \setminus \{v\}) < 1$. This happens in the following cases:

1. $\dim(v) = 1$ and $|U_1| = j$
2. $\dim(v) = i$ and $|U_i| = 1$ for an $i > j$

In either case, $\deg_U(v) = \sum_{k_i \in U_i, k_1 < \ldots < k_j} w_{k_1,\ldots,k_n} =: T$, that means each cluster element contains v. Furthermore, T is positive because $\rho_W(U) > 0$. Due to the non-triviality of U, there has to exist a j' such that either $j' = 1$ and $|U_{j'}| > j$ (case A), or $j' > j$ and $U_{j'} > 1$ (case B). Let u' be an instance of minimum degree in $U_{j'}$. Then, $T > \deg_U(u', j')$. For case A, this follows directly from the proof of Lemma 8. For case B, we have

$$T = \frac{1}{j} \sum_{u \in U_1} \deg_U(u, 1) \geq \frac{1}{j} |U_1| \cdot \deg_U(u', 1) > \deg_U(u', 1)\,,$$

provided that $\deg_U(u', 1) > 0$ (the case $\deg_U(u', 1) \leq 0$ is obvious). This leads to a contradiction, so the first statement of the lemma is true.

For the second part, we again distinguish two cases. If $\dim(v) = i$ and $i > j$, the statement follows immediately from Lemma 8. Otherwise, i.e., if $\dim(v) = 1$, the proof is also straightforward:

$$\begin{aligned}
&\rho_W(U_1 \setminus \{v\}, \dots, U_j \setminus \{v\}, U_{j+1}, \dots, U_n) - \rho_W(U_1, \dots, U_n) \\
&= \frac{\left(\frac{1}{j} \sum_{u \in U_1} \deg_U(u, 1)\right) - \deg_U(v, 1)}{\binom{|U_1| - 1}{j} \prod_{i=j+1}^{n} |U_i|} - \frac{\frac{1}{j} \sum_{u \in U_1} \deg_U(u, 1)}{\binom{|U_1|}{j} \prod_{i=j+1}^{n} |U_i|} \\
&= \frac{\frac{1}{|U_1|} \sum_{u \in U_1} \deg_U(u, 1) - \deg_U(v, 1)}{\binom{|U_1| - 1}{j} \prod_{i=j+1}^{n} |U_i|} \\
&\geq 0
\end{aligned}$$

□

6.5.4 Details

To perform the dense cluster enumeration, we can use similar data structures and speed-up rules as described in Section 6.3.5. However, as the instances may now belong to multiple dimensions, the initialization and update procedures have to be adapted. In particular, the first $|V_1|$ adjacency lists may contain duplicate entries, and several tensor entries can be relevant for the initialization of one single entry in the degree array. Moreover, we need to adjust the computation of $g_U(u^*, v)$, which is the number of new entries in the u^*-slice of the cluster U after adding v, where $u^*, v \in \mathcal{V}$, $u^* \neq v$. Here, we consider the general case, allowing for an arbitrary number of symmetry groups among the n dimensions. Recall that $\dim(v)$ yields in case of symmetry groups just one representative dimension. Further, let s_i denote the total number of dimensions belonging to the same symmetry group as the i-th dimension, i.e., $s_i > 1$ if the i-th dimension has a symmetry relationship with respect to other dimensions, and $s_i = 1$ otherwise. Finally, $\text{size}(U)$ is the number of distinct cluster entries before adding v (i.e., entries that are equivalent due to symmetry are counted only once). Then,

$$g_U(u^*,v) = r_U(u^*,v) \cdot \mathrm{size}(U), \text{ where } r_U(u^*,v) =$$
$$\begin{cases} \frac{1}{|U_{\mathrm{dim}(v)}|\cdot|U_{\mathrm{dim}(u^*)}|} & \text{if } \mathrm{dim}(v) \neq \mathrm{dim}(u^*) \wedge s_{\mathrm{dim}(v)} = 1 \wedge s_{\mathrm{dim}(u^*)} = 1 \\ \frac{s_{\mathrm{dim}(u^*)}}{|U_{\mathrm{dim}(v)}|\cdot|U_{\mathrm{dim}(u^*)}|} & \text{if } \mathrm{dim}(v) \neq \mathrm{dim}(u^*) \wedge s_{\mathrm{dim}(v)} = 1 \wedge s_{\mathrm{dim}(u^*)} > 1 \\ \frac{s_{\mathrm{dim}(v)}}{(|U_{\mathrm{dim}(v)}|-s_{\mathrm{dim}(v)}+1)|U_{\mathrm{dim}(u^*)}|} & \text{if } \mathrm{dim}(v) \neq \mathrm{dim}(u^*) \wedge s_{\mathrm{dim}(v)} > 1 \wedge s_{\mathrm{dim}(u^*)} = 1 \\ \frac{s_{\mathrm{dim}(v)} s_{\mathrm{dim}(u^*)}}{(|U_{\mathrm{dim}(v)}|-s_{\mathrm{dim}(v)}+1)|U_{\mathrm{dim}(u^*)}|} & \text{if } \mathrm{dim}(v) \neq \mathrm{dim}(u^*) \wedge s_{\mathrm{dim}(v)} > 1 \wedge s_{\mathrm{dim}(u^*)} > 1 \\ \frac{s_{\mathrm{dim}(v)}(s_{\mathrm{dim}(v)}-1)}{|U_{\mathrm{dim}(v)}|(|U_{\mathrm{dim}(v)}|-s_{\mathrm{dim}(v)}+1)} & \text{if } \mathrm{dim}(v) = \mathrm{dim}(u^*) \wedge s_{\mathrm{dim}(v)} > 1 \\ 0 & \text{otherwise} \end{cases}$$

These equations follow directly from the definition of $g_U(u^*,v)$, taking into account that two of the n dimensions are fixed (to u^* and v) and $|U_{\mathrm{dim}(v)}|$ is increased by 1. The extensions described in Section 6.4 are, with small modifications, also applicable in the case of symmetry constraints; the difference is that dimensions involved in symmetry relationships cannot be treated separately anymore, but have to be considered simultaneously. This affects for instance the computation of the ranking criterion and the isolation-based pruning rules.

In this section, we discussed how to extend our enumeration method to extract partially symmetric clusters from partially symmetric data. Apart from that, other application scenarios are conceivable, which can be solved similarly. For example, searching for asymmetric clusters in symmetric tensors can be meaningful if there exist groups of instances with strong inter-group connections, but not necessarily strong inner-group connections. On the other hand, one might be interested in clusters that contain the same set of instances in several dimensions although the data are not symmetric.

6.6 Experimental Studies

To investigate the feasibility of dense cluster enumeration, we implemented our DCE method (Algorithm 4) in C++[2] and performed experiments on synthetic and real-world datasets. The implementation uses the sparse representation via adjacency lists described in Section 6.3.5, which allows to use the same code for input tensors with an arbitrary number of dimensions. By default, we report locally maximal

---

[2] The implementation is available at `http://www.kyb.tuebingen.mpg.de/~georgii/dce.html`.

clusters that do not contain isolated instances (see Section 6.4.1 and Section 6.4.3). In the following, we describe empirical analyses regarding efficiency, scalability, and retrieval performance properties. For biological results, we refer to Chapters 9 and 10.

6.6.1 Scalability

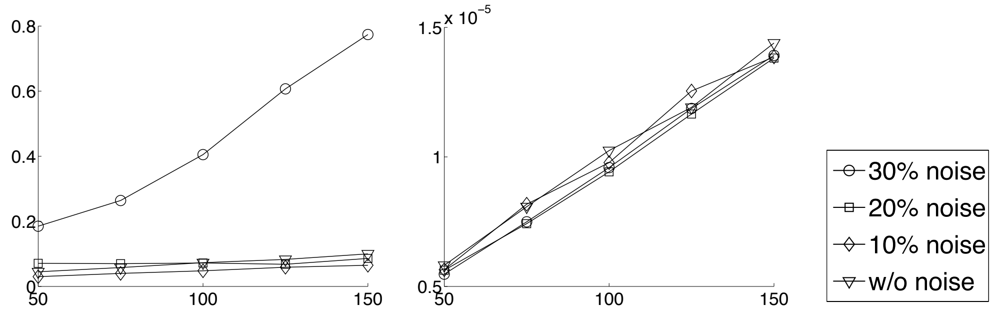
First, we tested the runtime performance of the DCE method on artificial datasets. For that purpose, we generated sparse tensors with hidden clusters. For simplicity, we used binary values, i.e., each tensor entry is either 0 or 1. Let n be the number of dimensions, and m the number of clusters. Furthermore, we assumed a hypercubic model where each dimension has the same index set size d and each hidden cluster contains exactly s instances in each dimension. The clusters were allowed to overlap without any restriction. In addition, we imposed different levels of noise onto the data. Here, the noise corresponds to random 0-1 flips. Initially, all tensor entries within clusters were set to 1. Given a noise level p, we randomly selected $p\%$ of all 1-entries and the same number of 0-entries. Then the selected elements were flipped, i.e., the 1-entries were set to 0 and vice versa.

Given this model for data generation, we investigated the scalability of DCE with respect to the different model parameters d, n, m, and s. Our basic setting was $d = 100$, $n = 3$, $m = 20$, and $s = 3$; this amounts to a total of 540 non-zero associations (1-entries) among 300 different instances (from three dimensions). Starting from that, we did four series of experiments, varying one of the parameters while keeping the others fixed. The maximum number of 1-entries was 540, 810, 4860, and 2500, respectively. For each parameter configuration, we generated ten random datasets and considered noise levels from 0% to 30%. The density threshold for the DCE algorithm was chosen in dependence of the noise level, $\theta = (100 - p)\%$. Figure 6.4 shows the resulting DCE runtime curves for each parameter. In addition to the total runtime, we report the delay, i.e., the runtime per solution. The values are averages across the ten random datasets. All measurements were performed on a 3.0 GHz machine.

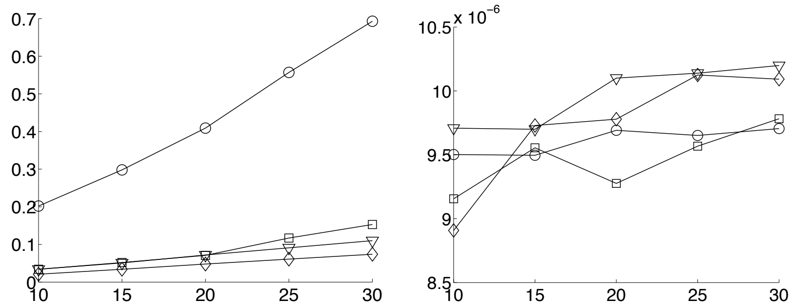
DCE scales favorably with d, the number of instances per dimension (Figure 6.4 (a)). For noise levels from 0% to 20%, the runtime remains approximately constant with increasing d, for 30% noise it increases linearly. In the 30% noise case, the noise elements cover more instances and, due to the lowered density threshold, there exist more cluster solutions; consequently, the curve depends much stronger on d than the curves for lower noise levels. The delay shows a linear increase in

Total time (s) Delay (s)

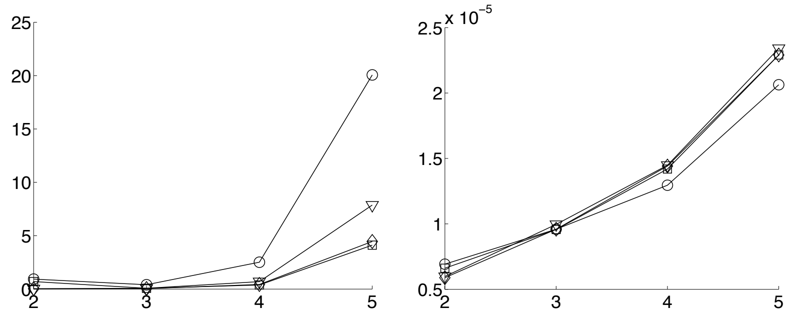
(a) Array cardinality d (per dimension)



(b) No. of clusters m



(c) No. of dimensions n



(d) Cluster cardinality s (per dimension)

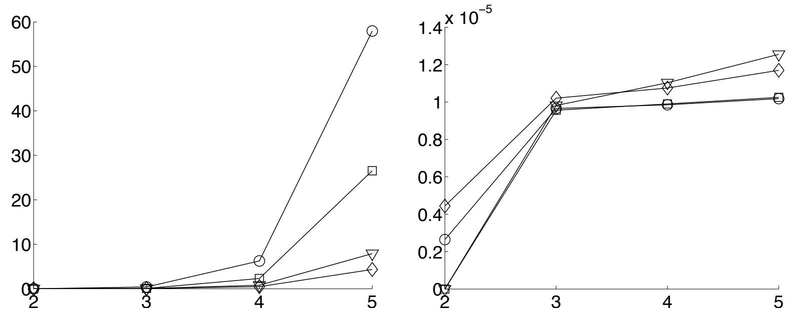


Figure 6.4: DCE runtime measurements for artifical data in dependence of different parameters. For each experiment, we report the total runtime and the delay. The x-axis contains the values of the varied parameter, the y-axis shows the time in seconds.

all cases. Regarding the number of hidden clusters, the total runtime increases linearly at all noise levels (Figure 6.4 (b)). The delay increases only very slightly; in fact, the higher number of clusters makes cluster overlaps (in terms of instances) more likely, so some instances may have longer adjacency lists, which can lead to an increased computational effort per solution. Actually, one of the main reasons for the 0% curve being on top is that the cluster instances have longer adjacency lists than in the noisy cases. The number of subcluster solutions per hidden cluster increases exponentially with the number of dimensions n, which is reflected by the total runtime (Figure 6.4 (c)). In contrast, the increase of the delay is much more moderate. Likewise, when increasing the number of instances per dimension in the hidden clusters, the delay grows very slowly, although the total runtime of DCE increases significantly (Figure 6.4 (d)). Again, this can be explained by the increased number of subcluster solutions. Also, the effect is much stronger for higher noise levels.

To conclude, DCE is appropriate for finding dense clusters in sparse settings; however, the number of solutions may grow considerably with the dimensionality and the cardinality of the clusters. Remarkably, the computational effort per solution scales very well in the latter case, even though we use the adjacency list representation of the tensor and therefore do not have constant-time access to elements. This indicates that our methods to speed up the search process are effective. Furthermore, it encourages to combine the DCE search with heuristics that restrict the generation of overlapping clusters while trying to catch the most significant ones, which is investigated in the next section.

6.6.2 Performance of Branching-Restricted Search

In the following experiments, we used the search strategy proposed in Section 6.4.4. The idea is to control the number of branches descending from a cluster. That means, in each iteration of the algorithm, we select the k most promising children (if available). Obviously, this restriction leads to a loss of the completeness property. We analyzed the behavior of DCE for different values of k in the context of our artificial datasets. Table 6.1 shows our results for varying dimension-wise cluster cardinality s at noise levels 0% and 30%. We used the same datasets as for Figure 6.4 (d) and compared k-values from 1 to 4 with the unrestricted (complete) DCE version. As can be seen, the overall runtime was drastically reduced by introducing the branching restriction, in particular in the high noise case.

In order to evaluate the quality of the results, we used the following precision and recall measures. The precision is given by the fraction of hidden cluster entries among all DCE cluster entries, and the recall is given by the fraction of DCE cluster entries among all hidden cluster entries. Formally, let D be the set of all tensor entries that belong to clusters detected by DCE (excluding single-entry clusters), and let H be the set of all tensor entries that belong to hidden clusters. Then, recall and precision are defined as follows:

$$\text{Recall} = \frac{|D \cap H|}{|H|}$$
$$\text{Precision} = \frac{|D \cap H|}{|D|}$$

The averages across the r random datasets (here, $r = 10$) were determined as

$$\text{Recall}_{\text{avg}} = \frac{\sum_{i=1}^{r} |D_i \cap H_i|}{\sum_{i=1}^{r} |H_i|}$$
$$\text{Precision}_{\text{avg}} = \frac{\sum_{i=1}^{r} |D_i \cap H_i|}{\sum_{i=1}^{r} |D_i|},$$

where D_i and H_i denote the entry sets of the detected and the hidden clusters for the i-th dataset, respectively. Note that in our experiments, all hidden clusters have the same number of entries ($|H_i| = s^n$). Since we know the size of our hidden clusters, we also evaluated recall and precision based on the predicted clusters that have at least this size (i.e., at least s instances in each dimension); these clusters are called *size-restricted*. In addition, we report recall and precision of the results satisfying the balance criterion described in Section 6.4.1.

Trivially, DCE achieved perfect precision and recall for 0% noise (see Table 6.1). This still held true if the branching was controled. But if we restricted the analysis to results satisfying the size constraint, we lost recall in some cases. However, the recall level was still quite high, and it was perfect for $k = 4$. Furthermore, at 0% noise (density threshold 100%), any predicted cluster trivially satisfies the balance criterion. In contrast, the balance constraint makes a big difference in the 30%

Table 6.1: Performance analysis of DCE and its extensions for artificial data with varying dimension-wise cluster cardinality s at 0% and 30% noise. For each setting, we took the average across ten random datasets. The parameter k refers to the optional branching restriction. For noise level 0%, the clusters are trivially balanced. See text for details.

		Noise level 0%				Noise level 30%			
		$s = 2$	$s = 3$	$s = 4$	$s = 5$	$s = 2$	$s = 3$	$s = 4$	$s = 5$
Time (s)	-	0.00	0.07	0.79	7.91	0.00	0.41	6.26	58.01
	$k = 4$	0.00	0.07	0.74	6.83	0.00	0.06	0.63	5.08
	$k = 3$	0.00	0.07	0.62	5.10	0.00	0.05	0.47	3.17
	$k = 2$	0.00	0.05	0.36	2.22	0.00	0.04	0.24	1.17
	$k = 1$	0.00	0.02	0.08	0.26	0.00	0.02	0.06	0.16
Recall (%)	-	100.00	100.00	100.00	100.00	88.19	98.91	99.96	100.00
	$k = 4$	100.00	100.00	100.00	100.00	88.19	98.89	99.94	100.00
	$k = 3$	100.00	100.00	100.00	100.00	88.19	98.78	99.90	100.00
	$k = 2$	100.00	100.00	100.00	100.00	88.19	98.28	99.80	100.00
	$k = 1$	100.00	100.00	100.00	100.00	84.81	95.24	98.41	99.78
Precision (%)	-	100.00	100.00	100.00	100.00	95.21	80.26	54.28	29.84
	$k = 4$	100.00	100.00	100.00	100.00	95.21	82.02	58.37	34.45
	$k = 3$	100.00	100.00	100.00	100.00	95.21	83.73	61.25	38.75
	$k = 2$	100.00	100.00	100.00	100.00	95.53	86.60	69.69	51.47
	$k = 1$	100.00	100.00	100.00	100.00	97.07	94.07	89.07	84.67
Recall for size-restricted clusters (%)	-	100.00	100.00	100.00	100.00	57.00	59.50	54.03	48.51
	$k = 4$	100.00	100.00	100.00	100.00	57.00	59.50	54.03	48.51
	$k = 3$	100.00	100.00	99.50	100.00	57.00	59.50	53.53	48.51
	$k = 2$	100.00	100.00	99.00	99.00	57.00	59.00	48.52	44.01
	$k = 1$	100.00	99.52	97.50	98.00	56.50	39.01	22.01	10.50
Precision for size-restricted clusters (%)	-	100.00	100.00	100.00	100.00	99.35	99.72	100.00	100.00
	$k = 4$	100.00	100.00	100.00	100.00	99.35	99.72	100.00	100.00
	$k = 3$	100.00	100.00	100.00	100.00	99.35	99.72	100.00	100.00
	$k = 2$	100.00	100.00	100.00	100.00	99.35	99.72	100.00	100.00
	$k = 1$	100.00	100.00	100.00	100.00	99.67	100.00	100.00	100.00
Recall for balanced clusters (%)	-	see above				71.94	84.62	94.14	98.57
	$k = 4$					71.94	84.55	93.93	98.41
	$k = 3$					71.94	84.49	93.70	98.15
	$k = 2$					71.81	83.99	92.42	97.07
	$k = 1$					69.87	78.42	84.43	89.02
Precision for balanced clusters (%)	-					98.38	96.72	94.31	91.58
	$k = 4$					98.38	96.72	94.32	91.61
	$k = 3$					98.38	96.71	94.33	91.67
	$k = 2$					98.37	96.72	94.38	91.73
	$k = 1$					98.42	96.58	94.08	91.27
Recall for balanced size-restricted clusters (%)	-					27.50	1.50	0.00	0.50
	$k = 4$					27.50	1.50	0.00	0.50
	$k = 3$					27.50	1.50	0.00	0.50
	$k = 2$					27.50	1.50	0.00	0.50
	$k = 1$					27.50	1.50	0.00	0.50
Precision for balanced size-restricted clusters (%)	-					100.00	100.00	-	100.00
	$k = 4$					100.00	100.00	-	100.00
	$k = 3$					100.00	100.00	-	100.00
	$k = 2$					100.00	100.00	-	100.00
	$k = 1$					100.00	100.00	-	100.00

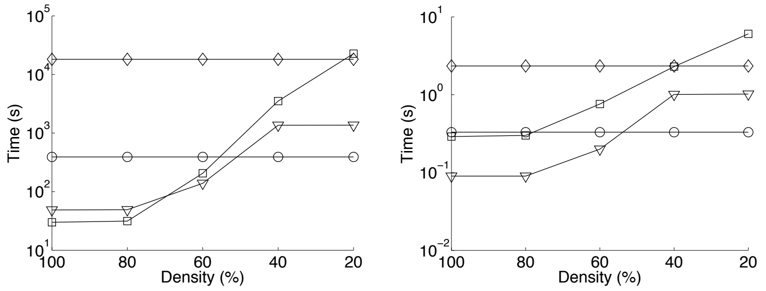
noise case. While we obtained 100% recall and approximately 29.84% precision for $s = 5$ using the unconstrained DCE algorithm, the balanced DCE clusters achieved 98.57% recall and 91.58% precision. Larger hidden clusters (i.e., higher values of

s) generally lead to higher recall and lower precision in the noisy case because they allow for more cluster variants. In conjunction with the minimum size constraint, DCE produced 48.51% recall with 100% precision (in the configuration with $s = 5$ and 30% noise). Note that during data generation, we fixed the overall fraction of flips across all clusters, not for each individual cluster, which explains the low recall value. Only a small fraction of the hidden clusters satisfied both size and balance constraints after adding 30% noise; the average recall was 0.5% for $s = 5$ and 27.5% for $s = 2$.

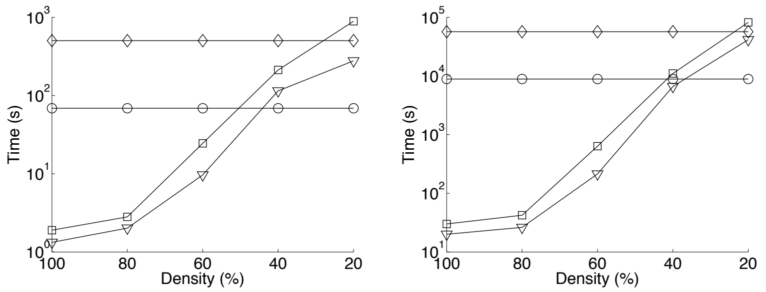
Regarding the influence of the branching parameter k, our empirical results suggest the following tendencies. Naturally, the recall increased for higher values of k. Remarkably, for $k = 3$ or $k = 4$ the recall was in most cases very close or equal to the recall obtained without branching constraints, although the runtime was significantly reduced. Sometimes, the recall level was already reached with $k = 2$. The precision of DCE clusters was higher for small k, which indicates that the heuristics is indeed successful in focusing on significant solutions. With respect to size or balance constraints, the precision remained approximately the same for all k and the unrestricted branching. In summary, the branching restriction is an effective technique to speed up the search procedure while maintaining the precision and recall levels of the complete algorithm.

6.6.3 Efficiency of Reverse Search

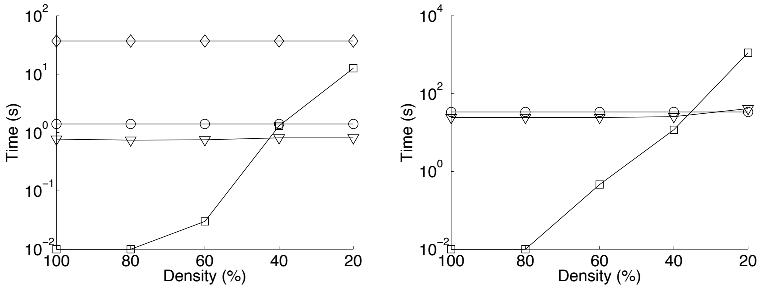
To investigate the efficiency of the reverse search approach, we compared its performance with other set enumeration strategies. For that purpose, we implemented two straightforward set enumeration approaches, which we call BruteForce and BruteNeigh. They use exactly the same data structures for tensor access and incremental density calculation as DCE (see description in Section 6.3.5), the only difference is in the traversal of the search space. BruteForce enumerates all possible subset combinations with respect to all dimensions and checks the corresponding cluster densities; the enumeration is done by nested loops, each of which constructs subsets incrementally by the aid of a lexicographical search tree (Section 5.3.1). BruteNeigh is a variant that is particularly suited for size-imbalanced and sparse datasets. It first chooses the dimension with the smallest number of instances and enumerates all its instance subsets in the same way as BruteForce; for each of these subsets, it exploits the corresponding adjacency relationships to determine relevant candidate instances ("neighbors") of the other dimensions, for which then all subsets are enumerated. This approach resembles mining approaches on bipartite graphs



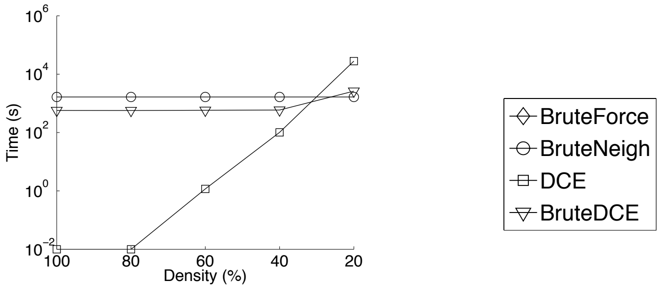
Mouse (max=2) Rat (max=2)



Rat (max=3) Rat (max=4)



Artificial (max=2) Artificial (max=3)



Artificial (max=4)

Figure 6.5: Runtime comparison of DCE and other set enumeration strategies in dependence of the minimum density threshold. The time axis is logarithmic. See text for details.

described in [206] and basically gains efficiency by eliminating isolated instances from the search space (Section 6.4.3). In addition, we looked at a combined version, BruteDCE, which enumerates all subsets in the smallest dimension and performs for each of them a separate reverse search with respect to the remaining dimensions, considering again only instances that are adjacent to the fixed instances of the smallest dimension.

To make the BruteForce approach feasible, we restricted our studies to two-dimensional data arrays and used maximum cardinality constraints for the dimension-specific instance sets (Section 6.4.4). On the one hand, we downloaded gene signatures for mouse and rat from GeneSigDB [46]; they contain a set of references to biological experiments, each of which is associated with a list of genes. Combining this information in a (binary-valued) data matrix allows to do a meta-analysis on results from different biological publications. The size of the data is 122×917 and 12×182 for mouse and rat, respectively; the densities of the whole datasets are 5% and 19%, respectively. On the other hand, we reused an artificial dataset from Section 6.6.1, with a size of 100×100 ($m = 20$, $s = 3$, noise level 30%) and an overall density of 2%.

The different search techniques were applied on the three datasets; the maximum cardinality (denoted as max) was set to the same value for both dimensions and ranged from 2 to 4. Figure 6.5 shows the resulting time curves for different density thresholds. The measurements were performed on a 2.8 GHz processor. We observed some general trends across the experiments. First of all, the neighborhood-based technique greatly payed off for straightforward set enumeration methods: BruteNeigh improved the performance of BruteForce by approximately one order of magnitude. Second, DCE showed an exponential runtime increase for decreasing density thresholds, but in the upper threshold range from 100% to 60% density, it outperformed BruteNeigh in most cases by a wide margin. Furthermore, considering fixed density thresholds, the performance gain of DCE increased with increasing levels of (maximum) cardinality. For density thresholds close to the overall density of the dataset, DCE was visibly worse than BruteForce (see rat plots); this is due to the overhead of the child generation process in the reverse search, which has to select the true children among all possible candidates (see Section 6.3.4).

Regarding the behavior of BruteDCE, there were major differences between the datasets. While the combined enumeration strategy was beneficial for the highly size-imbalanced rat dataset, consistently achieving lower runtimes than DCE, it had mainly a negative effect for the artificial dataset, where the number of rows equals the number of columns. For the mouse dataset, it outperformed DCE only for medium and low density thresholds. Note that the shape of the BruteDCE curve differs heavily between rat and artificial experiments, resembling the DCE curve and the BruteNeigh curve, respectively. This indicates that in the latter case, the brute force enumeration part dominated the computation time of BruteDCE, whereas in the former case the reverse search part played the main role. Overall, the empirical results show that in comparison with straightforward search approaches, the reverse search strategy considerably improves the efficiency of dense cluster enumeration.

6.6.4 Email Traffic Analysis

To illustrate higher-order cluster enumeration on real-world data, we applied DCE to a subset of the ENRON email dataset [119], which we took from [24]. It records information about the sender, the recipients, and the time stamp of emails. From this, we generated a three-way tensor as follows. We mapped each time stamp to the corresponding week and then determined the number of emails a certain sender sent to a certain recipient in a certain week. This yielded a $120 \times 143 \times 123$ tensor with 6995 non-zero entries. We were interested in groups of persons that regularly exchange many emails. The individual frequency values per week ranged from 1 to 202, however 81% of them were lower or equal to 10; we set entries with values greater than 10 to 10, in order to avoid cluster results that are dominated by one or very few outlier entries and consequently do not reliably describe associations between sets of instances.[3] After the preprocessing, we ran DCE with a density threshold of 80%. That means, for a valid cluster solution, each sender sends in each week on average at least 8 emails to each recipient, assuming a maximum number of 10 emails.

Restricting the maximum number of instances per dimension for each cluster to 4, we obtained approximately $3.5 \cdot 10^7$ clusters in total. This seems to be a large number of clusters, but it is reasonably small compared to the number of cluster candidates for the tensor at hand, which is $2.0 \cdot 10^{22}$ for the given maximum size constraint. Focusing on locally maximal patterns with at least two instances in each dimension, the size of the result set shrinks to 240675, and among them, there are only 142 clusters with at least three instances in each dimension. The top-ranking cluster (density: 82%, p-value: $4.7 \cdot 10^{-20}$) is shown in Figure 6.6. It contains three senders, four recipients and four weeks. Senders and recipients are given by personal identifiers. Two persons appear as both senders and recipients, one person appears only as sender, and two persons only as recipients. The only zero entries of the cluster are due to the absence of self-emails for the two persons in the overlap. This

---

[3] Other weighting schemes are possible, for instance binarization according to a predefined threshold.

Senders: 155, 162, 169
Recipients: 114, 155, 165, 169
Weeks: 103, 108, 118, 120

Sender	Recipient	No. of emails in week			
		103	108	118	120
155	114	≥10	≥10	≥10	≥10
155	155	0	0	0	0
155	165	≥10	≥10	≥10	8
155	169	≥10	≥10	≥10	8
162	114	≥10	≥10	≥10	≥10
162	155	≥10	≥10	≥10	≥10
162	165	≥10	≥10	≥10	≥10
162	169	≥10	≥10	≥10	≥10
169	114	≥10	≥10	≥10	≥10
169	155	≥10	≥10	≥10	≥10
169	165	8	≥10	≥10	8
169	169	0	0	0	0

Figure 6.6: Top-ranking cluster for email traffic data.

cluster remains the top-ranking cluster even if we drop the maximum cardinality constraints for senders and recipients, which means that there do not exist dense clusters involving more people.

6.7 Discussion

We presented a general framework for the systematic extraction of dense patterns from higher-order association data. It extends conventional relational set mining approaches [31, 100, 103] and clique-related network analysis [104, 166, 242]. The proposed reverse search algorithm allows for an effective pruning of the search space without missing any solutions. Remarkably, the complexity of the delay between two consecutive solutions is in the order of the input size. This property distinguishes the reverse search approach from straightforward set enumeration algorithms and makes it applicable in cases where the latter are infeasible. However, for large datasets or low density thresholds, the number of solutions is prohibitive, even if only maximal solutions are considered; consequently, the search method does not scale well.

There are several remedies for this problem. The first possibility is to maintain the enumerative search, but add further constraints based on additional criteria, prior knowledge, or external data. Often, it is possible to define anti-monotonic constraints, which contribute actively to the pruning of the search space. Or, if relevant subsets are prespecified for some dimensions (for instance, windows of consecutive time intervals), reverse search with respect to the other dimensions can be performed for each of these subsets individually. It is an open research question how minimum size or support constraints can be exploited most effectively in the dense pattern detection framework. On the other hand, one can still use the reverse search strategy, but apply heuristic criteria or sampling techniques to control the number, overlap, and relevance of solutions; this allows to directly trade off the runtime and the completeness of the solution set, which we illustrated in the experiments with a simple branching heuristic; similarly, heuristic pruning rules could be specified by appropriate thresholding of (relative) degree values. Even if it is not used for exhaustive exploration, the definition of the anti-monotonic search space has a value by itself, as solutions are visited with polynomial delay.

Finally, instead of applying the method to the whole dataset at once, it can be combined with different strategies of prepartitioning or preaggregation of the data [91, 122]. Furthermore, the reverse search strategy is compatible with distributed computation, and its efficiency can be further improved by adapting the data structures and pruning techniques to the specific task at hand.

Part III

Hierarchical Detection of Association Patterns

7 Hierarchical Biclustering

The enumerative approach to cluster detection discussed in the previous part is suitable for a systematic and detailed analysis of structured datasets, including two-way data. However, for large-scale applications it might be problematic due to the potentially huge number of solutions, which affects both the computational efficiency and the interpretability of the results. Here, we present a totally opposite approach, which provides clusters in two-way association data without aiming at completeness nor considering cluster overlaps. It extends the well-known paradigm of hierarchical clustering to a biclustering framework.

7.1 Motivation

Hierarchical clustering [97] is a common method to explore similarity or distance relationships between data instances; it is extremely popular in computational biology [84]. There exist two main strategies: top-down methods (also called *divisive*) iteratively split the set of instances into smaller sets, whereas bottom-up methods (also called *agglomerative*) iteratively merge sets of instances to larger ones. Let us focus on the latter case, which is the prevalent variant. It is computationally easier because it considers in each step only similarities between instance sets given from the previous step; this has, of course, the disadvantage that the obtained clusters reflect locally optimal rather than globally optimal structure in the data.

The starting point is a weight matrix that contains pairwise similarity or association values between instances. Based on that, a dendogram is constructed where the most similar instances are merged first (i.e., at the bottom), and less similar instances or instance sets join later, in decreasing order of similarity (see Figure 7.1). Common choices for similarity measures between instance sets and details of the procedure are given in the following section. A horizontal cut through the dendogram defines a specific set of instance clusters. In the generalized setting we are interested in, our input data are represented by a two-way matrix, and we would like to detect groups of related row instances as well as groups of related column

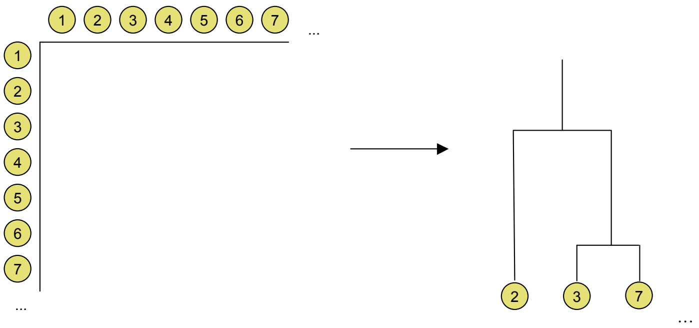


Figure 7.1: Schematic view of agglomerative clustering. Based on a pairwise association matrix, a dendogram is constructed.

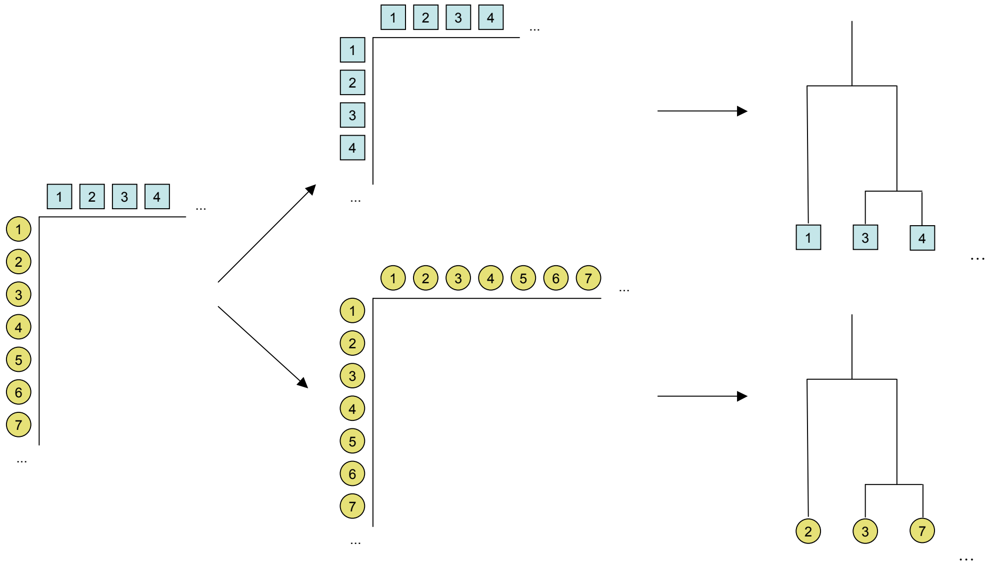
instances. The classical agglomerative way of tackling this problem is to compute two similarity matrices for row instances and column instances, respectively, and perform bottom-up clustering for each of them separately, see Figure 7.2 (a).

In contrast, we are aiming at a direct biclustering scheme, which builds a common dendogram for row and column instances directly from the data matrix, see Figure 7.2 (b). The critical difference compared to the classical two-way approach is that now grouping of row instances is no longer based on all columns, but only a subset of them, and vice versa. This comes closer to the motivation of biclustering (see Section 4.5), yielding direct assignments between row clusters and column clusters. It resembles a concept called two-mode hierarchical clustering [29, 217]. However, while these techniques employ a scoring criterion based on bicluster homogeneity, our approach focuses on biclusters with strong association weights in the original data matrix (see Section 4.5). Before we explain the algorithm in detail, we give a brief review on traditional agglomerative clustering. However, unlike many other presentations of that topic, we here merely start from an association matrix between instances and do not consider instances as original data points in real space. This allows us to deal with scenarios where we only know links or relationships between instances (e.g., in form of a graph), without being provided with feature information. In particular, we do not require metric properties for the distance measure between instances; also, the input matrix may contain negative values.

7.2 Review of Hierarchical Clustering

Here, we briefly review the foundations of agglomerative hierarchical clustering [54, 152]. Let V be the set of instances and $W = (w_{ij})_{i,j \in V}$ a symmetric weight ma-

(a) Independent two-way clustering



(b) Direct biclustering

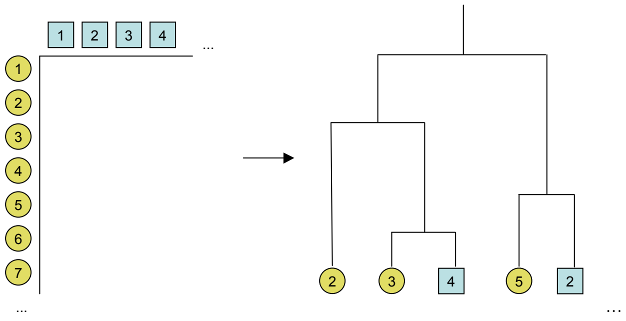


Figure 7.2: Comparison of agglomerative clustering strategies in two-way data. While the classical approach constructs for each dimension a similarity matrix and clusters them separately (a), our approach operates directly on the two-way data and yields a joint dendogram for both entity types (b).

trix indicating the pairwise similarity (or *association*) of instances. Initially, each instance constitutes an individual cluster, and W represents the cluster association values. Then, the clusters are iteratively merged to larger clusters. In each step, the two clusters with the largest association are selected for merging, and the cluster association matrix is updated accordingly. Regarding the definition of the association $a(S, U)$ between two clusters $S \subset V$ and $U \subset V$, there are the following three common variants:

$$a(S, U) = \max_{s \in S, u \in U} w_{su} \qquad \text{[single linkage]} \tag{7.1}$$

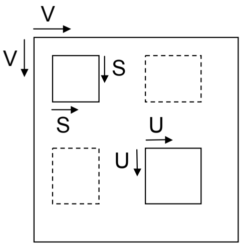


Figure 7.3: Two clusters S and U in a symmetric weight matrix. The dashed blocks indicate the matrix entries that associate S with U.

$$a(S,U) = \min_{s\in S,u\in U} w_{su} \text{ [complete linkage]} \tag{7.2}$$

$$a(S,U) = \frac{\sum_{s\in S,u\in U} w_{su}}{|S|\cdot|U|} \text{ [average linkage]} \tag{7.3}$$

Complete linkage has the strictest requirement, not allowing for any outlier values regarding the pairwise association between instances in a merged cluster; single linkage, on the other hand, is prone to join heterogeneous clusters as long as two instances from different clusters are closely associated; average linkage is a trade-off between these two extremes and will be our default choice. By construction, the different clusters are disjoint at each stage of the algorithm (i.e., $S \cap U = \{\}$ for all pairs S, U of clusters). Figure 7.3 visualizes two clusters as well as the entries of the weight matrix that determine their association. The iterative merging process is stopped if the best cluster association value is below a user-defined threshold; the current clusters are reported as the result. Alternatively, one can run the procedure until there is only one cluster left (containing all instances) and show the whole dendogram as a result.

As the procedure needs at most $|V| - 1$ merging steps to terminate, a naive implementation would require $O(|V|^3)$ time, using $O(|V|^2)$ operations in each step to update the between-cluster association values and search the maximum among them. The complexity can be further improved by applying computational standard techniques. On the one hand, the association values can be calculated incrementally, i.e., they can be derived from the association values in the previous step. On the other hand, appropriate data structures for accessing the association values can assist in obtaining the maximum value and performing the updates in an efficient way; typically, priority queues are used for that purpose (one for each cluster) [152]. With that, the overall complexity is reduced to $O(|V|^2 \log |V|)$.

A common scheme of doing the incremental update in the case of average linkage is expressed by the following recurrence formula [155]:

$$a(S \cup T, U) = \frac{|S| \cdot a(S,U) + |T| \cdot a(T,U)}{|S| + |T|} \tag{7.4}$$

To facilitate the generalization to the two-way setting, we slightly reformulate the above equation. Let $b(S,U)$ be the number of matrix entries that associate instances of S with instances of U (*between-cluster size*, see Figure 7.3); here, symmetric entries are counted only once, thus $b(S,U) = |S| \cdot |U|$. Then Equation (7.4) can be rewritten as indicated below:

$$a(S \cup T, U) = \frac{b(S,U) \cdot a(S,U) + b(T,U) \cdot a(T,U)}{b(S,U) + b(T,U)} \tag{7.5}$$

Further,

$$b(S \cup T, U) = b(S,U) + b(T,U)\,. \tag{7.6}$$

The initial values are set as follows:

$$a(\{s\},\{u\}) = w_{su} \qquad \forall s, u \in V \tag{7.7}$$

$$b(\{s\},\{u\}) = 1 \qquad \forall s, u \in V \tag{7.8}$$

It can be easily verified that the result of the recursive association calculation correponds to the direct expression obtained by Equation (7.3).

7.3 Agglomerative Biclustering Algorithm

Now we generalize the agglomerative clustering scheme to a biclustering scenario, aiming at detecting patterns of strong association.

7.3.1 General Scheme

Let us first introduce some notation. The data consist in a bipartite instance set V, with associations being only defined between instances of different type. We denote by V_1 and V_2 the two distinct sets of input instances (i.e., $V_1 \cap V_2 = \{\}$), and $W = (w_{ij})_{i \in V_1, j \in V_2}$ represents the corresponding association weights. Using matrix terminology, we call V_1 the row instances, and V_2 the column instances. Our goal is to extract bicluster patterns (S_1, S_2), $S_1 \subset V_1$ and $S_2 \subset V_2$, such that the average association weight between instances of S_1 and instances of S_2 is large. That means, equivalently to Chapter 6, a bicluster describes a submatrix of the association matrix with a large average across its entries (density). This criterion is only well-defined if the submatrix is non-empty, i.e., $S_1 \geq 1$ and $S_2 \geq 1$. Such biclusters are called *valid*. A merge of two biclusters (S_1, S_2) and (U_1, U_2) is defined by taking the union of the row instances and the union of the column instances, resulting in a larger bicluster $(S_1 \cup U_1, S_2 \cup U_2)$. For notational convenience, we represent a bicluster (S_1, S_2) by the joint set $S = S_1 \cup S_2$; as $V_1 \cap V_2 = \{\}$, the sets S_1 and S_2 can be easily reconstructed from S; accordingly, a merge of two biclusters S and U is written as $S \cup U$.

The method starts with a set of *base biclusters*, which correspond either to a row instance or to a column instance: $\{v\}$, $v \in V$. Then, the number of biclusters is successively reduced by merging in each step the bicluster pair with the largest association. To define the association between two biclusters, we use a generalized version of the average linkage measure (7.3):

$$a(S, U) = \frac{\sum_{s \in S_1, u \in U_2} w_{su} + \sum_{u \in U_1, s \in S_2} w_{us}}{|S_1| \cdot |U_2| + |U_1| \cdot |S_2|} \tag{7.9}$$

That means, the average association weight between instances from different biclusters determines how beneficial a merge would be. The matrix-based intuition of bicluster assocation is illustrated in Figure 7.4 (following immediately from Figure 7.3): the score corresponds to the average value of the matrix entries that would be added when merging the biclusters. It can be computed incrementally using the recurrence rules in Equations (7.5) and (7.6). However, we have to adapt the initialization to the bicluster setting. While the initial associations between row instances and column instances are given by W, base biclusters of the same kind (i.e., two row instances or two colum instances) are not qualified for merging because the resulting bicluster is not valid. This is indicated by setting the corresponding between-bicluster size value $b(.,.)$ to 0 and the association value $a(.,.)$ to a value that is lower than any entry in W:

$$a(\{s\}, \{u\}) = \begin{cases} w_{su} & \text{if } s \in V_1 \text{ and } u \in V_2 \\ w_{us} & \text{if } u \in V_1 \text{ and } s \in V_2 \\ \min\limits_{i \in V_1, j \in V_2} w_{ij} - 1 & \text{otherwise} \end{cases} \tag{7.10}$$

$$b(\{s\}, \{u\}) = \begin{cases} 1 & \text{if } s \in V_1 \text{ and } u \in V_2 \\ 1 & \text{if } u \in V_1 \text{ and } s \in V_2 \\ 0 & \text{otherwise} \end{cases} \tag{7.11}$$

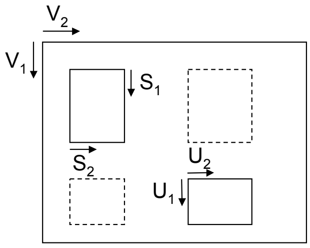


Figure 7.4: Two biclusters S and U. The dashed blocks indicate the matrix entries that are needed to determine the association between S and U.

It is not necessary to store the a- and b-values for all invalid merges; one can just use the defined default values if they are needed in the recurrence equation[1] and take care otherwise that only valid merges are considered; also, redundancy can be avoided by exploiting the symmetry of both a and b. Furthermore, it is possible to avoid an explicit storage of b by recording the set sizes of S_1 and S_2 for each bicluster S and using the following update rule:

$$a(S \cup T, U) = \frac{(|S_1| \cdot |U_2| + |U_1| \cdot |S_2|) \cdot a(S, U) + (|T_1| \cdot |U_2| + |U_1| \cdot |T_2|) \cdot a(T, U)}{(|S_1| \cdot |U_2| + |U_1| \cdot |S_2|) + (|T_1| \cdot |U_2| + |U_1| \cdot |T_2|)}$$

In analogy to the basic approach explained in Section 7.2, the iterative merging process can be stopped prematurely, based on the number of biclusters or the top association score among the current bicluster pairs. At any point of the algorithm, the biclusters are disjoint, so the result corresponds to a partitioning of the bipartite association graph.

7.3.2 Correctness

To obtain a well-defined hierarchy, we must guarantee that the merge scores decrease monotonically towards the top of the dendogram. That is, biclusters with larger association to each other should be combined earlier in the process. Formally, this is satisfied if – at each level of the hierarchy – all cluster association scores are smaller than or equal to the score of the previous merge (i.e., the best association score among the clusters in the previous level). Since only the association scores involving the newly merged cluster are changed compared to the previous step, it is sufficient to show the inequality for them. For standard agglomerative clustering methods, this property is proven in [155]. Similarly, we can formalize the anti-

---

[1] In fact, the a-value does not matter in that case (as long as it is finite) because the corresponding b-value equals 0.

monotonicity of our biclustering approach with the following lemma.

Lemma 13. *Given biclusters S, T, and U, let (S, T) be the bicluster pair with the maximum association score, i.e., $a(S, T) \geq a(S, U)$, $a(S, T) \geq a(T, U)$. Then,*

$$a(S \cup T, U) \leq a(S, T) .$$

Proof. By construction, $b(S, U) \geq 0$ and $b(T, U) \geq 0$. Further, $b(S, U) + b(T, U) > 0$ if $S \cup T$ is a valid bicluster (i.e., it contains row and column instances). Then, the inequation follows from the assumption by simple algebra:

$$\begin{aligned} a(S \cup T, U) &= \frac{b(S, U) \cdot a(S, U) + b(T, U) \cdot a(T, U)}{b(S, U) + b(T, U)} \\ &\leq \frac{b(S, U) \cdot a(S, T) + b(T, U) \cdot a(S, T)}{b(S, U) + b(T, U)} \\ &= \frac{a(S, T) \cdot (b(S, U) + b(T, U))}{b(S, U) + b(T, U)} \\ &= a(S, T) \end{aligned}$$

If $S \cup T$ is not a valid bicluster, the assumption implies that $S \cup U$ and $T \cup U$ are also not valid. Consequently, $(S \cup T) \cup U$ is not valid and $a(S \cup T, U) = a(S, T)$. □

The proposed algorithm produces valid biclusters in all merging steps because such merges are always possible and score better than merges yielding invalid biclusters. Due to the lemma, merge scores of biclusters are at least as good as merge scores of their parent bicluster in the hierarchy. As the merge score of a bicluster is defined as the average association value between instances of the first child and instances of the second child, this anti-monotonicity property also implies that the bicluster density (i.e., the average association value between all bicluster instances) monotonically decreases when going up the hierarchy.

7.3.3 Complexity

Now we analyze the time and space requirements of the hierarchical biclustering algorithm. For better readability, we define m_1 and m_2 as the cardinalities of the row instance set V_1 and the column instance set V_2, respectively. Without loss of generality, $m_1 \leq m_2$. In each step of the algorithm, two biclusters are merged, so there are at most $m_1 + m_2 - 1$ steps. In a naive implementation, one would maintain an array of association scores for all bicluster pairs, i.e., $O((m_1 + m_2)^2)$

entries. However, as base biclusters of the same type are not eligible for merging, the size can be reduced to $O(m_1 m_2)$. We suggest to use $O(m_1(m_1 + m_2))$ space, which allows for an easier bookkeeping, assigning each merged bicluster to exactly one row index rather than having to reserve both one row index and one column index for it. Similarly to the standard hierarchical clustering approach (Section 7.2), we can store these bicluster association values in priority queues. Using m_1 priority queues of maximum length (m_1+m_2), each update step requires $O(m_1 \log(m_1+m_2))$ operations, for removing the former entries of the merged biclusters and inserting the entries of the newly formed bicluster. So the time complexity of building a complete dendogram amounts to $O((m_1 + m_2)m_1 \log(m_1 + m_2))$.

In contrast to that, the independent two-way clustering method (Figure 7.2 (a)) needs $O((m_1)^2 m_2) + O((m_2)^2 m_1)$ time to compute the association matrices of row instances and column instances, respectively, and $O((m_1)^2 \log m_1) + O((m_2)^2 \log m_2)$ to compute the two separate dendograms (see Section 7.2). The space complexity for computing the the association matrices is $O((m_1 + m_2)m_1) + O((m_1 + m_2)m_2)$. Consequently, the proposed biclustering approach brings clear advantages in terms of computational resources if the size of the dataset is highly imbalanced, i.e., $m_1 \ll m_2$; in particular, it could be a practical alternative in cases where the calculation of the $m_2 \times m_2$ association table is infeasible. However, like all partitioning-based biclustering approaches, it carries the risk of missing relevant global relationships in favor of local relationships, whereas two-way clustering only considers global relationships.

8 Extensions of Hierarchical Biclustering

The hierarchical agglomerative biclustering scheme presented in the previous chapter provides a simple and quick way to obtain some high-density biclusters in two-way data. Here, we briefly mention some possible extensions of that approach, namely alternative association criteria, generation of alternative clusterings, and generalization to multi-way data analysis.

8.1 Alternative Association Criteria

Alternatively to the average linkage criterion proposed in Section 7.3, one can use single linkage or complete linkage measures to associate biclusters, in analogy to Equations 7.1 and 7.2. The corresponding recurrence formulae are straightforward, as the sets of entries defining an association are equivalent to the average linkage case (Figure 7.4), only the aggregation function changes. Beyond that, one can employ measures that additionally take the entries within biclusters into account. For instance, one could consider the overall average of entries within the merged bicluster as association criterion, equivalently to the bicluster density criterion defined in the enumerative pattern mining approach (Section 6.2):

$$a_{\text{total}}(S, U) = \frac{\sum_{s,u \in S \cup U} w_{su}}{\text{size}(S \cup U)}\,, \tag{8.1}$$

where $\text{size}(S) = |S_1| \cdot |S_2|$ denotes the number of entries in the bicluster submatrix. Then, the update rule looks as follows:

$$\begin{aligned}
&a_{\text{total}}(S \cup T, U) = \\
&[\,\text{size}(S \cup T) \cdot a_{\text{total}}(S, T) + \text{size}(S \cup U) \cdot a_{\text{total}}(S, U) + \text{size}(T \cup U) \cdot a_{\text{total}}(T, U) \\
&\;- \text{size}(S) \cdot a_{\text{total}}(S, S) - \text{size}(T) \cdot a_{\text{total}}(T, T) - \text{size}(U) \cdot a_{\text{total}}(U, U)\,]/ \\
&\text{size}(S \cup T \cup U)
\end{aligned}$$

This criterion would also lead to a dendogram with monotonically decreasing merge scores. However, considering the fact that the algorithm does not allow for overlapping biclusters, we prefer the average linkage criterion introduced in Section 7.3, i.e., biclusters should only be merged if they are strongly interlinked. Using average linkage, the overall bicluster density still decreases after each fusion step (due to Lemma 13), but the resulting hierarchy can differ from the hierarchy built with the density criterion. In addition, size specifications and constraints from other data sources, e.g., profile consistency (Sections 5.7.1 and 6.4.4) can be used to prohibit or restrict certain bicluster merges; however, due to the non-exhaustive nature of the algorithm, one might fail to find bicluster solutions even if they exist in the data.

8.2 Alternative Clusterings

Agglomerative clustering approaches are greedy procedures, performing the best possible merging step at each iteration. One obvious drawback is that decisions are never revised, that means, once instances are joined they will remain forever in the same bicluster. If there occurs a tie in determining the maximum pairwise bicluster association and the options exclude each other (i.e., they cannot be executed one after the other), one has to select one of them to continue the dendrogram. This introduces some arbitrariness in the process, and it might be the case that a particular choice harms the bicluster quality in all subsequent steps, while another choice would have yielded more significant biclusters. The problem is exacerbated in the case of binary-valued data. One possible solution is to construct multiple dendograms; this is feasible due to the moderate time complexity of the method, in particular if early stopping rules are applied. Each time a tie with mutually exclusive possibilities occurs, one can systematically try all of them or randomly sample a subset; in that way, different continuations of the current dendogram can be produced. In the end, the most promising biclusters across all dendograms can be chosen based on size and weight criteria. This results in a set of potentially overlapping biclusters. It is also conceivable to generate overlapping biclusters in a single dendogram by heuristic modifications of the algorithm scheme, e.g., by maintaining after each merge one of the two former biclusters in addition to the new bicluster. While the complexity per iteration remains the same as before, such an approach allows to reuse some biclusters, avoiding at the same time reoccurrences of identical merges.

8.3 Hierarchical Higher-Order Clustering

In section 7.3, we have shown how to generalize agglomerative hierarchical clustering to bipartite (i.e., biclustering) settings. Here, we discuss the problem of extending the framework to the multi-way scenario described in Chapter 6. More precisely, we consider n sets of instances, V_1 to V_n, and an n-way association weights $(w_{k_1,\dots,k_n})_{k_i \in V_i}$, which can be stored in an n-dimensional tensor. Again, we assume for notational convenience that the sets V_1 to V_n are disjoint and denote the union by V. A multi-way cluster S consists of n subsets $S_i \subset V_i$, and is denoted as $S = \bigcup_{i=1}^{n} S_i$. The goal of the hierarchical clustering approach is to extract multi-way clusters with large association between its instances. In analogy to the biclustering case, the possible initial merging steps join n instances, one from each set V_i; the merge score is given by the corresponding association weight. In later stages, the score is given by averaging[1] over the association weights between instances from different multi-way clusters (i.e., the newly added entries when merging the multi-way clusters). It would be natural to consider pairwise cluster merges whenever the resulting cluster is valid, containing at least one tensor entry. However, it turns out that the update scheme for merging scores is more complicated than in the bicluster setting. Namely, after merging two n-way clusters S and T, the association between the new cluster $S \cup T$ and another cluster U contains entries that are not contained in pairwise associations between S and U and between T and U.

More precisely, let $A(S,U)$ denote the set of tensor entries that will be added when merging S and U; we call it the *association set* of S and U. For the two-way case,

$$A(S,U) = \{(v_1,v_2) :\, v_1 \in S_1, v_2 \in U_2\} \cup \{(v_1,v_2) :\, v_1 \in U_1, v_2 \in S_2\}$$

(see Figure 7.4). In the general n-way case, $A(S,U)$ equals

$$\{(v_1,\dots,v_n) :\, v_i \in S_i \cup U_i\} \setminus (\{(v_1,\dots,v_n) :\, v_i \in S_i\} \cup \{(v_1,\dots,v_n) :\, v_i \in U_i\})\ .$$

For $n > 2$,

$$A(S,U) \cup A(T,U) \subsetneq A(S \cup T, U)\,;$$

for instance, an entry $(s_1, t_2, u_3, \dots, u_n)$ with $s_1 \in S_1$, $t_2 \in T_2$, and $u_i \in U_i$ belongs to $A(S \cup T, U)$, but it is neither a member of $A(S,U)$ nor a member of $A(T,U)$. Consequently, an anti-monotonicity property of the merging process cannot be guaranteed when focusing on pairwise merges. This makes it difficult to define stopping crite-

---

[1] Other aggregate functions are possible, e.g., maximum or minimum.

ria and to interpret the results. One solution could be to additionally consider all possible higher-order associations of clusters, corresponding to simultaneous merges of more than two clusters. In a straightforward implementation, such an approach would be computationally quite demanding; for a scalable multi-way analysis, a crucial point is the exploitation of sparsity in the input data. It is a subject of future work to develop a practicable agglomerative method for the detection of large-weight multi-way clusters.

Part IV

Biological Applications

9 Module Discovery in Protein Interaction Networks

Many cellular functions are performed by complexes that consist of multiple different proteins. The composition of these complexes may change according to the cellular environment, and one protein may be involved in several different processes. As there exist a number of experimental techniques to measure direct and indirect protein interactions (see Section 2.1), a common approach is to extract putative protein complexes from these data. The results can assist in the functional annotation of previously uncharacterized proteins as well as in revealing additional functionality of known proteins [193]. However, to perform this task, one has to face two challenges: on the one hand, the data are incomplete, that means many true interactions are missing (i.e., false negative), and on the other hand, the data are noisy, i.e., there is a large fraction of false positives. In fact, the reliability of the measured interactions strongly depends on the used experimental techniques, and it increases if the interactions are detected by several independent measurements. Therefore, it is meaningful to integrate all available interaction data into one network and assign weights to the edges based on their experimental evidence [99]. The DME algorithm presented in Section 5.3 uses a search criterion that naturally combines the two aspects by taking the average interaction weight, where missing interactions obtain a weight of zero. It can also incorporate additional information to guide the search towards the module patterns of interest.[1] Here, we show our DME results on data from yeast and human, which have been published in [67]. Several possible extensions are discussed at the end of the chapter.

---

[1] implementation available at `http://www.kyb.tuebingen.mpg.de/~georgii/dme.html`

9.1 Data Collection and Preprocessing

For the analysis of yeast (more precisely, *S. cerevisiae*), we combined protein interactions from DIP[2] [230] and MPact[3] [76] (which includes data from IntAct[4] [85], MINT[5] [35], and BIND[6] [15]), and interactions from the core datasets of the TAP mass spectrometry experiments by Gavin *et al.* [62] and Krogan *et al.* [132]. The human protein interactions were extracted from the IntAct, MINT, BIND, DIP, and HPRD[7] [174] databases.[8] For the computation of interaction weights, we followed the method in [99]. For that purpose, we first determined for each individual interaction the set of supporting sources. The MPact database reports for each measured interaction a corresponding set of experimental techniques. The IntAct, MINT, DIP, and BIND databases also list experimental methods, according to the PSI-MI standard. In these cases, we considered each experimental technique as separate source. Further, the HPRD dataset and the Gavin [62] and Krogan [132] datasets were labeled as individual sources.

For each combination of sources, we estimated a reliability score using gold standard sets of correct and incorrect protein pairs. For the gold standard set of positive examples, we collected protein pairs that share the same MIPS functional category.[9] These categories represent general functional relationships of proteins and are therefore more comprehensive than the known protein complexes used for evaluating the module predictions. The reason for this choice is to avoid overfitting the interaction weights to the reference complexes. For the negative set, we used protein pairs with different subcellular localization as given in the Gene Ontology database[10] [13]. Given these gold standard sets, a specific set of sources S was scored as follows. Let I_S be the set of interactions with evidence S, I_{pos} the set of positive interactions, and I_{neg} the set of negative interactions. To determine the weight of the interactions in I_S, we took the ratio between the true positive rate and the false positive rate:

$$w_{ij\,,\{i,j\}\in I_S} = \frac{|I_S \cap I_{\text{pos}}|/|I_{\text{pos}}|}{|I_S \cap I_{\text{neg}}|/|I_{\text{neg}}|} \tag{9.1}$$

---

[2] `http://dip.doe-mbi.ucla.edu`
[3] `http://mips.gsf.de/genre/proj/mpact`
[4] `http://www.ebi.ac.uk/intact`
[5] `http://mint.bio.uniroma2.it`
[6] `http://bind.ca`
[7] `http://www.hprd.org`
[8] For all datasets we used the versions available in May 2007.
[9] `http://mips.gsf.de/projects/funcat`
[10] `http://www.geneontology.org/`

Protein pairs without any support in the data received a default weight of zero because an interaction is a priori unlikely; alternatively, one could predict interaction weights based on additional information about the proteins. The raw ratio scores can have a large variance due to some outlier values. This can distort the analysis by undesired artifacts where modules with moderate density contain one node pair with extremely high interaction weight and poor interaction weights otherwise; larger density thresholds are only satisfied by two-node modules representing the most extreme outlier interactions, so they suffer from very low recall. Therefore, as in Section 6.6.4, we truncated the score distribution at a fixed value t and set all larger weight values to t. For the experiments reported here, we used $t = 2$, i.e., all interactions with an at least two-fold larger true positive rate for the sources of evidence received a top weight. To reduce the number of noisy interactions, we considered only interactions with weights greater than or equal to $1/t = 0.5$. For convenience, the interaction weights were scaled such that the maximum is 1.[11]

The resulting interaction network for yeast consisted of 3559 nodes with 14 212 non-zero interactions having an average weight of 0.67. The human network contained 9371 nodes and 32 048 non-zero interactions having an average weight of 0.47.

9.2 Comparative Analysis on the Yeast Interaction Network

First, we validated the performance of DME on the yeast interaction network in comparison with four other methods: clique detection (Clique), the clique percolation method (CPM) [166], a procedure for joining cliques of a certain size to larger clusters, CPMw [57], an extension of CPM that includes an additional clique filtering step, and Markov clustering (MCL) [53, 216], a popular graph clustering method simulating random walks.[12] As a reference set of confirmed protein complexes, we used the manually curated yeast complexes provided by MIPS[13] [75].

---

[11] Other scoring schemes are conceivable, e.g., log-transformation of ratio scores or discretization approaches; possible alternatives to the ratio score for indicating the confidence in an interaction are information retrieval measures like precision. Empirically, the DME module analysis turned out to be relatively robust in the sense that the top results of different weighting functions were similar; however, the recall of modules generally depends on the skewness of the weight distribution.

[12] For Clique, CPM, and CPMw we used the implementation from `http://www.cfinder.org`; MCL was downloaded from `http://micans.org/mcl`.

[13] `http://mips.gsf.de/genre/proj/yeast`

9.2.1 Precision-Recall Analysis

To evaluate the results, we chose performance measures that are based on protein pairs, in analogy to the measures introduced in Chapter 6.6 for the synthetic data analysis. In contrast to module-based measures, they take overlapping submodules only once into account. Defining the intersection of pairs from predicted modules and pairs from known complexes as *correctly predicted pairs*, precision and recall can be expressed as follows:

$$\text{Precision} = \frac{\text{No. of correctly predicted protein pairs}}{\text{No. of protein pairs in predicted modules}} \tag{9.2}$$

$$\text{Recall} = \frac{\text{No. of correctly predicted protein pairs}}{\text{No. of protein pairs in known complexes}} \tag{9.3}$$

To obtain precision-recall curves, we iteratively calculated the precision and recall values, each time extending the set of considered modules by the next module in a ranked order. As the other methods do not provide a module ranking and our p-value criterion from Section 5.4.2 is only applicable to enumerative approaches, we used the scoring scheme by Bader and Hogue [16], which is also mentioned in Section 5.4.2. In fact, it produced for our DME results almost the same ranking as the p-value criterion; the corresponding precision-recall curves were virtually equivalent. For each method, we tested a wide range of parameters and selected the configuration with the largest area under the precision-recall curve. For DME, we varied the density threshold from 100% to 95.5% using decrements of 0.5. As the number of solutions drastically increased between 96% and 95.5%, we further analyzed this range using decrements of 0.1%. The best result was achieved at a density threshold of 95.7%.

Instead of handling edge weights during the search, Clique and CPM preprocess the data based on a minimum edge weight parameter: before computing the cliques in the network, they remove all edges that violate the threshold. We varied this threshold from 1 to 0.35, using decrements of 0.05 (optimum for Clique: 0.4). CPM has in addition an integer parameter k to determine the size of the cliques that are considered for joining; it was tested in the default range between 3 and the maximum clique size in the network (optimum for CPM: edge selection threshold 0.9, k=9). This parameter k also exists for CPMw, but instead of preselecting the edges that may be used during the clique search, CPMw expects a threshold for the geometric mean of the edge weights in a clique. Only cliques satisfying this threshold are further processed. We tried the same thresholds as for DME; k was tested from 3 to

7, as the filtering step gets very expensive for higher values. We obtained the best results for k=6 and a clique selection threshold of 97%. For MCL, there exist two main parameters affecting the cluster granularity: the inflation parameter, which we varied from 1.5 to 8 using increments of 0.5, and the centering, varied between 1 and 5. Furthermore, we set the parameter to retain potentially generated cluster overlaps. Here, we got the best result for inflation 3.0 and centering 2.

Figure 9.1 shows the best precision-recall curve for each approach. Overall, the predictions of DME were competitive. Quite in the beginning, the curve shows a sudden drop, which is due to a big module that is not annotated as a known complex. Clique detected the same module, but it additionally found some other, higher-ranked modules, so the drop happens later. For higher recall levels, Clique has lower precision than DME; this indicates that for smaller modules (which appeared later in the ranking) the interaction weights got important for the module quality. The curves for MCL and CPM always stay below the DME curve. By explicitly using the edge weights and tuning the density parameter, DME allows for more flexibility than the two-stage procedure of CPM, first selecting edges and subsequently joining together all cliques that satisfy an overlap criterion. CPMw refines CPM by joining selected cliques only, but, in contrast to DME, it does not control the density of the merged modules and might also miss some dense modules. In our analysis, CPMw improved the result obtained by CPM, but was mostly inferior to Clique or DME.

One problem with the computational evaluation of module finding methods is the incomplete ground truth. For instance, if we predict additional compounds for a known complex, they are classified as wrong although they might be real. The analysis is further complicated by the fact that a protein can participate in multiple complexes. Also, the module characteristics differ from method to method as well as within methods. Therefore, we show in the next section additional statistics for comparing the results.

9.2.2 Further Result Statistics

Table 9.1 summarizes further statistics regarding the predicted modules of the different methods. As each of the enumerative methods (DME and Clique) produced a large number of nearly identical modules, we additionally computed a set of *distinct modules* for better comparability with the other methods. For that purpose, we grouped similar modules together and represented each group by its top-ranking module. To decide whether two modules M_1 and M_2 *match* each other, we here computed the *overlap score* proposed by Bader and Hogue [16]. It is defined as the

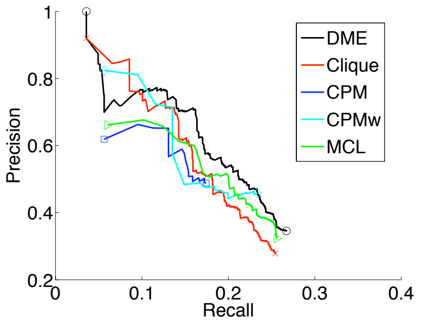


Figure 9.1: Comparative precision-recall analysis for yeast modules. To account for module overlap, the measures are based on protein pairs, see text.

fraction of overlapping proteins with respect to the size of the first module multiplied by the fraction of overlapping proteins with respect to the size of the second module:

$$\frac{|M_1 \cap M_2|^2}{|M_1| \cdot |M_2|} \tag{9.4}$$

For values of at least 0.5, the corresponding module pair was considered as a match. The same criterion was used to determine matches between predicted modules and known complexes. Note that the cutoff of 0.5 is relatively stringent [16]. All modules that are reachable from each other via a path of matching modules are assembled into one group. The distinct modules are given by the group representatives. They can share common proteins, so “distinct” is not equivalent to “non-overlapping” in that context. In fact, module overlaps happened in all methods.

While DME and Clique discovered a comparable number of distinct modules, the DME modules match many more known complexes. Among these, we also find small-sized complexes, so the overall average size of retrieved complexes is lower than for Clique. In addition, we report the number of complexes from which at least one protein pair was recovered (here called *partially recovered* complexes) as well as the area under the precision-recall curve from the pairwise analysis in the previous section. In both cases, DME was leading. Furthermore, we investigated the enrichment of the distinct modules with respect to Gene Ontology (GO) terms. For that, we applied the Expander tool [190] using the default setting with a p-value threshold of 0.05 after correction for multiple testing. Beside the total number of enriched modules, we also counted the number of enriched modules among the top-50 distinct modules, showing that for each method that produced more than 50 modules, most of the high-ranking modules satisfy the enrichment criterion. For small modules, the enrichment test failed even if they were totally pure.

Finally, we analyzed the properties of module overlaps in detail. Concerning the

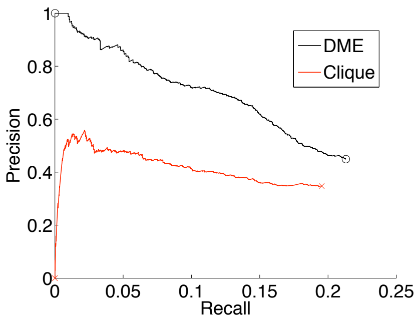


Figure 9.2: Precision-recall curves for overlapping interactions in yeast modules.

number of proteins or protein pairs that appear in more than one module, there is large variation among the different methods. DME and Clique produced the largest numbers of overlapping proteins and overlapping pairs. To evaluate the accuracy of the interactions in module overlaps, we sorted them according to the number of modules in which they occur (in descending order) and computed precision and recall values with respect to the MIPS protein complexes as before. Figure 9.2 shows the resulting precision-recall curves. Remarkably, in the DME results the precision is very high for the most frequently occuring interactions and monotonically decreases with decreasing frequency. In the Clique results, however, the top interactions are very unreliable, while the remaining curve has a similar shape as for DME. This difference is also reflected by the corresponding AUC values (see Table 9.1). The reason for the discrepancy in the behavior of DME and Clique is that DME respects edge weights and tends to reuse dense core sets, whereas Clique modules are solely based on the topology (after a preselection of edges). We also analyzed how many overlaps between known complexes were rediscovered by predicted modules. Formally, we counted the cases of overlapping known complexes C_1 and C_2 where there existed overlapping modules M_1 and M_2 such that the following conditions were satisfied:

1. $M_1 \cap M_2$ contains at least one element of $C_1 \cap C_2$.
2. $M_1 \setminus M_2$ contains at least one element of $C_1 \setminus C_2$.
3. $M_2 \setminus M_1$ contains at least one element of $C_2 \setminus C_1$.

The number of recovered overlaps was not significantly higher for DME.

Table 9.1: Module statistics of the comparative analysis for yeast (see text for details). The average size of the raw modules can be larger than for the distinct modules because larger modules allow for more variants. Time measurements were performed on a 2.66 GHz processor.

	DME	Clique	CPM	CPMw	MCL
No. of distinct modules	1083	916	19	32	648
Average size of distinct modules	3	4	16	14	3
No. of raw modules	24803	1971	19	33	648
Average size of raw modules	10	6	16	14	3
No. of matched complexes	84	54	9	20	59
Average complex size	5	7	19	14	7
No. of partially recovered complexes	133	107	20	33	117
No. of predicted interactions	5970	7066	2756	3935	6108
Area under prec.-rec. curve (AUC)	0.183	0.166	0.107	0.153	0.148
No. of enriched distinct modules	112	131	18	32	69
No. of enriched among top-50	47	44	-	-	45
No. of overlapping proteins	1010	1113	12	38	1
No. of overlapping interactions	3664	4340	24	114	0
AUC for overlapping interactions	0.152	0.082	0.000	0.001	-
No. of recovered complex overlaps	18	16	0	4	0
Time (s)	1167	4	4	267	4

9.3 Comparative Analysis on the Human Interaction Network

Similarly as for the yeast interaction network, we performed a comparative analysis on human data, considering the same module detection methods as before: DME, Clique, CPM, CPMw, and MCL. For validation, we used the reference set of human complexes published in the CORUM database [182] and applied the same evaluation measures as in the case of yeast. For DME, we checked density thresholds between 100% and 80% in decrements of 1%.[14] The best precision-recall results were obtained with 94%. The edge selection threshold of Clique and CPM was again varied from 1.0 to 0.35, and the clique size parameter k of CPM was checked in the whole range between 3 and the maximum detected clique size. Clique was optimal for the edge selection threshold 0.35, and CPM was optimal for the edge selection threshold 0.35 and k=6. In the case of CPMw, the clique selection threshold was set to the same values as the DME density threshold and k was varied from 3 to 7. Here, the optimal configuration was achieved for a clique selection threshold of 81% and k=6. Finally, the MCL parameters were tested in the same range as for the yeast experiments,

---

[14] As the human data are less dense (i.e., the average interaction weight is lower), we checked a wider range of density thresholds than for yeast.

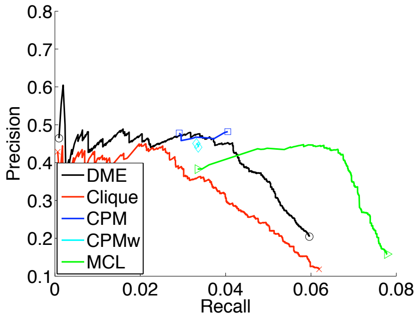


Figure 9.3: Comparative precision-recall analysis for human modules based on pair counts.

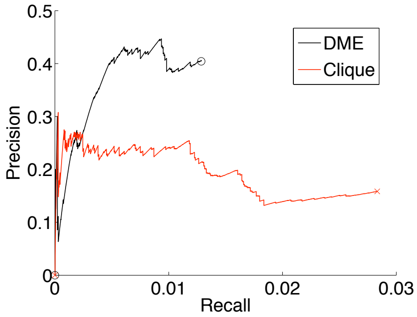


Figure 9.4: Precision-recall curves for overlapping interactions in human modules.

the inflation parameter from 1.5 to 8 using increments of 0.5 (best value: 2.0) and the centering parameter from 1 to 5 (best value: 3).

Figure 9.3 shows for each module detection method the precision-recall curve with the largest AUC. In general, the performance is much worse than on the yeast data (Figure 9.1), irrespective of the chosen method. This can be explained by the fact that the human data are sparser and, consequently, true complexes are harder to distinguish from noise by the density criterion. DME has a high peak in the beginning, but for medium recall, its precision is comparable to CPM and CPMw. Interestingly, CPM and CPMw achieved this by producing few large modules, whereas DME detected many small modules and therefore captured more diverse known complexes (see Table 9.2). The DME curve is above the Clique curve, but for higher recall values, DME is significantly outperformed by MCL, which also yields the largest area under the curve (see Table 9.2). Again, this might be an effect of the data sparseness: as one cannot gain very much accuracy by explicitly considering the module density, partitioning methods, which take the global connectivity structure of the network into account, are advantageous.

Table 9.2: Module statistics of the comparative analysis on the human network.

	DME	Clique	CPM	CPMw	MCL
No. of distinct modules	2321	2982	9	3	2092
Average size of distinct modules	2	3	12	19	3
No. of raw modules	3005	3420	9	3	2093
Average size of raw modules	3	3	12	19	3
No. of matched complexes	136	133	18	1	113
Average complex size	3	4	11	32	4
No. of partially recovered complexes	378	403	77	32	387
No. of predicted interactions	3925	7055	1131	1026	6616
Area under prec.-rec. curve (AUC)	0.025	0.021	0.019	0.015	0.030
No. of enriched distinct modules	24	58	8	3	64
No. of enriched modules among top-50	23	25	-	-	21
No. of overlapping proteins	970	1225	3	0	103
No. of overlapping interactions	428	2405	3	0	7
AUC for overlapping interactions	0.0046	0.0055	0.0002	-	0.0000
No. of recovered complex overlaps	942	1618	6	0	0
Time (s)	6	1	1	7	84

In Table 9.2, we collected various statistics for the results of the different methods, like in the yeast analysis. MCL required the longest computation time. DME matched the largest number of known complexes, closely followed by Clique. With respect to partially recovered complexes, Clique was leading, followed by MCL and DME. The number of GO-enriched modules was generally very low: CPM and CPMw produced only few modules, and the other methods predicted mainly modules that are too small to satisfy the enrichment p-value threshold, irrespective of their purity. Among the top-50 distinct modules, approximately one half is enriched. Regarding the number of protein pairs in the overlaps between different modules, the results are negligible for all methods except DME and Clique. Although the number is much higher for Clique, the area under the precision-recall curve is not very different, which means that overlapping interactions of DME are on average more accurate (see Figure 9.4). However, the accuracy is generally not higher than for the total set of predicted interactions (at comparable recall levels); a reason for that could be the small overlap sizes. Also, there is no clear dependency on the occurrence frequency, as the shapes of the curves reveal. Finally, the overlapping modules of Clique and DME also recovered a large number of overlaps between known complexes.

Table 9.3: Results of DME experiments with constraints.

	Phenotype (yeast)	Conservation (yeast)	Expression (human)
No. of distinct modules	137	1067	460
Average size of distinct modules	3	3	2
No. of raw modules	160	1816	736
Average size of raw modules	4	5	3
No. of matched complexes	14	49	52
Average complex size	4	4	4
No. of partially recovered complexes	30	103	217
Time (s)	13	3	2

9.4 Phenotype-Associated Modules in Yeast

An additional feature of DME is the possibility to directly integrate constraints from external data sources (see Section 5.7.1). As an example application, we investigated the yeast interaction network in the context of knockout phenotypes, in order to identify essential parts of protein complexes. For that purpose, we took the phenotype profiles for yeast knockout mutants under 21 experimental conditions from [50], considering three different phenotypic states: enhanced growth, normal growth, and growth defect. We then applied DME with a phenotype consistency constraint, requiring for each module at least one condition that is associated with growth defect for all members. To get a set of modules that covers a large number of proteins, but is at the same time as reliable as possible, we tested density thresholds between 95% and 80% using decrements of 1% and selected the one with the largest area under the precision-recall curve, namely 83%.

The results are summarized in Table 9.3. Each of the 13 top-ranking modules covers a considerable part of the mitochondrial ribosomal large subunit as annotated by MIPS. In addition, our output list contained one further module that overlaps with the complex. Figure 9.5 (a) shows the superposition of these 14 modules. Mrpl16 and Img2 appear in all, many other proteins in almost all of those modules, so they can be considered as the core of the complex. Knockout of any of the shown proteins caused growth defects with glycerol as carbon source. Several module members belong to other MIPS complexes, as depicted by the ellipses. In particular, there is a strong connection to the small subunit of the mitochondrial ribosome and to the mitochondrial translation complex. Furthermore, our results suggest that the

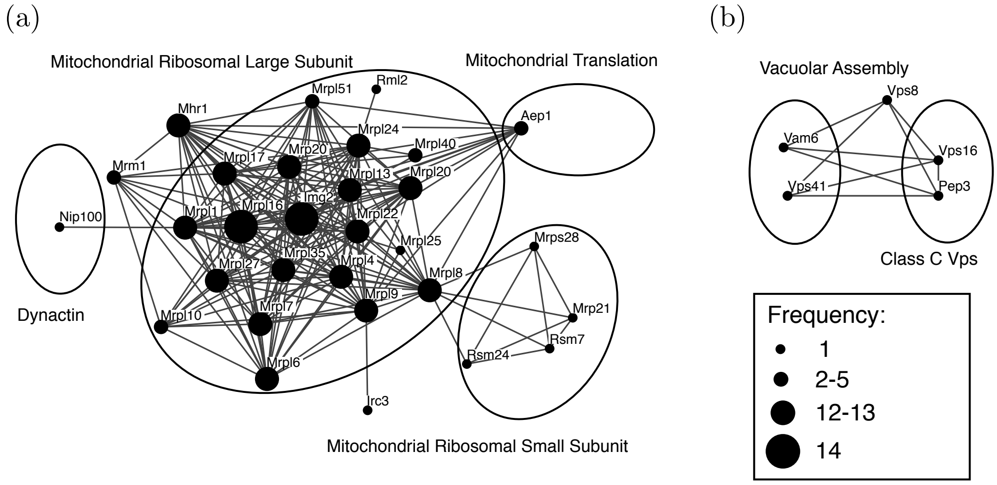


Figure 9.5: Phenotype-associated yeast modules. (a) Superposition of all 14 modules overlapping with the large subunit of the mitochondrial ribosome (node size depends on the number of modules in which the protein occurs). (b) Module linking two complexes. The ellipses mark protein sets belonging to known complexes. For module visualization, we used the Osprey tool [27].

mitochondrial ribosome is associated with Mhr1, a protein involved in homologous recombination of the mitochondrial genome [144].

Some modules that are not related to MIPS complexes nevertheless represent known complexes. For instance, we exactly recovered the nucleoplasmic THO complex (Hpr1, Mft1, Rlr1, Thp2), which is known to affect transcription elongation and hyper-recombination [36]. Interestingly, the corresponding mutants exhibit growth defects under the stress condition of adding ethanol to the medium. Finally, in Figure 9.5 (b) we show the highest ranking module that covers at least 50% of two different MIPS complexes. The corresponding proteins are associated with growth defects under addition of the aminoglycoside hygromycin B. The module links the vacuolar assembly complex with the class C Vps complex. The latter is a specific subgroup of proteins involved in vacuolar protein sorting. Indeed, it has been shown that this complex associates with Vam6 and Vps41 to trigger nucleotide exchange of a rab GTPase regulating the fusion of vesicles to the vacuole [229].

9.5 Evolutionary Conserved Modules in Yeast

Next, we used the evolutionary conservation of proteins as a side constraint for DME. For that purpose, we extracted from the InParanoid database [163][15] infor-

---

[15] `http://inparanoid.sbc.su.se`

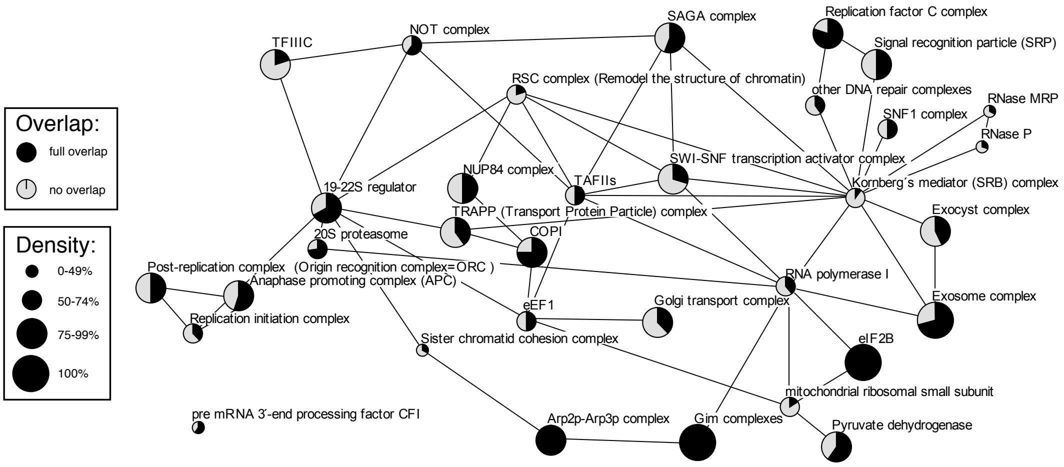

Overlap:
full overlap
no overlap
Density:
0-49%
50-74%
75-99%
100%
TFIIIC
NOT complex
SAGA complex
Replication factor C complex
Signal recognition particle (SRP)
RSC complex (Remodel the structure of chromatin)
other DNA repair complexes
RNase MRP
SNF1 complex
RNase P
SWI-SNF transcription activator complex
NUP84 complex
TAFIIs
Kornberg´s mediator (SRB) complex
19-22S regulator
TRAPP (Transport Protein Particle) complex
Exocyst complex
20S proteasome
COPI
Post-replication complex (Origin recognition complex=ORC)
Anaphase promoting complex (APC)
RNA polymerase I
Golgi transport complex
Exosome complex
Replication initiation complex
eEF1
Sister chromatid cohesion complex
eIF2B
mitochondrial ribosomal small subunit
pre mRNA 3´-end processing factor CFI
Arp2p-Arp3p complex
Gim complexes
Pyruvate dehydrogenase

Figure 9.6: Yeast complexes retrieved by DME and their overlap with conserved DME modules. Only complexes with size ≥5 are shown. The node size corresponds to the density of the confirmed complex, and the pie chart indicates to which degree the complex is covered by a conserved module. Nodes are connected if there exist interactions between the corresponding sets of matching modules.

mation about orthologs of *S. cerevisiae* genes with respect to ten other representative eukaryotic species: *S. pombe*, *D. discoideum*, *C. elegans*, *D. melanogaster*, *T. rubripes*, *X. tropicalis*, *M. musculus*, *H. sapiens*, *O. sativa*, and *A. thaliana*. For each *S. cerevisiae* gene, we created a profile indicating which of the other species have an ortholog with a full InParanoid score. Based on that, we searched for modules in the yeast interaction network such that there exist orthologs for all member proteins in at least three other species; the density threshold was determined using the same procedure as before and reached the optimum at 94%. For a summary of the resulting set of *conserved* (dense) modules, see Table 9.3. Note that the orthology constraint drastically reduced the running time compared to the unconstrained case (see Table 9.1); although the density threshold was lower, DME was now more than 400 times faster. Among the 103 at least partially recovered complexes, 49 were well matched.[16] In comparison, the unconstrained module enumeration partially retrieved 133 complexes, including 84 matches. The number of distinct modules was very similar (1083 for the unconstrained experiment, and 1067 for the constrained experiment).

Figure 9.6 shows an overview of the larger MIPS complexes that were recovered by DME. It includes matches from unconstrained and conserved modules. Appar-

---

[16] To define matches between complexes and predicted modules, we used the same criterion as in Section 9.2.

ently, it could identify some low-density complexes by discovering their dense core parts, for example the translation elongation factor complex eEF1 and the pre-mRNA 3'-end processing factor CFI. In black, we indicate the percentage of the known complex that is covered by a conserved module (in terms of shared proteins). From the total set of 33 retrieved complexes containing at least five proteins (see Figure 9.6), 19 complexes are hit to an extent of at least 50%; most of these (15 in total) are also matches of conserved modules according to the definition introduced in Section 9.2; among them, we find the 20S proteasome and its cap as well as the translation initiation factor eIF2B complex. The remaining complexes have rather small overlaps with conserved modules, even though they are quite accurately matched by their unconstrained counterparts. Our conserved module predictions reveal putative core parts of complexes that are conserved across several species. As an example, we consider the SNF1 complex, an essential element of the glucose response pathway consisting of six proteins. Indeed, while the components Snf1, Snf4, and Sip2 are strongly conserved in all eukaryotes and are covered by a conserved module, Sip1 and the transcription factor Sip4 do not have orthologs in other species, and the Gal83 component has orthologs in two species only [218]. Our approach predicted one additional conserved component of the complex, Sak1. This is biologically meaningful, as it functions as an activating kinase of the SNF1 complex [52]. The unconstrained module contained Sak1 and all SNF1 components except Sip4.

9.6 Tissue-Specific Modules in Human

Finally, we were interested in tissue-specific modules of the human interaction network. As side information, we downloaded the gene expression profiles by [202], containing measurements in 79 different human tissues that are classified into three states: present, absent, or marginal. For our purposes, we considered a gene to be expressed in a given tissue only if it was annotated as present in both of the duplicated measurements. To find complexes that are present in several, but not all tissues, we applied DME with the constraint that modules should be consistently expressed in at least three tissues and consistently absent in at least ten tissues. We used again the same procedure for selecting the density parameter and ended up with a threshold of 81%, which yielded 460 distinct modules (see Table 9.3).

The two top-ranking modules, shown in Figure 9.7 (a), cover the MCM complex, which is a hexameric protein complex required for the initiation and regulation of eukaryotic DNA replication. The DME modules contain two additional proteins,

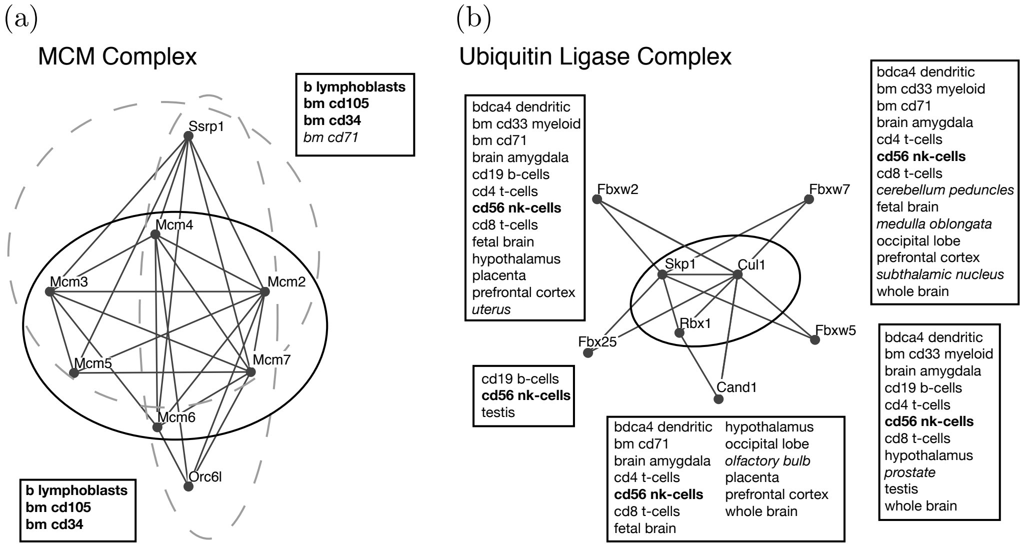


Figure 9.7: Tissue-specific modules in human. (a) The two top-ranking modules, covering the MCM complex. Known complexes are indicated as solid ellipses, modules as dashed ellipses. (b) Top-five modules around the SCF ubiquitin ligase complex, revealing its tissue-specific organization. Boxes show the tissues of consistent positive expression for the respective module. Tissues associated with all modules are marked in bold, uniquely appearing tissues in italics.

Ssrp1 and Orc6l. Orc6l is a member of the origin recognition complex (ORC), which plays a central role in replication initiation; in fact, the MCM and ORC complexes form the key components of the pre-replication complex [136]. This is nicely reflected by the large interaction density as well as the common expression profiles of the proteins: the module is completely expressed in three different types of bone marrow cells and completely missing in 42 tissues like brain, liver, and kidney, for example, where cells are differentiated and divide rarely. Ssrp1 is a member of the FACT complex, which is involved in chromatin reorganization [165].

Moreover, our analysis yields some insights about the tissue-specific organization of the SCF E3 ubiquitin ligase complex, which marks proteins for degradation. Figure 9.7 (b) depicts the five top-ranking modules that cover the complex (beyond these, there were three other modules covering only a single protein of the complex). One module contains as an additional component Cand1, a regulatory protein that inhibits the interaction of Cul1 with Skp1 [244]. The four other peripheral proteins are F-box proteins, which serve as substrate recognition particles for the SCF complex. Interestingly, the corresponding modules show different tissue specificities, indicating that the target proteins of SCF are selected in a tissue-dependent manner. This finding is in accordance with experimental studies [30, 117, 121]. On the one hand, it has been shown that in human cells multiple variants of the SCF complex exist, each one containing a different F-box protein for substrate recognition. On the other hand, brain and blood cells have been identified as tissues of major expression for some F-box components, and expression variation of F-box components has been observed in several tissues such as testis, prostate, and placenta. In our results, all detected module variants are active in natural killer (nk) cells, a specific type of white blood cell that plays an important role in immune response [98], whereas placenta, prostate, testis, uterus, or certain brain regions contain only one or two variants. As illustrated by the above examples, DME integrated with profile data can be a powerful tool to reveal functional and condition-dependent variants of protein complexes.

9.7 Disease-Related Module Analysis

Furthermore, DME predictions of a larger scale were performed on the human interaction network, with the goal to offer a comprehensive set of modules for disease-related analyses [48]. As an increased coverage of proteins generally reduces the module accuracy, module sets at different levels of granularity were made available. For that, we started with the total set of human interactions collected from the databases mentioned in Section 9.1, with assigned weights as described there. Then, we chose different quantiles of the weight distribution as cutoff thresholds to remove interactions with lower weight, and computed modules for different density thresholds on the remaining network (ignoring the edge weights). The results of three different settings are stored in a public web repository called DICS[17], augmented with additional information like orthology [163] and expression profiles [202], disease annotation according to HGMD [42], as well as references to the CORUM [182], DrugBank [226], KEGG [164], and Reactome [107] databases. The default module set covers 40% of the disease genes listed in the HGMD database [199], whereas the reference complexes of the CORUM database [182] contain only 11% of them.[18]

In addition, the DICS server allows for interactive exploration of gene lists derived from high-throughput experiments. Given a set of genes as query, it returns significantly enriched modules and known complexes, thereby providing information about putative or confirmed functional relationships among the gene products. For example, the analysis of genes that are differentially expressed in toxic oil syndrome patients [177] revealed potential links to other diseases, see [48] for details.

---

[17] `http://mips.helmholtz-muenchen.de/proj/dics/`

[18] state of November 2008

9.8 Discussion and Outlook

The dense module enumeration algorithm can assist in the systematic analysis of weighted interaction networks. By explicitly considering the module density as a search criterion and returning all solutions that satisfy this criterion, it is a complementary approach to the widely used graph partitioning methods [186]. Beside the completeness guarantee, a strength of the method lies in the possibility of transparent data integration, which is of crucial importance in systems biology applications. We illustrated some application scenarios in the context of protein complex prediction. While many known functional complexes were successfully reproduced, there is a lot of potential for further improvements. For instance, carefully designed probabilistic models for the edge weights would increase the attractiveness of enumeration-based module results. Moreover, as discussed in Section 5.5, we can choose stricter module criteria to make the approach more robust against noisy extensions of dense core modules.

Also, the consistency requirements (Section 5.7.1) are sensitive to noise in the data. According to our definition, they do not allow for exceptions, that means, the whole set of module proteins has to fulfill the specified conditions. This has the advantage of easy interpretability, e.g., one obtains only the part of a complex that is conserved in several species (with respect to a given orthology profile); on the other hand, some true complex members might be missed because of incomplete or incorrect profiles. As a possible extension, one could allow to include proteins that have unknown values for some of the shared profile conditions or consider inexact matches of subprofiles. In the case of continuous values, our current approach uses predefined thresholds for discretization; while this again facilitates the interpretation of the results, inappropriate choices can mislead the analysis; more flexible criteria that directly look at similarities of real-valued profiles are conceivable (see also Section 5.7.1).

So far, the consistency constraints consider profile information on the nodes. For example, in the case of orthology profiles, we take the evolutionary conservation of proteins into account while searching for densely interacting protein sets in a species. However, although it is sometimes possible to infer interactions based on sequence homology [33], the existence of orthologs does not guarantee that also the interactions between them are maintained in other species. Therefore, if interaction data are available for multiple species of interest, it is useful to coanalyze them. That means, rather than node conservation profiles, conservation profiles of interactions or densely interacting modules are considered. Recently, comparative interactomics has become a very active research field [33, 115, 142, 169, 238]. The STRING database [220] provides protein-protein interactions for several hundred organisms; it collects and ranks information from different sources, including high-throughput experiments, literature mining, and automatic predictions. One major challenge with multi-species interactomics approaches is that there does not necessarily exist a one-to-one correspondence between proteins of different species; rather, it can be a many-to-many relationship, which can be defined based on BLAST sequence similarity scores [191]. A valuable resource in that area is the COGs database (Clusters of Orthologous Groups of proteins) [208, 209], which contains orthology mappings for multiple species.

Given the orthologous relationships between proteins of different species, there are several ways to exploit them for the module search in comparative interactomics. The first option is to build a so-called network alignment graph [191, 192], where each node contains a set of orthologous proteins, one from each species; interaction weights are derived by integrating the interaction contributions of the individual species; then, conserved modules are identified as dense subgraphs. It is potentially promising to apply the DME algorithm for this task, replacing the heuristic search approach that was proposed originally. However, preliminary studies on interaction data from *S. cerevisiae* and *H. pylori* (the same species as considered in [191]) revealed that the heuristic method successfully detected the most relevant patterns; modules unique to DME were either very small or partly overlapped with other modules. More extensive comparisons of the different approaches would be desirable for future work. If the number of considered species increases, the network alignment approach becomes intractable because of the combinatorial explosion regarding the nodes in the graph. However, additional constraints from the phylogeny and an appropriate graph representation can help to solve this problem [108]. Alternatively, one can coarsen the analysis by introducing only one node for each group of orthologous proteins (COG). To define the interaction weight between two COG nodes with respect to a certain species, one could take for instance the maximum or average interaction weight across all protein pairs of the respective species where the first protein belongs to the first COG and the second protein belongs to the second COG.

Remarkably, our multi-way cluster detection approach (see Chapter 6) offers the opportunity to coanalyze interaction data from multiple species without summarizing the interaction weights beforehand. This allows to detect module patterns that cooccur in a subset of species. An example application of this algorithm using multiple networks of one species is presented in the next section.

10 Module Detection from Multiple Coexpression Networks

Integration of multiple data sources is of great importance in systems biology studies. Here, we consider the special task of coanalyzing multiple networks. In the following case study, we searched for common modules in gene coexpression networks stemming from different experiments. For that purpose, we used a data collection of yeast gene expression measurements. As all measurements refer to the same set of gene probes, the node mapping between different networks is trivial. The results described here are published in [68, 69].

10.1 Data

We took the gene expression dataset from [61] and preprocessed it in a similar way as described in [91]: after selecting the experiments with at least 6 individual measurements, we calculated for each of them the pairwise correlation coefficients regarding the expression profiles of all genes; if the correlation was positive and had a p-value below 10^{-5}, we connected the corresponding genes by an edge (of weight 1).[1] This resulted in 17 different coexpression networks on the same set of genes, each of which contained 9237 edges on average. These data can be represented as a three-dimensional tensor with the genes in the first two dimensions and the identifier of the experiment (the network) in the third dimension. As the networks are undirected, the tensor is symmetric with respect to the first two dimensions. Our goal was to analyze the set of networks for cooccurring dense substructures. For that purpose, we applied the DCE algorithm explained in Chapter 6.

---

[1] Correlation coefficients and p-values were calculated using the `corrcoef` function of MATLAB; the p-values are based on a t-statistic.

10.2 Related Approaches

There exist several competitive approaches to solve this task. The Cocain method (COC) [242] detects all frequent closed γ-quasi-cliques. A quasi-clique is a set of nodes U such that each of them has edges to at least $\lceil \gamma(|U|-1) \rceil$ other nodes in U (see Section 5.5). A set of nodes U is a frequent γ-quasi-clique if it is a γ-quasi-clique in at least minsup networks, where minsup is a natural number. This criterion is stricter than our cluster density criterion (Definition 23), even if we require balance constraints (Section 6.4.1); in fact, each frequent quasi-clique is a balanced dense cluster, but not vice versa. Like our local maximality criterion defined in Section 5.4, the closeness requirement aims at reducing redundancy in the results; it discards quasi-cliques if they are included in another solution with at least the same support among the networks.

Another approach to coanalyze multiple networks is the Codense algorithm (COD) [91]. It aims at detecting dense subnetworks where the edges have similar occurrence profiles across the whole set of networks. That means, in contrast to DCE and COC, it does not only require a (local) cooccurrence of dense subnetworks in a subset of the given networks, but a global correlation of the participating edges. For this, it first compiles edges with frequency $\geq$ minsup into a summary network, from which dense subgraphs are extracted; these subgraphs are then further analyzed with respect to the correlation of edges across all given networks. The dense cluster detection is based on a non-enumerative network partitioning strategy. The density criterion is the same as for DCE, but it is applied on the summary network. However, each dense subtensor consisting of frequent edges has a corresponding dense cluster in the summary network.

Finally, relational data mining approaches [103] are equivalent to DCE with density threshold 100%, so we do not consider them separately in our evaluation. Also note that the local maximality criterion (Section 6.4.1) yields in that case the set of maximal clusters, because any subcluster of a solution satisfies the density threshold.

10.3 Experimental Set-Up

We compared the different approaches on the coexpression networks described above. We obtained the COC code from the original authors [242], and COD was downloaded from `http://zhoulab.usc.edu/CODENSE/`. The minimum edge frequency threshold in COD was set to 3. This yielded a summary network with 1444 edges involving 411 nodes. For comparison purposes, we restricted the 17 individual networks considered in DCE and COC to the same set of edges, and set the minimum network support of clusters (i.e., the minimum number of instances in the third dimension) to 3. Furthermore, COD requires a p-value threshold for the similarity of occurrence profiles, which was set to 0.01. The minimum number of genes per cluster was set to 6 in all approaches.

10.4 Evaluation Measures

To evaluate clusters of genes, we performed a functional enrichment analysis with respect to the Gene Ontology annotation [13], using the Expander tool [190] with default parameters; this yielded functional categories that were significantly overrepresented in a predicted cluster, having p-values below 0.05 after correction for multiple testing. In addition to the number of functionally enriched clusters, we report the average genewise reliability, the average pairwise reliability, as well as the overall precision and recall. These measures were determined as follows. Given a cluster with one or several significantly enriched functional categories, genes that belong to the same enriched category are called *homogeneous*. Let hg_i be the size of the largest group of homogeneous genes in cluster i, and let g_i be the total number of genes in the cluster. Then, the genewise reliability of the cluster is given by

$$\frac{\mathrm{hg}_i}{g_i}\,. \tag{10.1}$$

Further, let hgp_i be the number of homogeneous gene pairs and gp_i the total number of gene pairs. The pairwise reliability of the cluster is defined as

$$\frac{\mathrm{hgp}_i}{\mathrm{gp}_i}\,. \tag{10.2}$$

Compared to the genewise reliability, this measure takes into account all different enriched categories of a cluster. It can be seen as the probability that an arbitrary gene pair taken from the cluster is homogeneous. For each of the two reliability measures, we determine the average across all clusters, weighted by the cluster size. Finally, the precision and recall measures refer to unique (homogeneous) gene pairs across all clusters. That means, each gene pair is only counted once even if it occurs in more than one predicted cluster. Note that all methods applied in this comparison predict overlapping clusters. In analogy to Section 9.2, precision

Table 10.1: Comparative evaluation on coexpression data. Abbreviations: max. (maximum), avg. (average), rel. (reliability), bal. (balanced). The parameter k refers to the optional branching restriction of DCE. See text for details.

	Density (%)	No. of clusters	No. of enriched clusters	Max. no. of genes	Avg. no. of genes	Gene-wise rel. (%)	Pair-wise rel. (%)	Precision (%)	No. of recalled pairs	Time (s)
DCE	100	53	52	9	6.7	95.2	92.6	84.3	215	2.9
	95	239	238	11	7.8	95.9	93.1	84.2	388	5.4
	90	1057	1048	13	8.6	95.6	92.9	81.7	642	25.7
	85	3269	3240	16	10.7	96.3	94.1	82.6	1041	179.2
	80	16982	n/a	18	11.8	n/a	n/a	n/a	n/a	2245.0
	75	95869	n/a	20	13.9	n/a	n/a	n/a	n/a	30011.2
DCE (bal.)	100	53	52	9	6.7	95.2	92.6	84.3	215	2.9
	95	425	416	9	6.5	96.3	94.9	83.3	219	5.4
	90	1288	1277	11	6.6	97.5	96.0	81.9	303	25.6
	85	3705	3684	11	7.0	98.0	96.9	82.6	409	179.7
	80	10697	n/a	13	7.2	n/a	n/a	n/a	n/a	2271.8
	75	24200	n/a	14	8.3	n/a	n/a	n/a	n/a	29968.1
DCE (bal., $k = 1$)	100	17	16	9	6.7	92.1	90.4	83.6	117	0.6
	95	17	16	9	6.8	91.4	88.8	82.5	118	0.6
	90	28	27	11	6.8	92.1	88.6	81.4	162	0.7
	85	38	37	11	7.0	94.0	91.4	82.9	194	0.7
	80	66	64	12	7.1	93.8	91.0	80.9	284	0.9
	75	71	69	14	8.0	94.9	92.9	82.2	332	1.0
DCE (bal., $k = 2$)	100	133	130	9	6.7	95.7	94.0	83.0	176	1.1
	95	136	133	9	6.8	95.8	94.2	83.3	185	1.4
	90	296	291	11	6.9	95.9	94.1	83.9	260	2.3
	85	590	584	11	7.4	97.3	96.1	82.6	338	4.2
	80	1247	1237	13	7.7	97.1	95.4	81.9	456	9.8
	75	2198	2192	14	8.9	97.7	96.0	82.8	521	23.5
COC	100	53	52	9	6.7	95.2	92.6	84.3	215	1.3
	95	53	52	9	6.7	95.2	92.6	84.3	215	1.3
	90	53	52	9	6.7	95.2	92.6	84.3	215	2.3
	85	109	108	10	8.2	97.2	95.4	85.0	260	7.2
	80	200	199	12	7.6	96.3	93.4	83.3	329	14.0
	75	520	512	13	8.2	95.7	93.2	82.9	474	54.2
COD	100	0	-	-	-	-	-	-	-	0.2
	95	3	3	11	9.7	100.0	100.0	100.0	80	1.6
	90	10	9	11	7.5	90.7	91.7	83.6	107	1.5
	85	9	8	10	7.9	84.5	81.3	76.1	140	1.6
	80	10	9	18	8.9	85.4	82.8	79.0	245	2.0
	75	8	7	21	11.2	85.6	84.3	80.9	314	1.6

is given by the number of homogeneous pairs relative to the number of all within-cluster pairs; the recall values given in the table correspond to absolute numbers of homogeneous pairs.

10.5 Results

Table 10.1 summarizes the results of DCE, COC, and COD for different density thresholds. For DCE, we also list the results with balance constraints (bal.) and with branching restrictions ($k = 1, 2$) (see Section 6.4.4). For these constrained DCE versions, the local maximality of clusters cannot be checked efficiently, due to similar reasons as discussed in Section 5.5 in the context of the DME algorithm. Instead, they return all clusters at leaf nodes of the search trees, which can increase the number of clusters compared with the unconstrained DCE runs.

For 100% density, DCE, DCE (bal.), and COC are all equivalent to the relational data mining setting and therefore yielded the same results. However, for lower density values DCE and DCE (bal.) are more flexible than the quasi-clique approach used by COC, so they achieved much higher recall, while precision and reliability remained in a comparable range. Interestingly, both for DCE and COC, the average cluster reliability with density threshold 85% was larger than with density threshold 100%. This can be explained by the fact that, at sufficiently high density levels, larger clusters are more likely to be biologically significant than small ones (note that the average number of genes per cluster increased). On the other hand, a decreasing density threshold allows the clusters to include genes that are less related. Therefore, the overall pairwise precision of DCE was slightly reduced when going from 100% to 85% density. In contrast, COC, which applies the more rigid quasi-clique criterion, kept the precision level. The edge correlation criterion required by COD is quite restrictive and its search method is not exhaustive, so the recall was lower for COD. However, while the precision and reliability values were perfect for a density threshold of 95%, they were considerably below the other approaches at 85% density.

For density thresholds below 85%, the number of solutions returned by DCE increased drastically, which came along with an exponential increase of the runtime (the measurements were performed on a 2.8 GHz processor). The reason for that is the increasing flexibility of patterns, which leads to strongly overlapping solutions. The generation of disconnected modules played only a minor role: the runtime and the total number of solutions dropped by about 10% when introducing a heuristic rule that prunes at the first occurrence of an isolated instance, which completely avoids the generation of disconnected modules and might in addition lead to the loss of connected solutions (Section 6.4.3). The balance criterion reduced the number and size of modules, but still the result set was much too large to be suitable for human inspection (also, the Expander tool for enrichment analysis failed); therefore, further criteria to restrict the search are needed. For comparison purposes, we again used the heuristic branching restriction introduced in Section 6.4.4 with values 1 and 2. With this, the performance was competitive with COC and COD. The branching restriction produced lower recall than the complete search (considering balanced clusters in both cases), but it could still compete with the recall values achieved by COC and COD. Furthermore, although our cluster criterion is less restrictive than the criteria for COC or COD, the clusters were biologically meaningful, achieving similar levels of reliability and precision. Beside that, DCE is applicable to more general settings, namely data with an arbitrary number of dimensions, binary or weighted values, including symmetries or not. As discussed in Sections 5.5 and 6.4.1, an interesting question for future research would be whether some techniques from quasi-clique mining can be generalized to these settings and how to combine ideas from DCE and quasi-clique mining to achieve the most efficient search strategies for enumerative pattern discovery in different types of data.

11 Biclustering of Gene Expression Data

Gene expression data are very helpful in analyzing relationships between genes, for instance in the form of coexpression links as described in the previous chapter. On the other hand, they can reveal similarities between different samples and cellular conditions. One common approach is to investigate groups of genes and groups of samples simultaneously, by applying biclustering techniques. The motivation behind this is that samples might exhibit local similarity with respect to a subset of genes and vice versa. In Section 6.6, we used a small-scale biclustering task with discrete gene signatures derived from multiple gene expression experiments to study the runtime behavior of different enumeration strategies. Here, we analyze a large gene expression dataset using the hierarchical biclustering approach described in Chapter 7.3.

11.1 Data

We downloaded the microarray data by Schmid *et al.* [187][1], which contain large-scale gene expression measurements for the plant *Arabidopsis thaliana*, covering 22 746 gene probes for 237 samples. The samples represent 79 different conditions, each of them being measured by three replicates. The conditions are labeled according to 8 major tissues, see Table 11.1. As the ground expression level may vary between different genes, we normalized the expression profiles for each gene by the median across all samples and then transformed the values by $\log_{10}$; this yielded relative expression changes in the range from -3.34 to 3.76.

---

[1] Available at `http://www.weigelworld.org/resources/microarray/AtGenExpress/`.

Table 11.1: Major tissues and their frequency among the 79 experiments.

Stem	3
Root	7
Seeds	8
Floral organs	10
Apex	11
Whole plant	11
Flowers	12
Leaf	17

Table 11.2: Biclustering versus global sample clustering results for different numbers of sample clusters c using average linkage (avg.) or complete linkage (compl.). See text for details.

Biclustering approach	$c = 50$		$c = 20$		$c = 10$		$c = 8$	
	avg.	compl.	avg.	compl.	avg.	compl.	avg.	compl.
Adjusted Rand index	0.38	0.41	0.53	0.34	0.57	0.28	0.47	0.25
Statistics for non-base clusters:								
No. of clusters	25	26	20	20	10	10	8	8
Maximum no. of samples	44	38	75	71	84	90	84	98
Minimum no. of samples	1	1	1	2	3	3	3	6
Average no. of samples	8	8	12	12	24	24	30	30
Statistics for larger clusters:								
No. of clusters	6	2	13	11	9	9	7	8
Average no. of samples	12	12	17	19	26	26	33	30
Average purity (%)	92	88	78	65	88	60	76	59
No. of 100% pure clusters	4	1	6	5	3	1	3	0
Average completeness (%)	92	100	96	83	100	84	100	86
No. of 100% complete clusters	5	2	11	5	9	3	7	3

Global approach	$c = 50$		$c = 20$		$c = 10$		$c = 8$	
	avg.	compl.	avg.	compl.	avg.	compl.	avg.	compl.
Adjusted Rand index	0.33	0.32	0.58	0.47	0.53	0.50	0.53	0.57
Statistics for non-base clusters:								
No. of clusters	50	50	20	20	10	10	8	8
Maximum no. of samples	27	27	51	33	66	39	66	60
Minimum no. of samples	1	3	3	3	3	9	9	9
Average no. of samples	5	5	12	12	24	24	30	30
Statistics for larger clusters:								
No. of clusters	13	15	15	17	8	10	8	8
Average no. of samples	10	9	15	13	29	24	30	30
Average purity (%)	98	100	89	86	69	75	67	75
No. of 100% pure clusters	12	15	10	9	2	1	1	0
Average completeness (%)	98	100	100	100	100	100	100	100
No. of 100% complete clusters	12	15	15	17	8	10	8	8

11.2 Sample-Based Evaluation

The preprocessed data were analyzed with the hierarchical biclustering approach from Chapter 7.3, stopping at different numbers of clusters; in all cases, the total computation with our C++ implementation took between 7 and 8 seconds on a 2.67 GHz processor. To define the stopping criterion more precisely, let the term "sample clusters" refer to biclusters that contain sample indices. The process is halted when the next merge would reduce the number of sample clusters such that it falls below a predefined value, i.e., base biclusters that contain only gene indices may still be added to other biclusters even if the critical threshold of sample clusters is already reached. The thus obtained clustering of samples was compared to the tissue-based partitioning of samples (cf. Table 11.1) using the adjusted Rand index [94], a very common evaluation measure (e.g., [185, 237]). It is defined by determining for each partitioning of instances the instance pairs in the same cluster and the instance pairs in different clusters and computing the overlaps of these sets between the two partitionings; the measure is adjusted such that perfectly matching partitionings obtain the maximum value of 1 and the expected value under the hypergeometric distribution is 0. Formally, the adjusted Rand index can be expressed as follows [224]:

$$\frac{2(AD - BC)}{(A+B)(B+D)+(A+C)(C+D)} \tag{11.1}$$

Here, A, B, C, and D denote numbers of pairs according to the following contingency table:

	Second partition	
First partition	Pair in the same cluster	Pair in different clusters
Pair in the same cluster	A	B
Pair in different clusters	C	D

The upper part of Table 11.2 shows the adjusted Rand index for different numbers of sample clusters. The ground truth contains 8 clusters (Table 11.1). In addition to the average linkage criterion, we tried the very strict complete linkage criterion to compute bicluster associations. In all cases, the adjusted Rand index was much greater than 0, indicating that the found clusters at least partly matched the tissue-based structure of the data. However, for cluster numbers below 50, complete linkage performed considerably worse than average linkage. One possible reason for that is the increased probability of outlier values when the clusters grow; the complete linkage criterion will prefer merges with the least extreme outliers, whereas average linkage will focus on merges with the greatest average. Application of the single linkage approach is generally considered to be problematic because a large association value of a single instance pair is sufficient to merge two clusters, which can cause chaining effects [152]; here, it led to degenerate clusterings composed of one large cluster and a set of singleton clusters.

For comparison, we computed sample partitionings via conventional hierarchical clustering. The similarities between samples were determined as the correlation of the profiles across all gene probes; this contrasts with our approach, where clustering of samples emerges through shared sets of upregulated genes. Again, we reported the results for certain numbers of sample clusters obtained by average linkage or complete linkage, see Table 11.2 (lower part). The adjusted Rand index was similar for both linkage types, so the outlier problem described above is less severe in the global similarity approach than in our local similarity approach. Regarding average linkage, the two approaches had similar performances across the considered settings: each of them won twice (once for a large cluster number c, and once for a small c), and their adjusted Rand index values differed by at most 0.06. The table lists some further statistics to describe the results. Base biclusters were only present for $c = 50$; otherwise, all samples were covered by true biclusters with non-empty sets in both dimensions of the matrix. The maximum number of samples per cluster was more moderate in the global approach, i.e., the cluster sizes had smaller variance than in the biclustering approach.

Furthermore, we evaluated the purity of individual sample clusters with respect to the tissue label as well as their completeness, which corresponds to the relative number of different conditions that occur with all three replicates in the cluster. As small clusters are likely to contain just the replicates of one condition, which have trivially the same tissue label, we restricted this analysis to clusters with at least four samples; biclusters were additionally required to include at least four genes. Averages of purity and completeness were computed across all selected clusters, weighting the individual terms according to the number of samples. In addition, the table shows the number of 100% pure and the number of 100% complete clusters. Again, we focus our discussion on the average linkage results. While the global sample clustering produced better purity values for large values of the cluster number c, our method achieved higher purity for small values of c (which produce larger clusters). Both approaches tended to include sample replicates in the same cluster, which is a sanity check for their usefulness. We also applied another biclustering method, SAMBA [206], which is commonly used in gene expression analysis. It does not allow to specify the number of clusters; instead, it has a parameter h

Table 11.3: Sample-based evaluation of SAMBA biclustering results. See text for details.

SAMBA approach	$h = 3$	$h = 4$	$h = 5$	$h = 6$	$h = 7$
Statistics for non-base clusters:					
No. of clusters	108	30	26	25	23
Maximum no. of samples	19	25	27	37	29
Minimum no. of samples	3	4	5	6	6
Average no. of samples	3	9	10	11	11
Statistics for larger clusters:					
No. of clusters	8	30	26	25	23
Average no. of samples	7	9	10	11	11
Average purity (%)	93	79	81	80	78
No. of 100% pure clusters	6	9	8	7	6
Average completeness (%)	3	13	18	21	18
No. of 100% complete clusters	0	0	0	0	0

that specifies sizes of sample sets considered for identifying seed biclusters. The implementation of SAMBA is available as a part of the Expander tool [190]; it takes only values up to 7 for h because the employed sample subset enumeration is costly. SAMBA discovers overlapping biclusters, so it does not produce a unique partitioning of samples. The average purity of larger clusters was similar to the hierarchical approaches, but replicates of the same condition were often separated, which is probably partly caused by the fact that the sample clusters are smaller on average (see Table 11.3).

11.3 Gene Function Analysis

Compared with hierarchical sample clustering, the main conceptual advantage of biclustering is that it yields for each sample cluster a set of characteristic genes. To evaluate the predicted biclusters with respect to genes, we performed a functional enrichment analysis using the Expander tool [190]; the results are summarized in Table 11.4. In particular, we determined for each setting the number of clusters that were enriched with at least one GO term; furthermore, we computed the gene-based reliability of individual clusters (i.e., the largest occurrence frequency of an enriched term among all cluster genes, see Section 10.4), as well as the number of enriched terms, and took the average across all clusters; as in the previous section, we considered only biclusters with at least four samples and at least four genes. To define the set of enriched GO terms, we tried three common cutoffs for the empirical p-values. However, the results were quite robust regarding the choice of the threshold.

Table 11.4: Gene function evaluation of biclusters obtained with our approach and with SAMBA.

Hierarchical biclustering	$c = 50$		$c = 20$		$c = 10$		$c = 8$	
	avg.	compl.	avg.	compl.	avg.	compl.	avg.	compl.
No. of non-base biclusters	25	26	20	20	10	10	8	8
Maximum no. of genes	288	267	987	6476	5363	6476	5983	6792
Minimum no. genes	1	1	5	9	483	136	604	1115
Average no. of genes	26	20	163	1140	1894	2281	2639	2851
Statistics for larger clusters:								
No. of clusters	6	2	13	11	9	9	7	8
Average no. of genes	25	19	212	465	2051	1815	2930	2851
No. of enriched clusters ($p = 0.001$)	1	1	5	5	9	7	7	7
No. of enriched clusters ($p = 0.01$)	1	1	9	8	9	9	7	8
No. of enriched clusters ($p = 0.05$)	2	1	10	8	9	9	7	8
Average reliability (%) ($p = 0.001$)	15	12	12	16	14	16	13	16
Average reliability (%) ($p = 0.01$)	15	18	14	15	14	16	14	13
Average reliability (%) ($p = 0.05$)	15	18	14	16	14	16	14	13
Average no. of terms ($p = 0.001$)	3	1	7	8	19	24	24	25
Average no. of terms ($p = 0.01$)	3	3	10	9	21	36	29	27
Average no. of terms ($p = 0.05$)	3	3	12	11	25	42	33	32

SAMBA biclustering	$h = 3$	$h = 4$	$h = 5$	$h = 6$	$h = 7$
No. of non-base biclusters	108	30	26	25	23
Maximum no. of genes	1627	1987	1834	1517	1474
Minimum no. genes	713	240	703	601	467
Average no. of genes	1144	1145	1131	1052	991
Statistics for larger clusters:					
No. of clusters	8	30	26	25	23
Average no. of genes	1223	1145	1131	1052	991
No. of enriched clusters ($p = 0.001$)	8	28	26	25	23
No. of enriched clusters ($p = 0.01$)	8	29	26	25	23
No. of enriched clusters ($p = 0.05$)	8	30	26	25	23
Average reliability (%) ($p = 0.001$)	19	18	18	15	15
Average reliability (%) ($p = 0.01$)	19	18	18	16	16
Average reliability (%) ($p = 0.05$)	19	17	18	16	16
Average no. of terms ($p = 0.001$)	18	16	12	13	10
Average no. of terms ($p = 0.01$)	19	19	19	19	17
Average no. of terms ($p = 0.05$)	22	20	21	22	20

Most biclusters containing large gene sets were significantly enriched with at least one function; in particular, all predicted biclusters for $c = 10$ and $c = 8$ in the hierarchical approach and all biclusters from the SAMBA approach were enriched; in all cases, the number of genes was greater than 100. In both approaches, the gene-based reliability never exceeded 20%, and a bicluster contained on average a few tens of enriched functional categories; this indicates that each bicluster represents a set of active biological processes rather than one specific biological process. To investigate whether the assigned functional categories match the tissue annotation of the biclusters, we examined the biclusters that were 100% pure with respect to the tissue label. Table 11.5 lists the most significant GO categories for three pure biclusters generated by average linkage for $c = 20$ (this setting produced the largest number of pure biclusters, see Table 11.2; the shown biclusters have the best enrichment p-values, namely below 0.001). For a bicluster of seed samples, we got GO terms related to embryonic and organismal development; a floral organ bicluster was enriched with membrane transport and enzyme activity functions; finally, a bicluster composed of stem samples yielded cell wall biogenesis and secondary cell-wall biogenesis as overrepresented categories.

To conclude, our hierarchical biclustering method returned biologically meaningful results despite employing a very simple procedure. Being computationally cheap, it is suitable for analyzing large datasets. The obtained biclusters can serve as a basis for further data exploration; for instance, it would be interesting to search for motifs in the promoter regions of the discovered gene sets in order to reveal tissue-specific gene regulation mechanisms.

Table 11.5: Enriched GO terms for 100% pure tissue clusters ($p = 0.001$). For each term, we show the number of cluster genes belonging to that category.

Seeds (samples: 15)	genes: 438
Seed development - GO:0048316	28
Multicellular organismal development - GO:0007275	49
Embryonic development - GO:0009790	24

Floral organs (samples: 6)	genes: 987
Protein amino acid phosphorylation - GO:0006468	82
Transport - GO:0006810	117
Kinase activity - GO:0016301	100
Intrinsic to membrane - GO:0031224	65
Active transmembrane transporter activity - GO:0022804	49
Substrate-specific transporter activity - GO:0022892	66
Transmembrane transporter activity - GO:0022857	70
Secondary active transmembrane transporter activity - GO:0015291	32
Hydrolase activity hydrolyzing O-glycosyl compounds - GO:0004553	38

Stem (samples: 6)	genes: 116
Cellulose and pectin-containing secondary cell wall biogenesis - GO:0009834	10
Cell wall organization and biogenesis - GO:0007047	14

12 SNP-Transcript Association Discovery

Beside the classical application field of gene expression analysis, biclustering approaches are useful in various tasks related to systems biology. In this chapter, we consider bicluster discovery in a context where information on sequence variation is brought together with gene expression data from human brain. The goal was to find groups of single nucleotide polymorphisms that are associated with the expression behavior of a set of genes; on the side of the polymorphisms, additional constraints have to be taken into account. To tackle this problem, we applied both biclustering approaches presented in this work: the enumerative approach from Chapter 6 and the hierarchical approach from Chapter 7. The results were evaluated with respect to functional annotation of SNPs.

12.1 Motivation

One source of genomic variation in the human population are single nucleotide polymorphisms (SNPs), i.e., single positions in the DNA sequence where at least one percent of the population exhibit an alternative nucleotide. While large-scale SNP genotyping of individuals has recently become possible and yields profiles with millions of SNPs [210], the functional roles of most SNPs are still unknown. An approach to close this gap are association studies [131, 157, 200, 201]. By determining both genotypic and phenotypic properties of a large set of individuals, they create a basis to statistically infer relationships between these properties; one common scenario is to compute associations between SNPs and transcript abundance (i.e., expression levels of genes). This is depicted schematically in Figure 12.1. The nucleotide variants of a SNP are called alleles; typically, there exist two alternative nucleotides; the one that occurs more frequently in the population is called *major allele* (denoted by A), the other one *minor allele* (denoted by a). As human cells contain two complete genome sequences, one from the mother and one from the father, the genotype at the SNP locus (genome position) is given by a pair of alleles; the alleles cannot be traced back regarding their genomic sequence membership, so

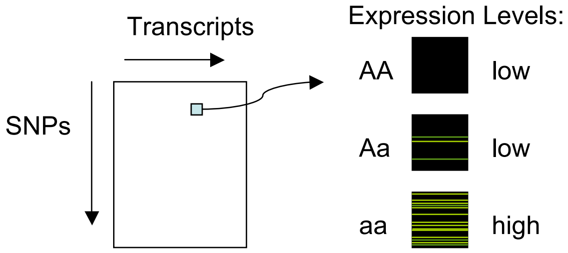


Figure 12.1: Illustration of a SNP-transcript association.

we distinguish three possible genotype states: “AA”, “Aa”, and “aa” The individuals can be grouped according to their genotype, and by comparing the three groups with respect to the expression levels of a certain gene, we can measure to what extent the transcript abundance is influenced by the genotype. Such an analysis can assist in deciphering potential functions of SNPs. The problem is computationally challenging because regulatory relationships can even exist between distant parts of the genome [157].

The goal of this study was to detect significant association patterns involving several SNPs and genes at the same time. More precisely, we searched for bicluster patterns in the SNP-transcript association matrix, following the basic workflow applied in [131] for the analysis of expression regulation in breast cancer. As the focus lies on effects that are shared across several SNPs or transcripts, the bicluster analysis is a useful extension of pairwise association analysis for deciphering SNP functions. In particular, it can also reveal relationships among SNPs and among genes, and individually weak associations might become significant if they are supported by several genes and several SNPs. However, artifacts may arise from SNP loci that are in linkage disequilibrium (LD). Such SNPs tend to be inherited together, for example due to genetic linkage hindering recombination or due to the population structure. Consequently, SNPs that are in LD do not add evidence to a certain association pattern because they behave similarly irrespective of the existence of functional relationships. To deal with this, we introduced a constraint forbidding that SNPs with large LD appear together in a pattern, i.e., we set a threshold for the maximum LD of SNP pairs within a bicluster. Alternatively, one could cluster the SNPs beforehand according to their pairwise LD values and choose one representative SNP out of each cluster to perform the further analysis. However, this leads to a loss of information, whereas in our approach all SNPs are kept so that the most appropriate SNP can be selected for each association pattern.

12.2 Data and Preprocessing

For our experiments, we used a subset of the compendium by Myers *et al.* [157], which contains SNP profiles and brain expression data of 193 human individuals. To focus the analysis, we obtained a selection of brain-related genes and SNPs located in the corresponding genomic regions by personal communication from M. Specht (Max Planck Institute of Psychiatry, D-80804 Munich); in total, there were 1521 SNPs, and 216 genes that were covered in the expression dataset. From those, we removed SNPs and genes having more than 5% missing values among the 193 samples. Then, we eliminated extreme outlier values from the gene expression matrix by considering each gene separately and iteratively applying the Shapiro test (in R); it computes the probability that the data (here, the expression values across the individuals) come from a normal distribution. As long as the p-value was below a threshold of 0.0001, we recursively selected the most extreme value and set it to NA. This data cleaning step was necessary because otherwise the outliers distorted the association analysis; typically, they were also strongly linked to some covariate (e.g., the source of the data samples), thereby obscuring biological effects.

As in [131], we computed a SNP-gene association matrix. For that, we calculated ANOVA-based association p-values between each SNP and each gene; this was done using the WG-Permer software by S. Ripke (Max Planck Institute of Psychiatry, D-80804 Munich), which is available at `http://www.wg-permer.org`; the parameters were set to standard values: minimum SNP call rate 0.1, threshold for Hardy-Weinberg equilibrium test 10^{-5}, minor allele frequency 0.05. Among the considered SNPs, 1074 turned out to have association p-values below 0.05 to at least one of the 134 selected genes. We took the -log10 values of those as entries in the association matrix; the remaining entries were set to zero. To smooth the distribution of non-zero entries, we calculated the 0.95-quantile q and set entries above q to q, i.e., strong outliers were set to the maximum of the remaining entries. Finally, pairwise LD correlation coefficients for SNPs were determined using the R genetics package.

12.3 Experimental Approach

To analyze the SNP-gene association matrix, we applied two different techniques for bicluster detection. First, we used the enumerative strategy explained in Chapter 6, which finds all submatrices where the average value exceeds a threshold $\theta \cdot w_{\max}$, where $w_{\max}$ is the maximum entry in the SNP-gene association matrix. To exclude undesired artifacts arising from LD, only SNPs with a pairwise LD correlation below 0.5 were permitted to belong to the same bicluster. The integration of this constraint into the enumeration scheme was done as described in Section 5.7. For the threshold θ, we tried the values $\{1.00, 0.98, \ldots, 0.90\}$. As the number of solutions increased drastically in the last setting, we chose the results with $\theta = 0.92$ for further analysis; only biclusters including at least two SNPs and at least two genes were reported. As a second method, we employed the average linkage hierarchical biclustering approach from Chapter 7, which iteratively combines biclusters to larger ones. To account for the pairwise LD constraint, we deleted after a merge some newly added SNPs from the bicluster if the maximum threshold of 0.5 was violated. The hierarchical process was stopped when the best linkage value sank below $r \cdot w_{\max}$. We tested r in the range from 0.9 to 0.3 using decrements of 0.1. For $r = 0.5$, the biclusters covered a similar number of SNPs as the enumerative approach (again focusing on biclusters with at least two SNPs and at least two genes); therefore, we used this setting in our comparative evaluation.

12.4 Results

We were interested in whether bicluster analysis helps to detect SNPs that have functional roles. There are many possible ways to define functional SNPs, and it is difficult to create gold standard sets. In our evaluation, we considered a SNP to be functional if it lies in a functional genomic region. Such regions are for instance exons (i.e., the parts of a gene that appear in the mature mRNA, in contrast to introns, which are removed from the primary transcript by splicing) or sequences of functional RNAs. For that purpose, we downloaded the exon annotation from the UCSC Genome Browser [109] and several annotation tracks from the UCSC Genome Browser for Functional RNA [116], more specifically RNAdb, NONCODE, partially intronic RNAs (pin RNAs), and totally intronic RNAs (tin RNAs); in addition, we included ultraconserved regions, transposon-free regions (transp. free), and indel-conserved regions with $p < 0.001$ because strong conservation of a region is an indicator for functional relevance.[1]

Figure 12.2 shows the number of functional SNPs plotted against the total number of SNPs ranked according to their occurrence in biclusters. For the enumerative approach (Enum. Bicl.), the biclusters were sorted by their count-based p-values

---

[1] The datasets were downloaded in February 2008.

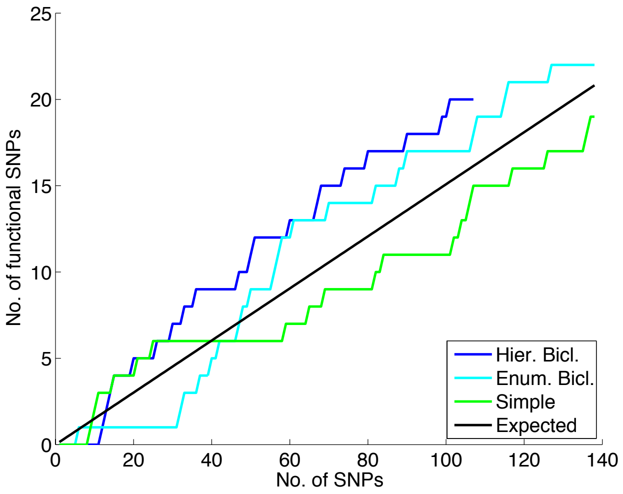


Figure 12.2: Comparative evaluation of different strategies to discover functional SNPs.

(Section 6.4.2); then, the SNPs were ranked in the order of their first appearance in the biclusters; SNPs of the same bicluster appeared in the order of descending association strength (summed across all genes of the bicluster). In the hierarchical approach (Hier. Bicl.), ranking was done analogously, except for the sorting of the different biclusters; as the p-values from Section 6.4.2 require enumeration of solutions, we used here a simple heuristic criterion inspired by the measure in [16]: the total weight of the bicluster divided by its cardinality (i.e., the number of its SNPs and its genes). For comparison, we applied a simple entry-wise analysis of the association matrix, ranking the SNPs according to the strongest association value they exhibit. As one can see from the figure, the bicluster approaches performed better than the simple approach, the hierarchical approach being on top. However, both bicluster curves are still quite close to the expected number of functional SNPs when randomly selecting a subset of SNPs, so inferring functional importance of SNPs from association data is a hard task; furthermore, the data selection and the limited amount of confirmed functional annotation might bias the evaluation.

In Table 12.1 and Table 12.2, we list the functional hits for the enumerative bicluster approach and the hierarchical bicluster approach, respectively. Five SNPs appear in both lists (marked in bold), indicating their significance. They are either annotated as exons or as totally intronic RNAs, which may act as regulators of cellular expression patterns [159]. Another interesting question is whether the SNPs in a bicluster show a specific kind of interaction, meaning that they have a much stronger effect on the expression if they occur together than can be explained by their individual contributions. This phenomenon is also called epistasis [43] and

SNP	Rank	Functional Annotation
rs6776501	6	indel conserved
rs434082	32	tin RNA
rs1570492	33	transp. free
rs3772069	37	transp. free
rs1047187	40	tin RNA
rs3796504	42	exon
rs2789417	47	transp. free, indel conserved, exon
rs4895642	48	indel conserved, exon
rs4841294	50	tin RNA
rs1051756	56	exon
rs164288	57	indel conserved, exon
rs2817178	58	transp. free
rs369487	61	tin RNA
rs3811888	70	exon
rs3740199	82	transp. free, exon
rs3088365	88	exon
rs3097830	90	tin RNA
rs1051219	107	indel conserved, exon
rs1042113	108	exon
rs739496	115	transp. free, exon
rs1919460	116	rnadb_mrna
rs3772479	127	tin RNA, exon

Table 12.1: Enumerative bicluster SNPs with functional annotation. Bold text indicates SNPs that also occur in hierarchical biclusters (Table 12.2).

could be an indicator for an underlying biological relationship between the genetic markers. To investigate that, we systematically computed interaction p-values of SNP marker combinations with respect to the genes of the same bicluster. However, after correction for multiple testing, the p-values were not significant enough to comply with standard thresholds. In fact, the bicluster criterion only requires that the SNPs are associated with the same set of genes, it does not especially favor interacting SNPs against independent regulatory factors.

Still, a more comprehensive bicluster analysis of association data might reveal interesting relationships between SNPs, which is a promising direction because direct approaches to detect SNP interactions are infeasible, in particular with regard to higher-order relationships. Due to its relatively low complexity, the hierarchical biclustering approach is suitable to be applied on large-scale datasets.

SNP	Rank	Functional Annotation
rs3796504	12	exon
rs164288	13	indel conserved, exon
rs1689512	14	transp. free, pin RNA
rs2298193	15	indel conserved
rs2228315	20	exon
rs10798	26	transp. free, exon
rs2789417	30	transp. free, indel conserved, exon
rs3097830	33	tin RNA
rs716615	36	transp. free
rs3735803	47	indel conserved, exon
rs1044729	50	exon
rs7128926	51	tin RNA
rs1124595	60	tin RNA
rs2230862	67	transp. free, indel conserved, exon
rs4841294	68	tin RNA
rs1891787	74	transp. free
rs2279587	80	exon
rs716417	90	indel conserved
rs12594	99	exon
rs6459166	101	transp. free, indel conserved, exon

Table 12.2: Hierarchical bicluster SNPs with functional annotation. Bold text indicates SNPs that also occur in enumerative biclusters (Table 12.1).

Part V

Conclusion

13 Summary

Structured data arise in many different application fields. One prominent example is computational systems biology, where networks represent multiple kinds of relationships involving genes or gene products. Our cluster detection methods provide novel approaches to systematically discover interesting patterns in such data.

The main focus of the thesis lies on an enumerative approach to extract dense clusters from structured data (Part II). In a nutshell, given a set of pairwise or higher-order relationships between objects as the input, the output consists of all subset patterns where the relative number or strength of relationships between objects exceeds a user-defined threshold. While classical set enumeration strategies from data mining turn out to be inappropriate for this task because they lack effective pruning rules, the reverse search paradigm provides an elegant way to define an anti-monotonic search procedure; with that, the time complexity for computing a single solution is in the order of the input size. In theory, the methodological framework can deal with a broad class of problems, including pattern mining in graphs or higher-order tensors with undirected or directed relationships, having weights or not. In practice, however, the number of solutions can grow prohibitively if there exist strongly overlapping patterns. Often, external data sources are available and can help to further restrict the search. We exemplarily discussed possibilities for a transparent integration of constraints from background information. Furthermore, alternative search criteria and practical speed-up techniques were presented.

We contrasted the enumerative cluster finding approach with a generalized variant of hierarchical clustering (Part III). While the enumerative method guarantees to yield all patterns that satisfy the prespecified interaction density criterion, the hierarchical method produces sets of disjoint clusters built in a hierarchical way. It also considers an interaction criterion, by successively merging the clusters with the largest strength of inter-cluster relationships. We extended this merging strategy to the bicluster scenario. Also, application to higher-order relations is conceivable; in this case, however, obtaining well-defined dendogram structures is challenging. As a biclustering procedure, the approach is very efficient because it constructs non-overlapping biclusters based on a local optimality criterion. In that sense, the enumerative and the hierarchical approach are opposite extremes in the spectrum of cluster detection methods. Both of them behave in a very transparent way, which facilitates the interpretation of their results. Most other methods produce results between those two extremes, making some (often implicit) trade-off between completeness and efficiency. Our complete enumeration method offers the opportunity to make this trade-off explicit by pushing some constraints into the search framework.

Both cluster detection methods were tested in real-world systems biology tasks (Part IV). The enumerative approach proved to be useful in discovering functional protein complexes based on experimental protein-protein interaction data. We applied it on data collections from two different organisms, yeast and human, and compared it with several competitive approaches. The potential of integrative data analysis was demonstrated by coanalyzing the interaction data of yeast with profiles of evolutionary conservation and phenotypic properties; as an interesting scenario for human data analysis, we showed predictions for tissue-specific variants of protein complexes. Furthermore, we considered the task of revealing functional modules that are shared across several coexpression networks of yeast genes. Practically, this was done by representing the input information as a three-way data cube from which dense patterns were extracted. By that, we successfully retrieved groups of functionally related genes; the results were competitive with those of alternative methods. Third, a large gene expression compendium of the plant *Arabidopsis thaliana* was analyzed with the proposed hierarchical biclustering approach. In spite of the greedy nature inherent to the agglomerative clustering paradigm, it detected meaningful biological structure. In another bicluster analysis problem, we applied both the hierarchical and the enumerative approach. The goal was to detect functional associations between single nucleotide polymorphisms in the genomic sequence and the expression of certain genes. The results differed between the methods, but the overlap consisted exclusively of polymorphisms in functionally confirmed regions.

Overall, the enumerative cluster mining method is valuable for a systematic analysis of structured data. The hierarchical biclustering, although simple in concept, can detect interesting patterns in large-scale data and thereby complements the enumerative method.

14 Discussion and Future Work

We conclude by discussing important aspects of this work and pointing out future directions. First of all, let us emphasize that the proposed approaches are generic and as such applicable in a large variety of usage scenarios beyond the ones described in the previous chapters. Staying in the field of biological data analysis, further use cases could be, for example, the discovery of sequence families based on sequence alignment scores [53] and the search for structural subunits of proteins [64]. Higher-order structure mining becomes necessary in gene expression experiments with additional dimensions like developmental stages, tissues, different populations etc. [243].

Often, predicted cluster patterns serve as input for further computational analyses. Gene or protein modules, for instance, are typically used for function prediction [193, 197]. Moreover, they can assist in elucidating transcriptional regulatory networks [154] or in comparing expression states of different cellular conditions [188]. Recently, module-based analysis of biological networks has been exploited to study the molecular basis of diseases as well as relationships between diseases [204]; features derived from network modules can also help to improve the classification accuracy in disease prediction [40]; see [96] for a review on using protein networks to understand disease. Furthermore, module prediction is an important step in network alignment [211]. In all these studies one should keep in mind that living cells cannot be described by one static set of modules; rather, there exist multiple levels of organization [166], and modules change dynamically [8].

Regarding the multitude of module discovery methods, what is the specific contribution of our approach in these applications? A clear advantage of our enumerative method is that it allows for a systematic discovery of dense substructures, providing the guarantee that no valid solution is missed. This is particularly promising in studies where multiple data sources need to be combined. The enumerative framework can integrate arbitrarily many auxiliary datasets, respecting individual constraints for each of them. From a biological perspective, it is crucial to use as many data sources as possible to analyze a certain phenomenon. By that, it is possible to draw more robust conclusions, detect context-specific peculiarities, or focus on new findings. In method development, however, some background knowledge (e.g., Gene Ontology classification) is intentionally left out to use it later for evaluation. An ideal workflow would start with that, and once a suitable method has been selected, one would integrate all available knowledge into the analysis and make new predictions, the most significant of which should be validated by biological experiments.

Technically, it is important that the predefined characteristics of patterns are exploited as early as possible during the search in order to avoid the generation of non-interesting patterns. For the auxiliary data, we introduced some classes of constraints where this can be easily done. And regarding our main criterion, the within-cluster interaction density, the method is designed such that pruning is effective; as we have shown, this framework even works for higher-order cluster analysis. However, while the density criterion is intuitive and widely used in cluster prediction (e.g., [16, 166]), it has some drawbacks. Namely, for many larger networks the task of clique finding is intractable because the number of solutions explodes. As our approach employs a more flexible pattern definition including cliques as a special case, the problem is exacerbated, in particular for low density thresholds or higher-order settings. The situation naturally improves if side constraints restrict the search space. Furthermore, one can introduce probabilistic or heuristic rules to explore only a subset of solutions; in particular, one important topic for future work is how to systematically exploit cluster overlap constraints. The hierarchical biclustering approach we presented yields dense clusters that are totally disjoint, but due to the agglomerative strategy, they might not be the most significant ones. Quasi-cliques [145, 172] constitute a compromise between the strict clique criterion and the flexible dense cluster criterion. It is an open question how to search most efficiently for quasi-cliques in higer-order or weighted data. Possibly, hybrid approaches of reverse search and neighborhood-based pruning can be developed. Moreover, one should further investigate the efficient integration of density and connectivity requirements.

Also from a biological point of view, the cluster criterion is open to debate. An alternative criterion that is often used is the homogeneity of the values within the subarray corresponding to a cluster [243]. This is a meaningful requirement in some applications, but it should be noted that in many systems biology scenarios clusters consisting of entries with low absolute values are not interesting, because low entries merely indicate the absence of an effect (i.e., such information is biologically rather unspecific); this introduces some asymmetry into the problem, which should be

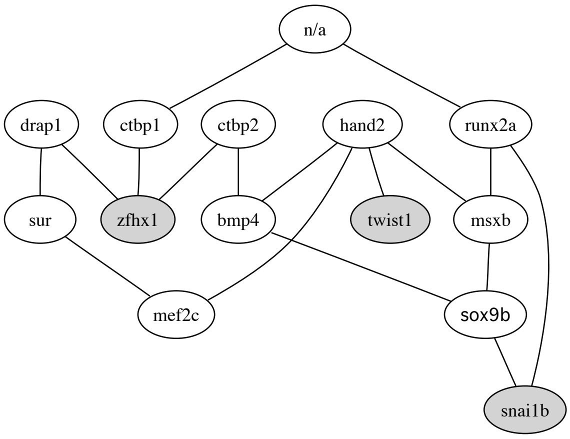


Figure 14.1: Connection subgraph discovered in a protein interaction network of zebrafish provided by the STRING database (version 7.1) [219], using the method by Koren *et al.* [126]. The query nodes are marked as shaded ellipses.

taken into account during the search. Other approaches look for patterns of coherent evolution [151], i.e., correlated up- and down-behavior in subarrays. In contrast, our criterion focuses either on strongly positive patterns, or on strongly negative patterns (in practice, the latter can be achieved by flipping the sign of all data values during the search). This makes sense in settings where the weight reflects the significance of an observation or the confidence of a prediction. While we used in our experiments a default value of zero for missing entries, our method is compatible to be combined with data modeling approaches that predict missing values in a preprocessing step. Finally, biologists are most frequently interested in deciphering a small subnetwork of gene relationships for a specific biological phenomenon, rather than in analyzing a whole genome network. The area of focus is typically defined by a set of genes that are known to be key players in a cellular process of interest. There already exist a few approaches to extract local subnetworks directed by query nodes [56, 86, 126]. An illustrative example from real-world data is given in Figure 14.1. One interesting extension would be to find the local module structure around such a connection subgraph, i.e., its embedding in the global network.

To conclude, the methods presented in this work provide novel ways of pattern discovery in structured data, which are potentially useful in various systems biology applications, complementing other approaches. However, systems biology is still in its infancy and needs further progress in both data acquisition and analysis methods. It remains an exciting and challenging task to investigate and infer relations between molecular components and understand how patterns of interaction translate into cellular functions.

Bibliography

[1] E. Acar, C. Aykut-Bingol, H. Bingol, R. Bro, and B. Yener. Multiway analysis of epilepsy tensors. *Bioinformatics*, 23(13):i10–i18, 2007.

[2] E. Acar, S. Çamtepe, and B. Yener. Collective sampling and analysis of high order tensors for chatroom communications. In *Intelligence and Security Informatics*, pages 213–224. Springer, 2006.

[3] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 94–105. ACM, 1998.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499. Morgan Kaufmann, 1994.

[5] E. A. Akkoyunlu. The enumeration of maximal cliques of large graphs. *SIAM J. Comput.*, 2(1):1–6, 1973.

[6] L. Alberghina and H. Westerhoff. *Systems Biology: Definitions and Perspectives.* Topics in Current Genetics 13. Springer, 2005.

[7] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the Cell.* Garland Science, fifth edition, 2007.

[8] R. P. Alexander, P. M. Kim, T. Emonet, and M. B. Gerstein. Understanding modularity in molecular networks requires dynamics. *Sci. Signal.*, 2(81):pe44, 2009.

[9] H. Arimura and T. Uno. An efficient polynomial space and polynomial delay algorithm for enumeration of maximal motifs in a sequence. *Journal of Combinatorial Optimization*, 13(3):243–262, 2007.

[10] H. Arimura and T. Uno. Mining maximal flexible patterns in a sequence. In *New Frontiers in Artificial Intelligence*, volume 4914 of *Lecture Notes in Computer Science*, pages 307–317. Springer, 2008.

[11] Y. Asahiro, R. Hassin, and K. Iwama. Complexity of finding dense subgraphs. *Discrete Appl. Math.*, 121(1-3):15–26, 2002.

[12] Y. Asahiro, K. Iwama, H. Tamaki, and T. Tokuyama. Greedily finding a dense subgraph. *J. Algorithms*, 34(2):203–221, 2000.

[13] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene Ontology: tool for the unification of biology. the Gene Ontology consortium. *Nat. Genet.*, 25(1):25–29, 2000.

[14] D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Appl. Math.*, 65:21–46, 1996.

[15] G. D. Bader, D. Betel, and C. W. V. Hogue. BIND: the Biomolecular Interaction Network Database. *Nucl. Acids Res.*, 31(1):248–250, 2003.

[16] G. D. Bader and C. W. Hogue. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, 4:2, 2003.

[17] A. Banerjee, S. Basu, and S. Merugu. Multi-way clustering on relation graphs. In *SDM '07: Proceedings of the 7th SIAM International Conference on Data Mining*, 2007.

[18] S. E. Baranzini, P. Mousavi, J. Rio, S. J. Caillier, A. Stillman, P. Villoslada, M. M. Wyatt, M. Comabella, L. D. Greller, R. Somogyi, X. Montalban, and J. R. Oksenberg. Transcription-based prediction of response to IFNβ using supervised computational methods. *PLoS Biol.*, 3(1):e2, 2004.

[19] R. J. Bayardo, Jr. Efficiently mining long patterns from databases. In *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 85–93. ACM, 1998.

[20] C. F. Beckmann and S. M. Smith. Tensorial extensions of independent component analysis for multisubject FMRI analysis. *Neuroimage*, 25(1):294–311, 2005.

[21] G. Bejerano, N. Friedman, and N. Tishby. Efficient exact p-value computation for small sample, sparse, and surprising categorical data. *J. Comput. Biol.*, 11(5):867–886, 2004.

[22] J. Besson, C. Robardet, L. De Raedt, and J.-F. Boulicaut. Mining bi-sets in numerical data. In *KDID '06: Knowledge Discovery in Inductive Databases, 5th International Workshop*, volume 4747 of *Lecture Notes in Computer Science*, pages 11–23. Springer, 2006.

[23] R. Bisiani. Beam search. In *Encyclopedia of Articial Intelligence*, pages 56–58. Wiley, 1987.

[24] K. M. Borgwardt, H.-P. Kriegel, and P. Wackersreuther. Pattern mining in frequent dynamic subgraphs. In *ICDM '06: Proceedings of the 6th International Conference on Data Mining*, pages 818–822. IEEE Computer Society, 2006.

[25] A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph Classes: A Survey.* SIAM Monographs on Discrete Mathematics and Applications. SIAM, 1987.

[26] B.-J. Breitkreutz, C. Stark, T. Reguly, L. Boucher, A. Breitkreutz, M. Livstone, R. Oughtred, D. H. Lackner, J. Bahler, V. Wood, K. Dolinski, and M. Tyers. The BioGRID interaction database: 2008 update. *Nucl. Acids Res.*, 36(suppl_1):D637–D640, 2008.

[27] B.-J. Breitkreutz, C. Stark, and M. Tyers. Osprey: a network visualization system. *Genome Biology*, 4(3):R22, 2003.

[28] K. Bryan and P. Cunningham. Bottom-up biclustering of expression data. In *Proceedings of the 2006 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 232–239. IEEE Computer Society, 2006.

[29] W. Castillo and J. Trejos. Recurrence properties in two-mode hierarchical clustering. In *Classification and information processing at the turn of the millennium*, pages 68–73, 2000.

[30] C. Cenciarelli, D. Chiaur, D. Guardavaccaro, W. Parks, M. Vidal, and M. Pagano. Identification of a family of human F-box proteins. *Curr. Biol.*, 9:1177–1179, 1999.

[31] L. Cerf, J. Besson, C. Robardet, and J.-F. Boulicaut. Data Peeler: Contraint-based closed pattern mining in n-ary relations. In *SDM '08: Proceedings of the 8th SIAM International Conference on Data Mining*, pages 37–48, 2008.

[32] L. Cerf, J. Besson, C. Robardet, and J.-F. Boulicaut. Closed patterns meet n-ary relations. *ACM Trans. Knowl. Discov. Data*, 3(1):1–36, 2009.

[33] G. Cesareni, A. Ceol, C. Gavrila, L. M. Palazzi, M. Persico, and M. V. Schneider. Comparative interactomics. *FEBS Lett.*, 579(8):1828–1833, 2005.

[34] M. Charikar. Greedy approximation algorithms for finding dense components in a graph. In *APPROX '00: Proceedings of the 3rd International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 84–95. Springer, 2000.

[35] A. Chatr-aryamontri, A. Ceol, L. M. Palazzi, G. Nardelli, M. V. Schneider, L. Castagnoli, and G. Cesareni. MINT: the Molecular INTeraction database. *Nucl. Acids Res.*, 35(suppl_1):D572–D574, 2007.

[36] S. Chavez, T. Beilharz, A. G. Rondon, H. Erdjument-Bromage, P. Tempst, J. Q. Svejstrup, T. Lithgow, and A. Aguilera. A protein complex containing Tho2, Hpr1, Mft1 and a novel protein, Thp2, connects transcription elongation with mitotic recombination in Saccharomyces cerevisiae. *EMBO J.*, 19(21):5824–5834, 2000.

[37] J. Chen and B. Yuan. Detecting functional modules in the yeast protein-protein interaction network. *Bioinformatics*, 22(18):2283–2290, 2006.

[38] Y. Cheng and G. M. Church. Biclustering of expression data. *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology*, 8:93–103, 2000.

[39] W. Chu, Z. Ghahramani, R. Krause, and D. L. Wild. Identifying protein complexes in high-throughput protein interaction screens using an infinite latent feature model. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 231–242, 2006.

[40] H. Y. Chuang, E. Lee, Y. T. Liu, D. Lee, and T. Ideker. Network-based classification of breast cancer metastasis. *Mol. Syst. Biol.*, 3:140, 2007.

[41] A. Clauset, M. E. J. Newman, , and C. Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70(6):066111, 2004.

[42] D. N. Cooper, E. V. Ball, and M. Krawczak. The human gene mutation database. *Nucleic Acids Res.*, 26(1):285–287, 1998.

[43] H. J. Cordell. Epistasis: what it means, what it doesn't mean, and statistical methods to detect it in humans. *Hum. Mol. Genet.*, 11(20):2463–2468, 2002.

[44] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* The MIT Press, third edition, 2009.

[45] C. Creighton and S. Hanash. Mining gene expression databases for association rules. *Bioinformatics*, 19(1):79–86, 2003.

[46] A. C. Culhane, T. Schwarzl, R. Sultana, K. C. Picard, S. C. Picard, T. H. Lu, K. R. Franklin, S. J. French, G. Papenhausen, M. Correll, and J. Quackenbush. GeneSigDB – a curated database of gene expression signatures. *Nucl. Acids Res.*, 38(suppl_1):D716–D725, 2010.

[47] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD '01: Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 269–274. ACM, 2001.

[48] S. Dietmann, E. Georgii, A. Antonov, K. Tsuda, and H.-W. Mewes. The DICS repository: module-assisted analysis of disease-related gene lists. *Bioinformatics*, 25(6):830–831, 2009.

[49] M. T. Dittrich, G. W. Klau, A. Rosenwald, T. Dandekar, and T. Müller. Identifying functional modules in protein-protein interaction networks: an integrated exact approach. *Bioinformatics*, 24(13):i223–i231, 2008.

[50] A. M. Dudley, D. M. Janse, A. Tanay, R. Shamir, and G. M. Church. A global view of pleiotropy and phenotypically derived gene function in yeast. *Mol. Syst. Biol.*, 1:2005.0001, 2005.

[51] S. Džeroski. Multi-relational data mining: an introduction. *SIGKDD Explor. Newsl.*, 5(1):1–16, 2003.

[52] K. Elbing, R. R. McCartney, and M. C. Schmidt. Purification and characterization of the three Snf1-activating kinases of Saccharomyces cerevisiae. *Biochem. J.*, 393(3):797–805, 2006.

[53] A. J. Enright, S. van Dongen, and C. A. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Res.*, 30(7):1575–1584, 2002.

[54] M. Ester and J. Sander. *Knowledge discovery in databases*. Springer, 2000.

[55] L. Everett, L. S. Wang, and S. Hannenhalli. Dense subgraph computation via stochastic search: application to detect transcriptional modules. *Bioinformatics*, 22(14):e117–e123, 2006.

[56] C. Faloutsos, K. S. McCurley, and A. Tomkins. Fast discovery of connection subgraphs. In *KDD '04: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 118–127. ACM, 2004.

[57] I. J. Farkas, D. Abel, G. Palla, and T. Vicsek. Weighted network modules. *New J. Phys.*, 9(180), 2007.

[58] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. Knowledge discovery and data mining: Towards a unifying framework. In *KDD '96: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 82–88. AAAI Press, 1996.

[59] U. Feige, G. Kortsarz, and D. Peleg. The dense k-subgraph problem. *Algorithmica*, 29(3):59–78, 2001.

[60] B. Gao, T.-Y. Liu, and W.-Y. Ma. Star-structured high-order heterogeneous data co-clustering based on consistent information theory. In *ICDM '06: Proceedings of the Sixth International Conference on Data Mining*, pages 880–884. IEEE Computer Society, 2006.

[61] A. P. Gasch, P. T. Spellman, C. M. Kao, O. Carmel-Harel, M. B. Eisen, G. Storz, D. Botstein, and P. O. Brown. Genomic expression programs in the response of yeast cells to environmental changes. *Mol. Biol. Cell*, 11(12):4241–4257, 2000.

[62] A. C. Gavin, P. Aloy, P. Grandi, R. Krause, M. Boesche, M. Marzioch, C. Rau, L. J. Jensen, S. Bastuck, B. Dumpelfeld, A. Edelmann, M. A. Heurtier, V. Hoffman, C. Hoefert, K. Klein, M. Hudak, A. M. Michon, M. Schelder, M. Schirle, M. Remor, T. Rudi, S. Hooper, A. Bauer, T. Bouwmeester,

G. Casari, G. Drewes, G. Neubauer, J. M. Rick, B. Kuster, P. Bork, R. B. Russell, and G. Superti-Furga. Proteome survey reveals modularity of the yeast cell machinery. *Nature*, 440(7084):631–636, 2006.

[63] A. C. Gavin, M. Bosche, R. Krause, P. Grandi, M. Marzioch, A. Bauer, J. Schultz, J. M. Rick, A. M. Michon, C. M. Cruciat, M. Remor, C. Hofert, M. Schelder, M. Brajenovic, H. Ruffner, A. Merino, K. Klein, M. Hudak, D. Dickson, T. Rudi, V. Gnau, A. Bauch, S. Bastuck, B. Huhse, C. Leutwein, M. A. Heurtier, R. R. Copley, A. Edelmann, E. Querfurth, V. Rybin, G. Drewes, M. Raida, T. Bouwmeester, P. Bork, B. Seraphin, B. Kuster, G. Neubauer, and G. Superti-Furga. Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature*, 415(6868):141–147, 2002.

[64] J.-C. Gelly, A. G. De Brevern, and S. Hazout. 'Protein Peeling': an approach for splitting a 3D protein structure into compact fragments. *Bioinformatics*, 22(2):129–133, 2006.

[65] G. F. Georgakopoulos and K. Politopoulos. MAX-DENSITY revisited: a generalization and a more efficient algorithm. *The Computer Journal*, 50(3):348–356, 2007.

[66] E. Georgii, S. Dietmann, T. Uno, P. Pagel, and K. Tsuda. Mining expression-dependent modules in the human interaction network. *BMC Bioinformatics*, 8(Suppl. 8):S4, 2007.

[67] E. Georgii, S. Dietmann, T. Uno, P. Pagel, and K. Tsuda. Enumeration of condition-dependent dense modules in protein interaction networks. *Bioinformatics*, 25(7):933–940, 2009.

[68] E. Georgii, K. Tsuda, and B. Schölkopf. Multi-way set enumeration in weight tensors. *Mach. Learn.*, Special issue on Mining and Learning with Graphs and Relations. To appear.

[69] E. Georgii, K. Tsuda, and B. Schölkopf. Multi-way set enumeration in real-valued tensors. In *DMMT '09: Proceedings of the 2nd Workshop on Data Mining using Matrices and Tensors*, pages 32–41 (Article No. 4). ACM, 2009.

[70] G. Gibson. Microarray analysis. *PLoS Biol.*, 1(1):e15, 10 2003.

[71] M. Girvan and M. E. Newman. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA*, 99(12):7821–7826, 2002.

[72] L. A. Goldberg. Efficient algorithms for listing unlabeled graphs. *J. Algorithms*, 13(1):128–143, 1992.

[73] L. A. Goldberg. Polynomial space polynomial delay algorithms for listing families of graphs. In *STOC '93: Proceedings of the 25th annual ACM Symposium on Theory of Computing*, pages 218–225. ACM, 1993.

[74] K. Gouda and M. J. Zaki. Efficiently mining maximal frequent itemsets. In *ICDM '01: Proceedings of the IEEE International Conference on Data Mining*, pages 163–170. IEEE Computer Society, 2001.

[75] U. Güldener, M. Münsterkötter, G. Kastenmüller, N. Strack, J. van Helden, C. Lemer, J. Richelles, S. J. Wodak, J. Garcia-Martinez, J. E. Perez-Ortin, H. Michael, A. Kaps, E. Talla, B. Dujon, B. Andre, J. L. Souciet, J. De Montigny, E. Bon, C. Gaillardin, and H. W. Mewes. CYGD: the Comprehensive Yeast Genome Database. *Nucl. Acids Res.*, 33(suppl_1):D364–D368, 2005.

[76] U. Güldener, M. Münsterkötter, M. Oesterheld, P. Pagel, A. Ruepp, H. W. Mewes, and V. Stümpflen. MPact: the MIPS protein interaction resource on yeast. *Nucleic Acids Res.*, 34(Database issue):D436–D441, 2006.

[77] D. Gunopulos, H. Mannila, and S. Saluja. Discovering all most specific sentences by randomized algorithms. In *ICDT '97: Proceedings of the 6th International Conference on Database Theory*, pages 215–229. Springer, 1997.

[78] J. Han and M. Kamber. *Data Mining: Concepts and Techniques.* The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, second edition, 2006.

[79] D. J. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining.* The MIT Press, 2001.

[80] D. Hanisch, A. Zien, R. Zimmer, and T. Lengauer. Co-clustering of biological networks and gene expression data. *Bioinformatics*, 18(suppl_1):S145–S154, 2002.

[81] M. Haraguchi and Y. Okubo. A method for pinpoint clustering of web pages with pseudo-clique search. In *Federation over the Web*, volume 3847 of *Lecture Notes in Computer Science*, pages 59–78. Springer, 2006.

[82] E. Hartuv and R. Shamir. A clustering algorithm based on graph connectivity. *Information Processing Letters*, 76:175–181, 1999.

[83] L. H. Hartwell, J. J. Hopfield, S. Leibler, and A. W. Murray. From molecular to modular cell biology. *Nature*, 402(6761 Suppl):C47–C52, 1999.

[84] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* Springer, second edition, 2009.

[85] H. Hermjakob, L. Montecchi-Palazzi, C. Lewington, S. Mudali, S. Kerrien, S. Orchard, M. Vingron, B. Roechert, P. Roepstorff, A. Valencia, H. Margalit, J. Armstrong, A. Bairoch, G. Cesareni, D. Sherman, and R. Apweiler. IntAct: an open source molecular interaction database. *Nucl. Acids Res.*, 32(suppl_1):D452–D455, 2004.

[86] P. Hintsanen and H. Toivonen. Finding reliable subgraphs from large probabilistic graphs. *Data Min. Knowl. Discov.*, 17(1):3–23, 2008.

[87] H. Hirsh. Data mining research: Current status and future opportunities. *Stat. Anal. Data Min.*, 1(2):104–107, 2008.

[88] D. Hochbaum. Approximating clique and biclique problems. *Journal of Algorithms*, 29:174–200, 1998.

[89] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *IJCAI '99: Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 668–693, 1999.

[90] J. Hopcroft, O. Khan, B. Kulis, and B. Selman. Natural communities in large linked networks. In *KDD '03: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 541–546. ACM, 2003.

[91] H. Hu, X. Yan, Y. Huang, J. Han, and X. J. Zhou. Mining coherent dense subgraphs across massive biological networks for functional discovery. *Bioinformatics*, 21(suppl_1):i213–i221, 2005.

[92] Y. Huang, H. Li, H. Hu, X. Yan, M. S. Waterman, H. Huang, and X. J. Zhou. Systematic discovery of functional modules and context-specific functional annotation of human genome. *Bioinformatics*, 23(13):i222–i229, 2007.

[93] W. Huber, A. von Heydebreck, H. Sultmann, A. Poustka, and M. Vingron. Variance stabilization applied to microarray data calibration and to the quantification of differential expression. *Bioinformatics*, 18(suppl_1):S96–S104, 2002.

[94] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

[95] T. Ideker, O. Ozier, B. Schwikowski, and A. F. Siegel. Discovering regulatory and signalling circuits in molecular interaction networks. *Bioinformatics*, 18(suppl_1):S233–S240, 2002.

[96] T. Ideker and R. Sharan. Protein networks in disease. *Genome Res.*, 18(4):644–652, 2008.

[97] A. K. Jain and R. C. Dubes. *Algorithms for clustering data.* Prentice-Hall, 1988.

[98] C. Janeway, P. Travers, M. Walport, and M. Shlomchik. *Immunobiology: Immune System in Health and Disease.* Garland Science, sixth edition, 2005.

[99] R. Jansen, H. Yu, D. Greenbaum, Y. Kluger, N. J. Krogan, S. Chung, A. Emili, M. Snyder, J. F. Greenblatt, and M. Gerstein. A Bayesian networks approach for predicting protein-protein interactions from genomic data. *Science*, 302(5644):449–453, 2003.

[100] R. Jäschke, A. Hotho, C. Schmitz, B. Ganter, and G. Stumme. TRIAS – an algorithm for mining iceberg tri-lattices. In *ICDM '06: Proceedings of the 6th International Conference on Data Mining*, pages 907–911. IEEE Computer Society, 2006.

[101] S. Jegelka, S. Sra, and A. Banerjee. Approximation algorithms for tensor clustering. In *Algorithmic Learning Theory*, pages 368–383, 2009.

[102] L. J. Jensen, M. Kuhn, M. Stark, S. Chaffron, C. Creevey, J. Müller, T. Doerks, P. Julien, A. Roth, M. Simonovic, P. Bork, and C. von Mering. STRING 8 – a global view on proteins and their functional interactions in 630 organisms. *Nucl. Acids Res.*, 37(suppl_1):D412–D416, 2009.

[103] L. Ji, K.-L. Tan, and A. K. H. Tung. Mining frequent closed cubes in 3D datasets. In *VLDB '06: Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 811–822. VLDB Endowment, 2006.

[104] D. Jiang and J. Pei. Mining frequent cross-graph quasi-cliques. *ACM Trans. Knowl. Discov. Data*, 2(4):1–42, 2009.

[105] D. Jiang, C. Tang, and A. Zhang. Cluster analysis for gene expression data: A survey. *IEEE Trans. on Knowl. and Data Eng.*, 16(11):1370–1386, 2004.

[106] X. Jiang, H. Xiong, C. Wang, and A.-H. Tan. Mining globally distributed frequent subgraphs in a single labeled graph. *Data Knowl. Eng.*, 68(10):1034–1058, 2009.

[107] G. Joshi-Tope, M. Gillespie, I. Vastrik, P. D'Eustachio, E. Schmidt, B. de Bono, B. Jassal, G. R. Gopinath, G. R. Wu, L. Matthews, S. Lewis, E. Birney, and L. Stein. Reactome: a knowledgebase of biological pathways. *Nucleic Acids Res.*, 33(Database issue):D428–D432, 2005.

[108] M. Kalaev, V. Bafna, and R. Sharan. Fast and accurate alignment of multiple protein networks. In *RECOMB '08: Proceedings of the 12th Annual International Conference on Research in Computational Molecular Biology*, pages 246–256, 2008.

[109] D. Karolchik, R. M. Kuhn, R. Baertsch, G. P. Barber, H. Clawson, M. Diekhans, B. Giardine, R. A. Harte, A. S. Hinrichs, F. Hsu, K. M. Kober, W. Miller, J. S. Pedersen, A. Pohl, B. J. Raney, B. Rhead, K. R. Rosenbloom, K. E. Smith, M. Stanke, A. Thakkapallayil, H. Trumbower, T. Wang, A. S. Zweig, D. Haussler, and W. J. Kent. The UCSC Genome Browser Database: 2008 update. *Nucleic Acids Res.*, 36(Database issue):D773–D779, 2008.

[110] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[111] G. Karypis, V. Kumar, and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48:96–129, 1998.

[112] C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda. Learning systems of concepts with an infinite relational model. In *AAAI '06: Proceedings of the 21st National Conference on Artificial Intelligence*, pages 381–388. AAAI Press, 2006.

[113] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(1):291–307, 1970.

[114] S. Khuller and B. Saha. On finding dense subgraphs. In *ICALP '09: Proceedings of the 36th International Colloquium on Automata, Languages and Programming*, pages 597–608. Springer, 2009.

[115] L. Kiemer and G. Cesareni. Comparative interactomics: comparing apples and pears? *Trends Biotechnol.*, 25(10):448–454, 2007.

[116] T. Kin, K. Yamada, G. Terai, H. Okida, Y. Yoshinari, Y. Ono, A. Kojima, Y. Kimura, T. Komori, and K. Asai. fRNAdb: a platform for mining/annotating functional RNA candidates from non-coding RNA sequences. *Nucleic Acids Res.*, 35(Database issue):D145–D148, 2007.

[117] E. Kipreos and M. Pagano. The F-box protein family. *Genome Biology*, 1(5):reviews3002.1–reviews3002.7, 2000.

[118] H. Kitano. Systems Biology: A Brief Overview. *Science*, 295(5560):1662–1664, 2002.

[119] B. Klimt and Y. Yang. The Enron Corpus: A new dataset for email classification research. In *ECML '04: Proceedings of the 15th European Conference on Machine Learning*, pages 217–226. Springer, 2004.

[120] Y. Kluger, R. Basri, J. T. Chang, and M. Gerstein. Spectral biclustering of microarray data: Coclustering genes and conditions. *Genome Res.*, 13(4):703–716, 2003.

[121] D. M. Koepp, L. K. Schaefer, X. Ye, K. Keyomarsi, C. Chu, J. W. Harper, and S. J. Elledge. Phosphorylation-dependent ubiquitination of cyclin E by the SCFFbw7 ubiquitin ligase. *Science*, 294(5540):173–177, 2001.

[122] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. Technical Report SAND2007-6702, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, November 2007.

[123] T. G. Kolda, B. W. Bader, and J. P. Kenny. Higher-order web link analysis using multilinear algebra. In *ICDM '05: Proceedings of the 5th IEEE International Conference on Data Mining*, pages 242–249. IEEE Computer Society, 2005.

[124] T. G. Kolda and J. Sun. Scalable tensor decompositions for multi-aspect data mining. In *ICDM*, 2008.

[125] R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *ICML 02: Proceedings of the 19th International Conference on Machine Learning*, pages 315–322. Morgan Kaufmann, 2002.

[126] Y. Koren, S. C. North, and C. Volinsky. Measuring and extracting proximity in networks. In *KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 245–255. ACM, 2006.

[127] M. Koyutürk, W. Szpankowski, and A. Grama. Biclustering gene-feature matrices for statistically significant patterns. In *CSB '04: Proceedings of the 2004 IEEE Computational Systems Bioinformatics Conference*, pages 480–483. IEEE Computer Society, 2004.

[128] M. Koyutürk, W. Szpankowski, and A. Grama. Assessing significance of connectivity and conservation in protein interaction networks. *J. Comput. Biol.*, 14(6):747–764, 2007.

[129] S. Kramer, L. De Raedt, and C. Helma. Molecular feature mining in HIV data. In *KDD '01: Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 136–143. ACM, 2001.

[130] H.-P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Trans. Knowl. Discov. Data*, 3(1):1–58, 2009.

[131] V. N. Kristensen, H. Edvardsen, A. Tsalenko, S. H. Nordgard, T. Sørlie, R. Sharan, A. Vailaya, A. Ben-Dor, P. E. Lønning, S. Lien, S. Omholt, A.-C. Syvänen, Z. Yakhini, and A.-L. Børresen-Dale. Genetic variation in putative regulatory loci controlling gene expression in breast cancer. *Proc. Natl. Acad. Sci. USA*, 103(20):7735–7740, 2006.

[132] N. J. Krogan, G. Cagney, H. Yu, G. Zhong, X. Guo, A. Ignatchenko, J. Li, S. Pu, N. Datta, A. P. Tikuisis, T. Punna, J. M. Peregrin-Alvarez, M. Shales, X. Zhang, M. Davey, M. D. Robinson, A. Paccanaro, J. E. Bray, A. Sheung, B. Beattie, D. P. Richards, V. Canadien, A. Lalev, F. Mena, P. Wong, A. Starostine, M. M. Canete, J. Vlasblom, S. Wu, C. Orsi, S. R. Collins, S. Chandran, R. Haw, J. J. Rilstone, K. Gandi, N. J. Thompson, G. Musso, P. St Onge, S. Ghanny, M. H. Lam, G. Butland, A. M. Altaf-Ul, S. Kanaya, A. Shilatifard, E. O'Shea, J. S. Weissman, C. J. Ingles, T. R. Hughes, J. Parkinson, M. Gerstein, S. J. Wodak, A. Emili, and J. F. Greenblatt. Global landscape of protein complexes in the yeast Saccharomyces cerevisiae. *Nature*, 440(7084):637–643, 2006.

[133] M. Kuramochi and G. Karypis. An efficient algorithm for discovering frequent subgraphs. *IEEE Trans. on Knowl. and Data Eng.*, 16(9):1038–1051, 2004.

[134] M. Kuramochi and G. Karypis. Finding frequent patterns in a large sparse graph. *Data Min. Knowl. Discov.*, 11(3):243–271, 2005.

[135] L. Lazzeroni and A. Owen. Plaid models for gene expression data. *Statistica Sinica*, 12(1):61–86, 2002.

[136] M. Lei and B. K. Tye. Initiating DNA synthesis: from recruiting to activating the MCM complex. *J. Cell. Sci.*, 114(8):1447–1454, 2001.

[137] H. C. M. Leung, Q. Xiang, S. M. Yiu, and F. Y. L. Chin. Predicting protein complexes from PPI data: a core-attachment approach. *J Comput. Biol.*, 16(2):133–144, 2009.

[138] B. Lewin. *Genes IX*. Jones and Bartlett, ninth edition, 2007.

[139] G. Li, Q. Ma, H. Tang, A. H. Paterson, and Y. Xu. QUBIC: a qualitative biclustering algorithm for analyses of gene expression data. *Nucl. Acids Res.*, 37(15):e101, 2009.

[140] T. Li. A general model for clustering binary data. In *KDD '05: Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 188–197. ACM, 2005.

[141] D.-I. Lin and Z. M. Kedem. Pincer search: a new algorithm for discovering the maximum frequent sets. In *EDBT '98: Proceedings of the 6th International Conference on Extending Database Technology*, pages 105–119. Springer, 1998.

[142] J. Lin and J. Qian. Systems biology approach to integrative comparative genomics. *Expert Rev. Proteomics*, 4(1):107–119, 2007.

[143] Y.-R. Lin, J. Sun, P. Castro, R. Konuru, H. Sundaram, and A. Kelliher. MetaFac: community discovery via relational hypergraph factorization. In *KDD '09: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 527–536. ACM, 2009.

[144] F. Ling, H. Morioka, E. Ohtsuka, and T. Shibata. A role for MHR1, a gene required for mitochondrial genetic recombination, in the repair of damage spontaneously introduced in yeast mtDNA. *Nucl. Acids Res.*, 28(24):4956–4963, 2000.

[145] G. Liu and L. Wong. Effective pruning techniques for mining quasi-cliques. In *ECML PKDD '08: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases – Part II*, pages 33–49. Springer, 2008.

[146] H. M. Lodhi and S. H. Muggleton. *Elements of Computational Systems Biology.* Wiley, 2010.

[147] H. Lodish, A. Berk, C. A. Kaiser, M. Krieger, M. P. Scott, A. Bretscher, H. Ploegh, and P. Matsudaira. *Molecular Cell Biology.* Freeman, sixth edition, 2007.

[148] B. Long, Z. M. Zhang, X. Wú, and P. S. Yu. Spectral clustering for multi-type relational data. In *ICML '06: Proceedings of the 23rd International Conference on Machine Learning*, pages 585–592. ACM, 2006.

[149] F. Luo, Y. Yang, C.-F. Chen, R. Chang, J. Zhou, and R. H. Scheuermann. Modular organization of protein interaction networks. *Bioinformatics*, 23(2):207–214, 2007.

[150] U. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[151] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 1(1):24–45, 2004.

[152] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval.* Cambridge University Press, 2008.

[153] D. Medini, A. Covacci, and C. Donati. Protein homology network families reveal step-wise diversification of type III and type IV secretion systems. *PLoS Comput. Biol.*, 2(12), 2006.

[154] T. Michoel, R. De Smet, A. Joshi, Y. Van de Peer, and K. Marchal. Comparative analysis of module-based versus direct methods for reverse-engineering transcriptional regulatory networks. *BMC Systems Biology*, 3(1):49, 2009.

[155] G. W. Milligan. Ultrametric hierarchical clustering algorithms. *Psychometrika*, 44(3):343–346, 1979.

[156] N. Mishra, D. Ron, and R. Swaminathan. A new conceptual clustering framework. *Mach. Learn.*, 56(1-3):115–151, 2004.

[157] A. J. Myers, J. R. Gibbs, J. A. Webster, K. Rohrer, A. Zhao, L. Marlowe, M. Kaleem, D. Leung, L. Bryden, P. Nath, V. L. Zismann, K. Joshipura, M. J. Huentelman, D. Hu-Lince, K. D. Coon, D. W. Craig, J. V. Pearson, P. Holmans, C. B. Heward, E. M. Reiman, D. Stephan, and J. Hardy. A survey of genetic human cortical gene expression. *Nat. Genet.*, 39(12):1494–1499, 2007.

[158] S.-I. Nakano and T. Uno. Constant time generation of trees with specified diameter. In *Graph-Theoretic Concepts in Computer Science, 30th International Workshop*, volume 3353 of *Lecture Notes in Computer Science*, pages 33–45. Springer, 2004.

[159] H. I. Nakaya, P. P. Amaral, R. Louro, A. Lopes, A. A. Fachel, Y. B. Moreira, T. A. El-Jundi, A. M. da Silva, E. M. Reis, and S. Verjovski-Almeida. Genome mapping and expression analyses of human intronic noncoding RNAs reveal tissue-specific patterns and enrichment in genes related to regulation of transcription. *Genome Biol.*, 8(3):R43, 2007.

[160] M. E. Newman. Modularity and community structure in networks. *Proc. Natl. Acad. Sci. USA*, 103(23):8577–8582, 2006.

[161] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

[162] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69(2):026113, 2004.

[163] K. P. O'Brien, M. Remm, and E. L. L. Sonnhammer. Inparanoid: a comprehensive database of eukaryotic orthologs. *Nucl. Acids Res.*, 33(suppl_1):D476–D480, 2005.

[164] H. Ogata, S. Goto, K. Sato, W. Fujibuchi, H. Bono, and M. Kanehisa. KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.*, 27(1):29–34, 1999.

[165] G. Orphanides, W.-H. Wu, W. S. Lane, M. Hampsey, and D. Reinberg. The chromatin-specific transcription elongation factor FACT comprises human SPT16 and SSRP1 proteins. *Nature*, 400(6741):284–288, 1999.

[166] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[167] G. Pandey, G. Atluri, M. Steinbach, C. L. Myers, and V. Kumar. An association analysis approach to biclustering. In *KDD '09: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 677–686. ACM, 2009.

[168] P. M. Pardalos and J. Xue. The maximum clique problem. *Journal of Global Optimization*, 4(3):301–328, 1994.

[169] D. Park, S. Lee, D. Bolser, M. Schroeder, M. Lappe, D. Oh, and J. Bhak. Comparative interactomics analysis of protein family interaction networks using PSIMAP (protein structural interactome map). *Bioinformatics*, 21(15):3234–3240, 2005.

[170] J. A. Parkkinen and S. Kaski. Searching for functional gene modules with interaction component models. *BMC Syst. Biol.*, 4:4, 2010.

[171] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *ICDT '99: Proceedings of the 7th International Conference on Database Theory*, pages 398–416. Springer, 1999.

[172] J. Pei, D. Jiang, and A. Zhang. Mining cross-graph quasi-cliques in gene expression and protein interaction data. In *ICDE '05: Proceedings of the 21st International Conference on Data Engineering*, pages 353–354. IEEE Computer Society, 2005.

[173] J. B. Pereira-Leal, A. J. Enright, and C. A. Ouzounis. Detection of functional modules from protein interaction networks. *Proteins*, 54(1):49–57, 2004.

[174] S. Peri, J. D. Navarro, T. Z. Kristiansen, R. Amanchy, V. Surendranath, B. Muthusamy, T. K. B. Gandhi, K. N. Chandrika, N. Deshpande, S. Suresh, B. P. Rashmi, K. Shanker, N. Padma, V. Niranjan, H. C. Harsha, N. Talreja, B. M. Vrushabendra, M. A. Ramya, A. J. Yatish, M. Joy, H. N. Shivashankar, M. P. Kavitha, M. Menezes, D. R. Choudhury, N. Ghosh, R. Saravana, S. Chandran, S. Mohan, C. K. Jonnalagadda, C. K. Prasad, C. Kumar-Sinha, K. S. Deshpande, and A. Pandey. Human protein reference database as a discovery resource for proteomics. *Nucl. Acids Res.*, 32(suppl_1):D497–D501, 2004.

[175] J. Pinney and D. Westhead. Betweenness-based decomposition methods for social and biological networks. In *Interdisciplinary Statistics and Bioinformatics*, pages 87–90, Leeds, 2006. Leeds University Press.

[176] A. Prelić, S. Bleuler, P. Zimmermann, A. Wille, P. Buhlmann, W. Gruissem, L. Hennig, L. Thiele, and E. Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, 2006.

[177] C. Quero, N. Colomé, M. R. Prieto, M. Carrascal, M. Posada, E. Gelpí, and J. Abian. Determination of protein markers in human serum: Analysis of protein expression in toxic oil syndrome studies. *Proteomics*, 4(2):303–315, 2004.

[178] J. Ramon and S. Nijssen. Polynomial-delay enumeration of monotonic graph classes. *J. Mach. Learn. Res.*, 10:907–929, 2009.

[179] I. Rivals, L. Personnaz, L. Taing, and M.-C. Potier. Enrichment or depletion of a GO category within a class of genes: which test? *Bioinformatics*, 23(4):401–407, 2007.

[180] A. W. Rives and T. Galitski. Modular organization of cellular networks. *Proc. Natl. Acad. Sci. USA*, 100(3):1128–1133, 2003.

[181] C. Robardet. Constraint-based pattern mining in dynamic graphs. In *ICDM '09: Proceedings of the 9th IEEE International Conference on Data Mining*, pages 950–955. IEEE Computer Society, 2009.

[182] A. Ruepp, B. Brauner, I. Dunger-Kaltenbach, G. Frishman, C. Montrone, M. Stransky, B. Waegele, T. Schmidt, O. N. Doudieu, V. Stümpflen, and H. W. Mewes. CORUM: the comprehensive resource of mammalian protein complexes. *Nucl. Acids Res.*, 36(suppl_1):D646–D650, 2008.

[183] A. Ruepp, A. Zollner, D. Maier, K. Albermann, J. Hani, M. Mokrejs, I. Tetko, U. Güldener, G. Mannhaupt, M. Münsterkötter, and H. W. Mewes. The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucl. Acids Res.*, 32(18):5539–5545, 2004.

[184] R. Rymon. Search through systematic set enumeration. In *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning*, pages 539–550, 1992.

[185] J. M. Santos and M. Embrechts. On the use of the adjusted Rand index as a metric for evaluating supervised classification. In *ICANN '09: Proceedings of the 19th International Conference on Artificial Neural Networks*, pages 175–184. Springer, 2009.

[186] S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

[187] M. Schmid, T. S. Davison, S. R. Henz, U. J. Pape, M. Demar, M. Vingron, B. Schölkopf, D. Weigel, and J. U. Lohmann. A gene expression map of arabidopsis thaliana development. *Nat. Genet.*, 37(5):501–506, 2005.

[188] E. Segal, N. Friedman, D. Koller, and A. Regev. A module map showing conditional activity of expression modules in cancer. *Nat. Genet.*, 36(10):1090–1098, 2004.

[189] E. Segal, H. Wang, and D. Koller. Discovering molecular pathways from protein interaction and gene expression data. *Bioinformatics*, 19(suppl_1):i264–i271, 2003.

[190] R. Shamir, A. Maron-Katz, A. Tanay, C. Linhart, I. Steinfeld, R. Sharan, Y. Shiloh, and R. Elkon. EXPANDER – an integrative program suite for microarray data analysis. *BMC Bioinformatics*, 6(1):232, 2005.

[191] R. Sharan, T. Ideker, B. Kelley, R. Shamir, and R. M. Karp. Identification of protein complexes by comparative analysis of yeast and bacterial protein interaction data. *J. Comput. Biol.*, 12(6):835–846, 2005.

[192] R. Sharan, S. Suthram, R. M. Kelley, T. Kuhn, S. McCuine, P. Uetz, T. Sittler, R. M. Karp, and T. Ideker. Conserved patterns of protein interaction in multiple species. *Proc. Natl. Acad. Sci. USA*, 102(6):1974–1979, 2005.

[193] R. Sharan, I. Ulitsky, and R. Shamir. Network-based prediction of protein function. *Mol. Syst. Biol.*, 3:88, 2007.

[194] B. A. Shoemaker and A. R. Panchenko. Deciphering protein-protein interactions. part I. experimental techniques and databases. *PLoS Comput. Biol.*, 3(3):e42, 2007.

[195] B. A. Shoemaker and A. R. Panchenko. Deciphering protein-protein interactions. part II. computational methods to predict protein and domain interaction partners. *PLoS Comput. Biol.*, 3(4):e43, 2007.

[196] K. Sim, J. Li, V. Gopalkrishnan, and G. Liu. Mining maximal quasi-bicliques to co-cluster stocks and financial ratios for value investment. In *ICDM '06: Proceedings of the 6th International Conference on Data Mining*, pages 1059–1063, 2006.

[197] J. Song and M. Singh. How and when should interactome-derived clusters be used to predict functional modules and protein function? *Bioinformatics*, 25(23):3143–3150, 2009.

[198] V. Spirin and L. A. Mirny. Protein complexes and functional modules in molecular networks. *Proc. Natl. Acad. Sci. USA*, 100(21):12123–12128, 2003.

[199] P. D. Stenson, E. V. Ball, M. Mort, A. D. Phillips, J. A. Shiel, N. S. T. Thomas, S. Abeysinghe, M. Krawczak, and D. N. Cooper. Human Gene Mutation Database (HGMD): 2003 update. *Hum. Mutat.*, 21(6):577–581, 2003.

[200] B. E. Stranger, M. S. Forrest, A. G. Clark, M. J. Minichiello, S. Deutsch, R. Lyle, S. Hunt, B. Kahl, S. E. Antonarakis, S. Tavar, P. Deloukas, and E. T. Dermitzakis. Genome-wide associations of gene expression variation in humans. *PLoS Genet.*, 1(6):e78, 2005.

[201] B. E. Stranger, M. S. Forrest, M. Dunning, C. E. Ingle, C. Beazley, N. Thorne, R. Redon, C. P. Bird, A. de Grassi, C. Lee, C. Tyler-Smith, N. Carter, S. W. Scherer, S. Tavar, P. Deloukas, M. E. Hurles, and E. T. Dermitzakis. Relative impact of nucleotide and copy number variation on gene expression phenotypes. *Science*, 315(5813):848–853, 2007.

[202] A. I. Su, T. Wiltshire, S. Batalov, H. Lapp, K. A. Ching, D. Block, J. Zhang, R. Soden, M. Hayakawa, G. Kreiman, M. P. Cooke, J. R. Walker, and J. B. Hogenesch. A gene atlas of the mouse and human protein-encoding transcriptomes. *Proc. Natl. Acad. Sci. USA*, 101(16):6062–6067, 2004.

[203] J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: dynamic tensor analysis. In *KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 374–383. ACM, 2006.

[204] S. Suthram, J. T. Dudley, A. P. Chiang, R. Chen, T. J. Hastie, and A. J. Butte. Network-based elucidation of human disease similarities reveals common functional modules enriched for pluripotent drug targets. *PLoS Comput Biol*, 6(2):e1000662, 2010.

[205] A. Tanay, R. Sharan, M. Kupiec, and R. Shamir. Revealing modularity and organization in the yeast molecular network by integrated analysis of highly heterogeneous genomewide data. *Proc. Natl. Acad. Sci. USA*, 101(9):2981–2986, 2004.

[206] A. Tanay, R. Sharan, and R. Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18(suppl_1):S136–S144, 2002.

[207] A. Tanay, R. Sharan, and R. Shamir. Biclustering algorithms: A survey. In *Handbook of Computational Molecular Biology.* Chapman and Hall, 2005.

[208] R. L. Tatusov, N. D. Fedorova, J. D. Jackson, A. R. Jacobs, B. Kiryutin, E. V. Koonin, D. M. Krylov, R. Mazumder, S. L. Mekhedov, A. N. Nikolskaya, B. S. Rao, S. Smirnov, A. V. Sverdlov, S. Vasudevan, Y. I. Wolf, J. J. Yin, and D. A. Natale. The COG database: an updated version includes eukaryotes. *BMC Bioinformatics*, 4:41, 2003.

[209] R. L. Tatusov, E. V. Koonin, and D. J. Lipman. A genomic perspective on protein families. *Science*, 278(5338):631–637, 1997.

[210] The International HapMap Consortium. A second generation human haplotype map of over 3.1 million SNPs. *Nature*, 449(7164):851–861, 2007.

[211] F. Towfic, M. H. W. Greenlee, and V. Honavar. Aligning biomolecular networks using modular graph kernels. In *Algorithms in Bioinformatics, 9th International Workshop, WABI 2009, Proceedings*, pages 345–361, 2009.

[212] I. Ulitsky and R. Shamir. Identification of functional modules using network topology and high-throughput data. *BMC Syst. Biol.*, 1:8, 2007.

[213] I. Ulitsky and R. Shamir. Identifying functional modules using expression profiles and confidence-scored protein interactions. *Bioinformatics*, 25(9):1158–1164, 2009.

[214] T. Uno. An efficient algorithm for enumerating pseudo cliques. In *Algorithms and Computation, 18th International Symposium (ISAAC 2007), Proceedings*, pages 402–414, 2007.

[215] T. Uno, M. Kiyomi, and H. Arimura. LCM ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In *FIMI '04, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, 2004.

[216] S. van Dongen. *Graph Clustering by Flow Simulation.* PhD thesis, University of Utrecht, 2000.

[217] I. Van Mechelen, H. H. Bock, and P. D. Boeck. Two-mode clustering methods: a structured overview. *Stat. Methods Med. Res.*, 13(5):363–394, 2004.

[218] O. Vincent and M. Carlson. Gal83 mediates the interaction of the Snf1 kinase complex with the transcription activator Sip4. *EMBO J.*, 18(23):6672–6681, 1999.

[219] C. von Mering, L. J. Jensen, M. Kuhn, S. Chaffron, T. Doerks, B. Kruger, B. Snel, and P. Bork. STRING 7 – recent developments in the integration and prediction of protein interactions. *Nucl. Acids Res.*, 35(suppl_1):D358–D362, 2007.

[220] C. von Mering, L. J. Jensen, B. Snel, S. D. Hooper, M. Krupp, M. Foglierini, N. Jouffre, M. A. Huynen, and P. Bork. STRING: known and predicted protein-protein associations, integrated and transferred across organisms. *Nucl. Acids Res.*, 33(suppl_1):D433–D437, 2005.

[221] J. Wang, J. Han, and J. Pei. CLOSET+: searching for the best strategies for mining frequent closed itemsets. In *KDD '03: Proceedings of the9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 236–245. ACM, 2003.

[222] P. Wang, C. Domeniconi, and K. B. Laskey. Latent dirichlet bayesian co-clustering. In *ECML PKDD '09: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 522–537. Springer, 2009.

[223] Z. Wang, M. Gerstein, and M. Snyder. RNA-Seq: a revolutionary tool for transcriptomics. *Nat. Rev. Genet.*, 10(1):57–63, 2009.

[224] M. J. Warrens. On the equivalence of cohen's kappa and the hubert-arabie adjusted rand index. *Journal of Classification*, 25(2):177–183, 2008.

[225] T. Washio and H. Motoda. State of the art of graph-based data mining. *SIGKDD Explor. Newsl.*, 5(1):59–68, 2003.

[226] D. S. Wishart, C. Knox, A. C. Guo, S. Shrivastava, M. Hassanali, P. Stothard, Z. Chang, and J. Woolsey. DrugBank: a comprehensive resource for in silico drug discovery and exploration. *Nucleic Acids Res.*, 34(Database issue):D668–D672, 2006.

[227] T. Wolf, B. Brors, T. Hofmann, and E. Georgii. Global biclustering of microarray data. In *ICDMW '06: Proceedings of the 6th IEEE International Conference on Data Mining – Workshops*, pages 125–129. IEEE Computer Society, 2006.

[228] Z. Wu, R. A. Irizarry, R. Gentleman, F. Martinez-Murillo, and F. Spencer. A model-based background adjustment for oligonucleotide expression arrays. *Journal of the American Statistical Association*, 99(468):909–917, 2004.

[229] A. E. Wurmser, T. K. Sato, and S. D. Emr. New component of the vacuolar class C-Vps complex couples nucleotide exchange on the Ypt7 GTPase to SNARE-dependent docking and fusion. *J. Cell Biol.*, 151(3):551–562, 2000.

[230] I. Xenarios, D. W. Rice, L. Salwinski, M. K. Baron, E. M. Marcotte, and D. Eisenberg. DIP: the database of interacting proteins. *Nucleic Acids Res.*, 28(1):289–291, 2000.

[231] T. Yamada, M. Kanehisa, and S. Goto. Extraction of phylogenetic network modules from the metabolic network. *BMC Bioinformatics*, 7:130, 2006.

[232] C. Yan, J. G. Burleigh, and O. Eulenstein. Identifying optimal incomplete phylogenetic data sets from sequence databases. *Mol. Phylogenet. Evol.*, 35(3):528–535, 2005.

[233] X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. In *ICDM '02: Proceedings of the 2nd IEEE International Conference on Data Mining*, pages 721–724. IEEE Computer Society, 2002.

[234] X. Yan and J. Han. CloseGraph: mining closed frequent graph patterns. In *KDD '03: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 286–295. ACM, 2003.

[235] X. Yan, M. R. Mehan, Y. Huang, M. S. Waterman, P. S. Yu, and X. J. Zhou. A graph-based approach to systematically reconstruct human transcriptional regulatory modules. *Bioinformatics*, 23(13):i577–i586, 2007.

[236] X. Yan, X. J. Zhou, and J. Han. Mining closed relational graphs with connectivity constraints. In *KDD '05: Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 324–333. ACM, 2005.

[237] K. Y. Yeung and W. L. Ruzzo. Principal component analysis for clustering gene expression data. *Bioinformatics*, 17(9):763–774, 2001.

[238] K. Y. Yip, H. Yu, P. M. Kim, M. Schultz, and M. Gerstein. The tYNA platform for comparative interactomics: a web tool for managing, comparing and mining multiple networks. *Bioinformatics*, 22(23):2968–2970, 2006.

[239] N. Yosef, Z. Yakhini, A. Tsalenko, V. Kristensen, A.-L. Børresen-Dale, E. Ruppin, and R. Sharan. A supervised approach for identifying discriminating genotype patterns and its application to breast cancer data. *Bioinformatics*, 23(2):e91–e98, 2007.

[240] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. In *KDD '97: Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, pages 283–286. AAAI Press, 1997.

[241] M. J. Zaki, M. Peters, I. Assent, and T. Seidl. Clicks: An effective algorithm for mining subspace clusters in categorical datasets. *Data Knowl. Eng.*, 60(1):51–70, 2007.

[242] Z. Zeng, J. Wang, L. Zhou, and G. Karypis. Coherent closed quasi-clique discovery from large dense graph databases. In *KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 797–802. ACM, 2006.

[243] L. Zhao and M. J. Zaki. TRICLUSTER: an effective algorithm for mining coherent clusters in 3D microarray data. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pages 694–705. ACM, 2005.

[244] J. Zheng, X. Yang, J. M. Harrell, S. Ryzhikov, E. H. Shim, K. Lykke-Andersen, N. Wei, H. Sun, R. Kobayashi, and H. Zhang. CAND1 binds to unneddylated CUL1 and regulates the formation of SCF ubiquitin E3 ligase complex. *Mol. Cell*, 10(6):1519–1526, 2002.

[245] F. Zhu, X. Yan, J. Han, and P. S. Yu. gPrune: A constraint pushing framework for graph pattern mining. In *PAKDD '07: Proceedings of the 11th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pages 388–400. Springer, 2007.

List of Figures

3.1 A weighted graph with four nodes. 16
3.2 A tree structure. 17
3.3 Example relations. 17

5.1 Different cluster finding concepts: graph partitioning versus cluster enumeration. 30
5.2 Schematic view of typical overlap scenarios in protein complex analysis. 31
5.3 Integration of profile data into protein interaction analysis. 31
5.4 Motivating example for module enumeration strategy. 34
5.5 Illustration of reverse search with the odd-even output method. 44
5.6 Minimum relative degree versus density. 50
5.7 Anti-monotonicity of consistency constraints. 58
5.8 Example reduction of a module leading to disconnected ancestor modules. 60

6.1 Generalization of module mining to two-way and higher-order data. 64
6.2 Illustration of the n-set definition in the relational approach and in our dense cluster approach. 66
6.3 Visualization of a three-way cluster. 70
6.4 DCE runtime measurements for artifical data in dependence of different parameters. 85
6.5 Runtime comparison of DCE and other set enumeration strategies. 90
6.6 Top-ranking cluster for email traffic data. 93

7.1 Schematic view of agglomerative clustering. 98
7.2 Comparison of agglomerative clustering strategies in two-way data. 99
7.3 Classical symmetric scenario for hierarchical clustering. 100
7.4 Hierarchical biclustering scenario. 103

9.1 Comparative precision-recall analysis for yeast modules. 118
9.2 Precision-recall curves for overlapping interactions in yeast modules. 119
9.3 Comparative precision-recall analysis for human modules. 121
9.4 Precision-recall curves for overlapping interactions in human modules. 121
9.5 Phenotype-associated yeast modules. 124
9.6 Yeast complexes retrieved by DME and their overlap with conserved DME modules. 125
9.7 Tissue-specific modules in human. 127

12.1 Illustration of a SNP-transcript association. 146
12.2 Comparative evaluation of different strategies to discover functional SNPs. 149

14.1 Connection subgraph discovered in a protein interaction network of zebrafish. 159

List of Tables

6.1 Performance analysis of DCE and its extensions. 88

9.1 Module statistics of the comparative analysis for yeast. 120
9.2 Module statistics of the comparative analysis on the human network. 122
9.3 Results of DME experiments with constraints. 123

10.1 Comparative evaluation on coexpression data. 134

11.1 Major tissues and their frequency among the 79 experiments in the *A. thaliana* data. 138
11.2 Comparative analysis of biclustering and global sample clustering. 138
11.3 Sample-based evaluation of SAMBA biclustering results. 141
11.4 Gene function evaluation of biclusters. 142
11.5 Enriched GO terms for pure tissue clusters. 144

12.1 Enumerative bicluster SNPs with functional annotation. 150
12.2 Hierarchical bicluster SNPs with functional annotation. 151

List of Algorithms

1 Dense module enumeration. 39
2 Enumeration of locally maximal dense modules. 46
3 Enumeration of dense leaf modules. 46

4 Pseudocode of general dense cluster enumeration. 72

Publications

Publications related to this thesis are marked with •.

Journal Articles

• **E. Georgii**, K. Tsuda, and B. Schölkopf. Multi-way set enumeration in weight tensors. *Machine Learning, Special issue on Mining and Learning with Graphs and Relations.* To appear.

• **E. Georgii**, S. Dietmann, T. Uno, P. Pagel, and K. Tsuda. Enumeration of condition-dependent dense modules in protein interaction networks. *Bioinformatics*, 25(7):933–940, 2009.

• S. Dietmann, **E. Georgii**, A. Antonov, K. Tsuda, and H.-W. Mewes. The DICS repository: module-assisted analysis of disease-related gene lists. *Bioinformatics*, 25(6):830–831, 2009.

E. Georgii, L. Richter, U. Rückert, and S. Kramer. Analyzing microarray data using quantitative association rules. *Bioinformatics*, 21(Suppl. 2):ii123–ii129, 2005.

Papers at International Workshops

• **E. Georgii**, K. Tsuda, and B. Schölkopf. Multi-way set enumeration in real-valued tensors. *DMMT '09: Proceedings of the KDD '09 Workshop on Data Mining using Matrices and Tensors*, pp. 32–41, ACM, 2009.

T. Wolf, B. Brors, T. Hofmann, and **E. Georgii**. Global biclustering of microarray data. *ICDMW '06: Proceedings of the 6th IEEE International Conference on Data Mining – Workshops*, pp. 125–129, IEEE Computer Society, 2006.

Abstracts

• **E. Georgii**, S. Dietmann, T. Uno, P. Pagel, and K. Tsuda. Mining expression-dependent modules in the human interaction network. *BMC Bioinformatics* 8(Suppl. 8):S4, 2007. *[Oral presentation and best poster award at the 3rd ISCB Student Council Symposium jointly held with ISMB '07.]*

Curriculum Vitae

Personal Information

Name	Elisabeth Georgii
Date of birth	May 17, 1980
Place of birth	Bielefeld, Germany
Citizenship	German
Email	`elisabeth.georgii@tkk.fi`

Education and Employment

09/1990 - 06/1999	*Graf-Rasso-Gymnasium Fürstenfeldbruck, Germany* (secondary school)
06/1999	Abitur
10/1999 - 09/2000	Basic studies in Computer Science at *Ludwig-Maximilians-Universität München*, Germany
10/2000 - 05/2002	Basic studies in Bioinformatics at *Ludwig-Maximilians-Universität München* and *Technische Universität München*, Germany
05/2002	Intermediate diploma in Bioinformatics
05/2002 - 03/2005	Advanced studies in Bioinformatics at *Ludwig-Maximilians-Universität München* and *Technische Universität München*
03/2005	Diploma in Bioinformatics; Diploma thesis: "Quantitative association rules", supervised by Prof. Dr. Stefan Kramer, *Technische Universität München*
08/2005 - 06/2006	Research associate at *Technische Universität Darmstadt*, Germany, supervised by Prof. Dr. Thomas Hofmann
07/2006 - 12/2009	PhD student at *Max Planck Campus Tübingen*, Germany, supervised by Dr. Koji Tsuda, *Max Planck Institute for Biological Cybernetics*, and Dr. Gunnar Rätsch, *Friedrich Miescher Laboratory of the Max Planck Society*
since 01/2010	Researcher at *Aalto University, School of Science and Technology*, Finland

Lebenslauf

Persönliche Daten

Name	Elisabeth Georgii
Geburtstag	17. Mai 1980
Geburtsort	Bielefeld
Staatsangehörigkeit	deutsch
E-mail	`elisabeth.georgii@tkk.fi`

Ausbildung und Beruf

09/1990 - 06/1999	*Graf-Rasso-Gymnasium Fürstenfeldbruck*
06/1999	Abitur
10/1999 - 09/2000	Informatik-Grundstudium an der *Ludwig-Maximilians-Universität München*
10/2000 - 05/2002	Bioinformatik-Grundstudium an der *Ludwig-Maximilians-Universität München* und der *Technischen Universität München*
05/2002	Vordiplom in Bioinformatik
05/2002 - 03/2005	Bioinformatik-Hauptstudium an der *Ludwig-Maximilians-Universität München* und der *Technischen Universität München*
03/2005	Diplom in Bioinformatik; Diplomarbeit "Quantitative association rules" bei Prof. Dr. Stefan Kramer, *Technische Universität München*
08/2005 - 06/2006	Wissenschaftliche Mitarbeiterin an der *Technischen Universität Darmstadt* bei Prof. Dr. Thomas Hofmann
07/2006 - 12/2009	Doktorandin am *Max-Planck-Campus Tübingen* bei Dr. Koji Tsuda, *Max-Planck-Institut für Biologische Kybernetik*, und Dr. Gunnar Rätsch, *Friedrich-Miescher-Laboratorium der Max-Planck-Gesellschaft*
seit 01/2010	Angestellte an der *Aalto University, School of Science and Technology*, Finnland