JCell

A Java Framework for Inferring Genetic Networks

- Manual and Documentation -

Christian Spieth

WSI-2005-07

ISSN 0946-3852

Centre for Bioinformatics Tübingen (ZBIT),
Eberhard-Karls-University Tübingen,
Sand 1, 72076 Tübingen, Germany

JCell User Guide
JCell: A Java based framework to reconstruct genetic interactions

Dipl.-Ing., MSc Christian Spieth

Published $ Date: 2004/07/07 09:16:14 $

Updated $ Date: 2005/04/26 14:08:41 $

http:\\www.jcell.de

Contents

Chapter 1

Preface

JCell is a Java based application to simulate intra-cellular processes based on gene regulatory mechanisms. The project is aiming to model and reconstruct genetic interactions in silico, and helps researchers to gain new insights in the organizational structure of an organism.

This manual is aimed to give an overview over the features of JCell. It is divided into several chapter as follows: ” Chapter Introduction briefly introduces the problem of inferring regulatory systems and artificial embryology. Further on, a motivation is given. ” An overview over the biological background can be found in chapter Biology. This chapter will introduce the basic concepts of genetics together with a brief historical digression. ” Detailed information about each menu item of JCell can be found in chapter Menu description. ” The mathematical founding of the implemented algorithms and inference strategies are explained in chapter Algorithms. ” And the last chapter (Command Line Options) lists all command line parameters for JCell, which can be used for running the program in batch mode.

JCell project is developed at the Centre for Bioinformatics of the University of Tübingen (ZBIT). The main project website is www.jcell.org (http://www.jcell.de)

Chapter 2

Introduction

2.1 Motivation

In the past few years, DNA microarrays have become one of the key techniques in the area of gene expression analysis. This technology enables the monitoring of thousands of genes in parallel and can therefore be used as a powerful tool to understand the regulatory mechanisms of gene expression in a cell. Gene regulatory networks (GRNs) represent the dependencies of the different actors in a cell operating at the genetic level. They dynamically determine the level of gene expression for each gene in the genome by controlling whether a gene will be transcribed into RNA. A simple GRN consists of one or more input signalling pathways, several target genes, and the RNA and proteins produced from those target genes. In addition, such networks often include dynamic feedback loops that provide for further regulation of network regulation activities and output. In order to understand the underlying structures of activities and interactions of intra-cellular processes one has to understand the dependencies of gene products and their impact on the expression of other genes. Therefore, finding a GRN for a specific biological process would explain this process from a logical point of view. However, due to the huge number of components within the regulatory system, a large amount of experimental data is needed to infer genome-wide networks. This requirement is almost impracticable to meet today, because of the high costs of these experiments and due to the fact that the investigated processes are too short and do not allow for more sampling points in time. To bypass this problem, additional data has to be acquired like knock-out, over-expression experiment data or data sets with different starting conditions that decrease the uncertainties in the system. JCell is a framework for simulating GRNs. It is completely implemented in Java and can be used for two different applications:

- reverse-engineering and inferring regulatory mechanisms based on the evaluation of given biological and medical data coming from DNA microarray experiments, and

- simulating cell growth and mitosis by finding GRNs suitable for a given problem (e.g. limited growth).

2.2 Inference

Researchers are interested in understanding the mechanisms of gene regulatory processes and therefore in inferring the underlying networks. This has recently become one of the major topics in bioinformatics due to the increased amount of data available. Gene regulatory network analysis exploits massively parallel measurements of interacting biochemicals, namely with DNA microarray techniques. The measurements at different states of the cell can be used for studying the relationships between each component of cellular processes by mathematical modeling of the dependencies in the data set. There are different types of mathematical methods implemented in JCell for simulating GRNs like

- Random boolean networks (RBN),
- quantitative Models,
- Weight matrices,
- pseudolinear weight matrices,
- S-Systems,
- H-Systems, and
- arbitrary differential equations.

The parameters of each model are evaluated either by optimization with evolutionary algorithms or by straight-forward heuristics, if available. The main focus of the current research is on GRNs related to immune-specific diseases, as JCell is developed as part of the TuebinGENome (*http://www.tuebingenome.de*) project of the NGFN - German National Genome Research Network (*http://www.ngfn.de*) in cooperation with the University Hospital of Tübingen.

2.3 Cell Simulation and Morphogenesis

The translation of the dynamics of gene regulatory networks in terms of pattern and form is a central problem, not only for biology, but also for Artificial Life. The translation of genetic information into shapes and patterns is what links genetics to morphology. Our goal is to use gene regulatory networks as genotypic representation for phenotypic shapes that are subject to evolution. We want to evolve the parameters of the gene regulatory networks to optimize shapes for design optimization through Evolutionary Algorithms.

2.4 Limited Growth

A basic property of an organism is that although it originates from a single cell and needs to grow and multiply, it finally needs to stop growing and to remain stable. Therefore we performed experiments to evolve the behavior of limited growth by optimizing the gene regulatory genotype of the organism by the use of Evolutionary Algorithms. We could show that a gene regulatory network producing continuous metabolites, controlling the behavior of the cells in the organism, together with diffusion between cells, was able to show the desired behavior, even if only two genes were simulated.

2.4.1 Ability of Self-Repair

In another experiment we wounded the organisms and tried to evolve the ability of self-repair. With an additional metabolite generated by dying cells we were also able to evolve the behavior of self-repair. In some experiments the organisms were severed by the random wounding. Nevertheless, both parts grew to the original size. A behavior that can also be observed for some organisms in nature like hydras or flatworms. In this experiment also cell differentiation occurs, although this was not an objective in the evolutionary process. In the stable configuration of the organism there are cells in two distinct states: 1. Low activity of all metabolites and 2. High activity of metabolites a and c. Cells of type 2 are usually splitting and generating new cells but by diffusion the cells of type 1 control the cells of type 2 and prevent them from actually growing. When the cell is wounded, the additional metabolite d causes the cells of type 1 to change into cell type 2 and the organism grows new cells until a new equilibrium is reached.

Chapter 3

Biology

From the 1950's onwards, several advances in the field of genetics were made to the understanding of the function of DNA, as there were

- the replication of DNA,
- protein synthesis and the role of messenger RNA (mRNA), transfer RNA (tRNA) and ribosomes
- and the regulation of gene activities.

These important steps lead to the image of DNA functionality we have today. It is interesting to note that Mendel's laws about heredity from 1866 and the theories of Watson and Crick from the 50's are still valid these days, although genetics is a fast developing area that is subject to quickly changes.

3.0.2 DNA

The famous double helix of the DNA is the foundation of all life on this planet. As we know today, organisms use DNA as their genetic material. An exception to this general rule are viruses, which use RNA (ribonucleic acid) instead of DNA for inheritance. The complete copy of the DNA in an organism is called the organism's genome. The genome of a diploid, i.e. having two sets of chromosomes[1], organism is its set of chromosomes and all the genes of which it consists of. The DNA is a polymer of repeating units called nucleotides. A generic nucleotide has three components: a deoxyribose sugar joined to a phosphate group and to a nitrogenous base. The sugar and the phosphate group have mainly structural purposes. Only

[1] A chromosome is a self-replicating molecule in the cell nucleus of all plants and animals through which characteristics are inherited. Each organism of a species normally has a characteristic number of chromosomes in its cells, 46 being the number normally present in a human being, including the two (XX or XY) which determine the sex of the organism.

the bases are of interest in this thesis, therefore the other parts and functions of the DNA will not been explained.

Four different nitrogenous bases are found in DNA: Two are derivatives of *purine* [adenine (A) and guanine (G)] and two of the chemical class of *pyrimidine* [cytosine (C) and thymine (T)]. The two strands in the double helix are formed by the chemical interaction between bases. These bases exhibit hydrogen bonding and thus serve as the rungs on the twisted ladder of DNA that hold the strands together. Nitrogenous bases interact in a specific way. G only bonds with C, whereas A only bonds with T (or U in RNA[2]). Because of this specific behavior of the bases, one strand in the DNA is the reverse of the second strand; hence they are called *complementary*. Furthermore, the doublestranded DNA has two grooves, the major and minor grooves, that twist around its exterior. The major groove is larger than the minor because of the chemical nature of the sugar-phosphate-structure.

3.0.3 From DNA to Proteins

In molecular genetics we call the part of DNA that carries the information about a single protein a gene. The sequence of nucleotide bases of this gene leads to the production of specific amino acids and thus to a specific protein. Each amino acid is specified by a triplet of nucleotide bases, a so called *codon*. With four different base pairs (A, C, G and T) and three bases at a time we get $4^3 = 64$ possible combinations for amino acids. Many of these combinations are synonyms, i.e. more than one codon represents the same acid. AGA, AGG, AGT, AGC, TCA and TCG all stand for the same amino acid: Serine (Ser). In this example, there are six combinations expressing serine, other acids are represented by fewer codons (e.g. methionine (Met) is only represented by the single combination TAC). In addition to the 20 amino acids, there are three combinations which can be interpreted as termination sequences that stop the evaluation of the base sequence. A table of all protein codons can be found in 3.1.

DNA is not translated into proteins directly, however. Genes are rather expressed through the production of messenger RNA (mRNA). mRNA is an intermediate product in building proteins. It is synthesized in the same way as DNA replicates, with two important differences: It copies only one strand of the DNA and T (thymine) is replaced by U (uracil). This process is called *transcription*.

To produce a protein from mRNA, the mRNA has to be transported to the sites of protein synthesis which are in the cytoplasm of a cell. Hence the RNA is called messenger RNA. This is common to both types of cells, it happens in eucaryotic[3] as well as in prokaryotic[4] cells. The difference is that in eucaryotes, mRNA has to pass

[2]RNA is different from DNA in the chemical formula, i.e. RNA contains the five carbon sugar ribose instead of deoxyribose found in DNA.

[3]Eucaryotes are organisms whose DNA is enclosed in a nucleus.

[4]Procaryotes are organisms with no enclosed nucleus.

Table 3.1: Table of all 20 amino acids.

Alanine (Ala — A)	Glycine (Gly — G)	Proline (Pro — P)
Arginine (Arg — R)	Histidine (His — H)	Serine (Ser — S)
Asparagine (Asn — N)	Isoleucine (Ile — I)	Threonine (Thr —T)
Aspartic acid (Asp — D)	Leucine (Leu — L)	Tryptophan (Trp — W)
Cysteine (Cys — C)	Lysine (Lys — K)	Tyrosine (Tyr — Y)
Glutamic acid (Glu — E)	Methionine (Met — M)	Valine (Val — V)
Glutamine (Gln — Q)	Phenylalanine (Phe — F)	

the membrane around the nucleus. Ribosomes are cytoplasmic organelles found in procaryotes and eukaryotes. They are large complexes of proteins and three (procaryotes) or four (eucaryotes) rRNA (ribosomal ribonucleic acid) molecules called subunits made in the nucleolus. Once mRNA is bound to ribosomes in the cytoplasm, the actual conversion into a protein begins. Transfer RNA (tRNA) brings the correct amino acids in the proper sequence to the synthesis site, where they combine to form a protein.

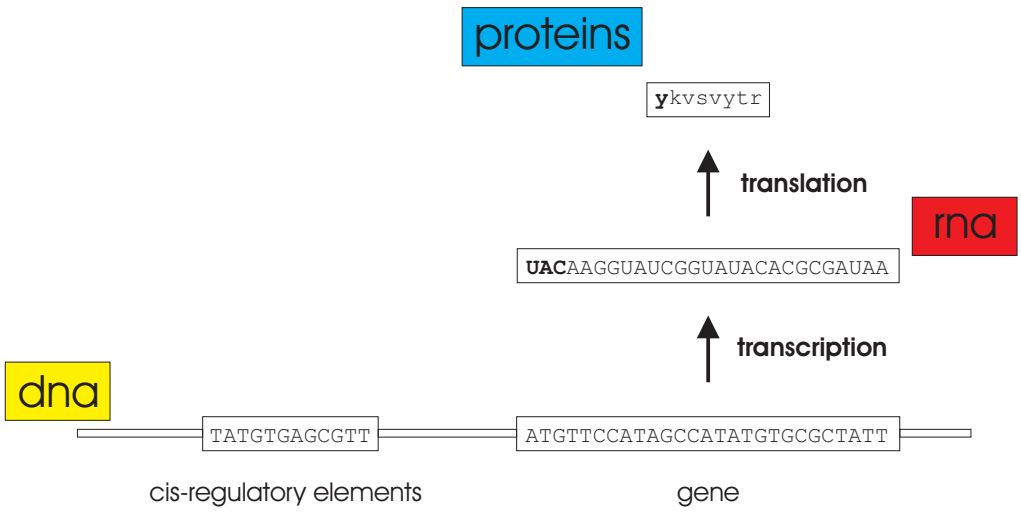


Figure 3.1: From DNA to protein

Figure 3.1 shows the sequence of actions in the synthesis of proteins, where DNA is transcribed into RNA and then translated into a protein. A more detailed explanation can be found in [7].

3.0.4 Gene Regulation

Since every gene contains information about a specific protein and this gene could be transcribed at any time, there must be a mechanism that controls the gene expression. Otherwise proteins would be produced whether or not they were required and in random quantity. This is especially important in multicellular organisms where cells take on specialized tasks. Every cell in a human being carries the gene for producing insulin[5], but only cells in the pancreas actually do it. Each cell in our body therefore produces different proteins because different genes are switched on or off in each case. Another case of gene regulation can be found in the different stages of the development of an organism. A seed of a plant needs special enzymes to metabolize its food reserves while germinating. Later, proteins for photosynthesis become more important which are of no use in the first phase of growing. This means that the right genes have to be switched on or off the appropriate time to control development.

But how does a gene know when it has to be expressed? The transcription process starts, when a ribosomal unit binds at special regions at the beginning of a gene. This region is called a promoter. To enable the ribosomal unit to bind, some other proteins have to form a complex molecule (a so called transcription factor), which itself docks to transcription sites on the DNA. Thus, these transcription factors trigger the expression of the corresponding gene.

Figure 3.2 shows schematically the process of transcription, translation, and regulation. As one can see, a gene is expressed and translated into a protein due to the signals reaching the cell (Signal A and B). The new protein performs the biological tasks within the cell by reacting with other proteins or biochemicals. However, the protein might also serve as a signal itself, regulating the expression of other genes. Therefore, genes regulate other genes with their expressed proteins. This is in general known as a gene regulatory system. The entirety of all gene regulating dependencies is called a gene regulatory network.

[5]Insulin is a polypeptide hormone secreted by the beta cells of the islets of Langerhans, a specific groups of cells in the pancreas.

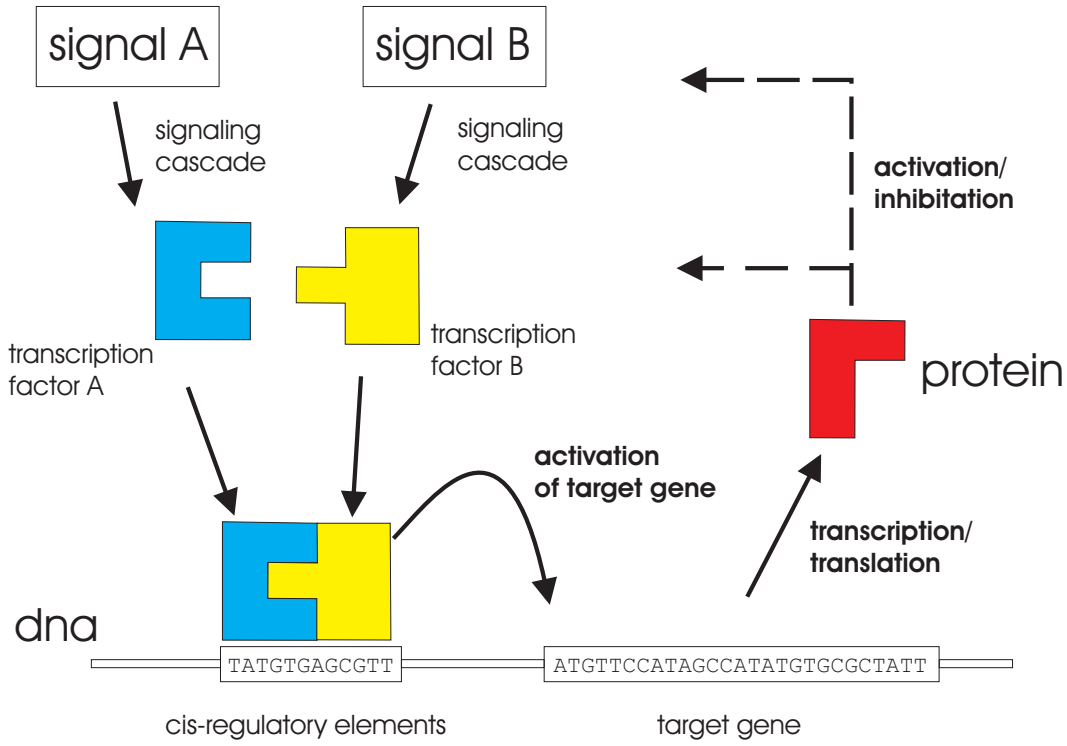


Figure 3.2: Schematic view of genetic regulation.

Chapter 4

Algorithms

This chapter briefly introduces the problem of inferring dynamic systems. The interested user will find descriptions of the implemented algorithms and mathematical models to simulate regulatory networks.

4.1 Inference

Goal of the inference is to determine the topological structure of the regulatory network as well as the kinetic parameters of the underlying mathematical model. The topology shows the dependencies between the components (for example genes) of the system. This graph represents the qualitative view of the regulatory network. The mathematical model on the other hand represents the quantitative view of the system. With the parameters of the model a researcher is able to simulate the whole system or only parts of it to understand the interactions and to find ways to modify the system to reach a desired state or behavior. The main idea behind the whole process is to explain the experimental data ba finding a mathematical model, which results in a similar data set or time dynamics, respectively. In general, this is done by simulating the artificial model to get a set of time series data. After that, this data set is compared to the experimental data set to determine the differences. These differences are then incorporated into a second iteration of the process to modify the system in such a way that it better fits the experiment data. The actual computational inference process is divided into several phases, which are performed successively and might require to go back to a previous state to gain optimal solutions.

- Preprocess expression data. In most cases the experimental expression data has to be preprocessed to reduce the dimensionality of the problem, i.e. of the solution space. This can be done by statistical methods, where components (for example genes) are filtered that do not seem to participate in the regulatory process of interest. For example, a gene that does not change its

expression level over time is not very likely to be involved in the biological process that is under investigation. Another way to preprocess data is to cluster the time series, i.e. to group similar time dynamics. The underlying idea is that genes that show similar expression profiles are likely to be co-regulated, i.e. depending on the same regulatory mechanisms. To reduce the dimensionality, each group can then be replaced by a representative of the group (cluster). Thus, the number of components is reduced drastically.

- Import additional biological data. The problem of finding the correct topology together with determining the correct model parameters is very difficult. The solution space of the problem grows very fast with the number of components (depending on the mathematical model). Therefore, it is important to incorporate additional knowledge to reduce the complexity of the problem. This can be done by including information about the network structure or by incorporating known dependencies between components of the system. The information can be retrieved from public databases or from the researchers themselves. An example for a public database is KEGG, which lists known pathway topologies. This can be done within JCell, which has interfaces to public databases.

- Select appropriate model. The choice of the mathematical model is a crucial part of the whole inference process. Each model has advantages as well as disadvantages as described in the next section. For example, Boolean Networks show a low computational complexity, which makes them perfect to simulate huge systems. Unfortunately, they show the same time a very high level of abstraction, i.e. they do not really resemble biological systems. In "plain vanilla" Boolean Networks there are only two states in which a gene can be in: on (true) or off (false). Obviously, this is not the case for real gene expression. On the other side, arbitrary differential equations (DE) are the closest approximation of regulatory networks, for many biological process have differential equations been found describing correctly the dependencies between the components. But DEs are very difficult to handle with computational algorithms. One class of algorithms able to examine DEs is Genetic Programming.

- Initialize model parameters. The parameters of the system have to be initialized before starting the actual inference process. This can be done either randomly or by incorporating the additional biological knowledge imported before. In the latter case the internal matrices representing the structure of the network and the corresponding parameters are initialized with the values of the imported information.

- Find correct parameters. In the solution phase the program tries to determine the correct topology and the corresponding model parameters. This is either

done by using optimization algorithms like Evolutionary Algorithms or by direct solving using special heuristics. To evaluate the quality of a solution found in the inference process, the fitness of this model is determined. This can be for example done by calculating the euclidian difference between the time series vectors of the experimental data and the vectors of the simulated data.

- Hypothesize a network topology. After the parameters have been found – either by finding the correct values or by hitting a termination criterion like the maximum number of fitness evaluations in case of EAs – the model has to be "translated" into a network hypothesis. This is easy for simple model like RBNs or weight matrices, where the dependencies can be directly transferred into a graph.

- Verification. The last phase of the inference is the verification and validation of the results with biological or medical "wet" experiments. This phase is very important because the user will find multiple set of parameters in almost every in silico inference process and needs to determine the true system by testing the hypothesis with real world experiments.

4.2 Mathematical Models

Researchers are interested in understanding the mechanisms of gene regulatory processes and therefore in simulating the underlying networks. This has recently become one of the major topics in bioinformatics due to the increased amount of data available. The following section briefly describes the different mathematical models that can be used to simulate regulatory systems.

On an abstract level, the behavior of a cell is represented by a gene regulatory network of N genes. Each gene gi produces a certain amount of RNA xi when expressed and therefore changes the concentration of this RNA level over time:

$$\vec{x}(t+1) = h(\vec{x}(t))\,,\ \vec{x}(t) = (x_1, \cdots, x_n) \tag{4.1}$$

where h is a function that describes the changing.
There are different types of mathematical methods for simulating GRNs like

- Random boolean networks (RBN),

- quantitative Models,

- Weight matrices,

- pseudolinear weight matrices,

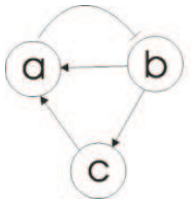
- S-Systems,
- H-Systems, and
- arbitrary differential equations.

The following sections describe the mentioned models and lists corresponding publications if available.

4.2.1 Random Boolean networks (RBN)

One kind of model to simulate regulatory systems found in the literature are Boolean or Random Boolean Networks (RBN) [5, 18]. In Boolean Networks gene expression levels can be in one of two states: either 1 (on) or 0 (off). The quantitative level of expression is not considered.

According to this model the network is represented as an oriented graph $G = (V, F)$, whose nodes V represent element of the network, and F defines a topology of edges between the nodes and a set of boolean functions. A node may represent either a gene or a biological stimulus, where a stimulus is any relevant physical or chemical factor which influences the network and is itself not a gene or gene product. A node has an associated steady-state expression level x, representing the amount of gene product or the amount of stimulus present in the cell. This level is approximated as high or low and represented by the binary value 1 or 0, respectively. Network behavior over time is modeled as a sequence of discrete synchronous steps. The set of boolean functions assigned to the nodes defines the value of a node on the next step depending on values of other nodes, which influence it. The functions f are uniquely defined using truth tables. An edge directed from one node to another represents the influence of the first gene or stimulus on that of the second, so that the expression level of a node v is a Boolean function f of the levels of the nodes in the network which connect (have a directed edge) to v.



		0;0	0;1	1;0	1;1
a	b;c	0	1	1	0
b	a;a	1	0	0	1
c	b;b	0	1	1	1

Figure 4.1: Boolean Network.

Figure 4.1 gives an example of a simple boolean network and associated truth table. This example shows a network of three nodes – a, b and c, respectively. As one can see, expression of c directly depends on expression of b, which directly depends

on a. However, influence of b and c on a is more complex. For example, high level of expression of both b and c leads to inhibition of a.

4.2.2 Qualitative models

In contrast to discrete methods like RBNs, qualitative network models allow for multiple levels of gene regulation. An example for this kind of approach is given by Thieffry and Thomas in [15].

Qualitative models are not implemented in JCell and are therefore not discussed here.

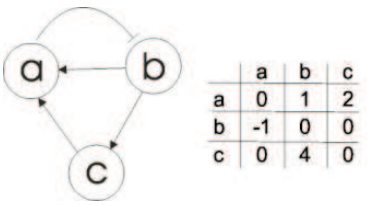
4.2.3 Weight matrices

Quantitative models like the weighted matrix model by Weaver et al. [17] consider the continuous level of gene expression. Inference methods based on linear models for gene regulatory networks are given for example in [1] and [3].

Weight matrices are well known in bioinformatics in the field of sequence alignment. Several papers have been published on this topic. They main idea of weight matrices is having a matrix of values representing numerical relationships between genes. They have the form of the following equation:

$$r_i(t) = \sum_j w_{ij} x_j(t) \tag{4.2}$$

The example from above is modeled with weight matrices as shown in the following figure. Note that there are negative values whenever a gene is inhibited by another gene. Weight matrices have the advantage of modeling interactions quantitative in comparison to Boolean Networks where only qualitative relationships are considered. However, weight matrices are of linear type, i.e. there are only linear dependencies between components in the system.



	a	b	c
a	0	1	2
b	-1	0	0
c	0	4	0

Figure 4.2: Weight Matrix.

4.2.4 Pseudolinear weight matrices

Weaver et al. enhanced the standard weight matrix that it resembles artificial neural networks with sigmoidal activation functions in the nodes (genes) to overcome the problem of linearity. To further improve the model, we extended the model of Weaver et al. in such a way that not only nodes have sigmoidal activation functions but also the connecting edges. Thus, we created a flexible model without the limitation of linearity.

4.2.5 Bayesian Networks

Bayesian networks or belief networks are a type of models for representing uncertainty in our knowledge. They are stochastically, i.e. they use probability theory to manage uncertainty by explicitly representing the conditional dependencies between the different knowledge components. The network is modeled with a directed acyclic graph of dependence structure between multiple interacting quantities where the nodes represent random variables and the edges indicate conditional dependencies.

For a directed model, we must specify the Conditional Probability Distribution at each node. To give a simple example, we consider only discrete variables in the following. If the variables are discrete they can be represented as a table, which lists the probability that the child node takes on each of its different values for each combination of values of its parents. Consider the following example, in which all nodes are binary, i.e., have two possible values, which are denoted by T (true) and F (false), respectively. The following graph shows the simple case for modeling the trivial example of a meadow.

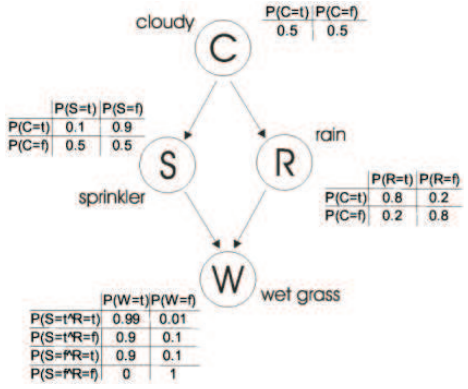


Figure 4.3: Bayesian Network.

Looking at the graph it is obvious that the event "grass is wet" (W=true) has two possible causes: either the water sprinkler is on (S=true) or it is raining (R=true).

The strength of this relationship is shown in the tables. For example, we see that P(W=true — S=true, R=false) = 0.9, and hence, P(W=false — S=true, R=false) = 1 - 0.9 = 0.1, since each row must sum to one.

4.2.6 H-Systems

H-Systems are another type of enhanced weight matrices, where an additional term ensures non-linearity in the model. They have the form of:

$$\frac{dx_i(t)}{dt} = c_i + \sum_k a_{ik}x_k(t) + x_i(t)\sum_k b_{ik}x_k(t) \tag{4.3}$$

4.2.7 S-Systems

S-Systems employ a general formalism, which allows for capturing the non-linearity and general dynamics of the gene regulation. S-Systems are a type of power-law formalism, which has been suggested by [10] and can be described by a set of nonlinear differential equations:

$$\frac{dx_i(t)}{dt} = \alpha_i \prod_{j=1}^{N} x_j(t)^{\mathcal{G}_{i,j}} - \beta_i \prod_{j=1}^{N} x_j(t)^{\mathcal{H}_{i,j}} \tag{4.4}$$

where $\mathcal{G}_{i,j}$ and $\mathcal{H}_{i,j}$ are kinetic exponents, α_i and β_i are positive rate constants and N is the number of equations in the system. The equations in Eqn. 4.4 can be seen as divided into two components: an excitatory and an inhibitory component. The kinetic exponents $\mathcal{G}_{i,j}$ and $\mathcal{H}_{i,j}$ determine the structure of the regulatory network. In the case $\mathcal{G}_{i,j} > 0$, gene g_j induces the synthesis of gene g_i. If $\mathcal{G}_{i,j} < 0$, gene g_j inhibits the synthesis of gene g_i. Analogously, a positive (negative) value of $\mathcal{H}_{i,j}$ indicates that gene g_j induces (suppresses) the degradation of the mRNA level of gene g_i. Since the modeling of genetic networks involves abstraction and mapping of interactions, the generality of this formula is essential. However, they suffer from the high number of model parameters ($2N^2 + 2N$). And although high performance computing has made significant progress over the last decade, this is still a major issue for larger biological systems.

4.2.8 Arbitrary differential equations

Arbitrary differential equations are the most flexible way to model biological systems. Unfortunately, they are hard to create with algorithms. They can be handled with a special type of Evolutionary Algorithms namely Genetic Programming (see section Genetic Programming).

4.3 Inference Strategies

This section is aimed to give a brief overview over different strategies for inference. The following list shows the algorithms that are implemented in JCell and for which publications are available.

- Standard inference
- Separated inference
- BitSet inference
- Island strategy
- Virtual Knockouts
- Singleton inference
- Skeletalizing

The references and details are given in the subsections for each inference strategy.

Standard Inference

The standard inference strategy. Straightforward modeling of the system together with an exhaustive parameter search. Using this algorithm it is very likely to find a solution to the problem, which does not correspond to the biological system, due to the flexibility of the models.

Tominaga et al. [16] used a GA to infer regulatory systems of very low complexity using an algorithm, which corresponds to a standard inference strategy in our notation.

Separated Inference

The separated approach is the standard algorithm enhanced with a preprocessing phase, in which all components of the system are inferred separately. This is done by modeling the system such that only the component of interest has free parameters to determine, all the other components are "simulated" by using the corresponding experimental data. The resulting parameters for each single component are then used in the actual inference computation as an initial solution. Having a better initialization, it is more likely to find the global optimal solution.

So far, there are no publications using this method.

BitSet Inference

The bitset algorithm is a representative of the class of Memetic Algorithm. This method separates the inference problem into two subproblems. It uses a global algorithm to determine the optimal topology, i.e. the optimal structure of the unknown network, together with a local algorithm, searching for the best parameters for a topology.

This method was published for example in [13].

Island Strategy

The idea of the island strategy is to preserve the diversity of solutions within the optimization population Secondly, it reduces the chance of premature convergence. The general principle of an island strategy is a set of l EA populations, which evolve independently for m generations. Then migration occurs and the best individuals (possible networks) are exchanged between the independent EA populations to "inject" good solutions into each other.

The idea was tested and verified in [14].

Virtual Knockouts

Due to the limited number of available data the inferring problem is under-determined and ambiguous. Further on, the problem often is multi-modal and therefore appropriate optimization strategies become necessary. A large number of different sets of model parameters fit the given data with comparably good fitness values (in respect to the fitness function mentioned above) but with only small resemblance to the true system. Virtual Knockouts are one way to avoid finding solutions with good fitness values but bad parameter distances. In virtual knockout this is successively done by optimizing the parameters of systems of differential equations modeling the interactions in the network for the given data, followed by a second phase, aimed to reduce the ambiguities by suggesting subsequent knockout experiments. Information gained by these follow-up experiments are incorporated into the first phase to increase the probability of finding the correct network model.

Virtual Knockouts have been successfully used in [12].

Singleton Inference

This strategy is aimed to determine the dependencies between a single gene of interest and all other genes within the data set. This question becomes interesting in the case of examination of only a special gene, which is in the focus of the current research project. The algorithm works analogous to the separated strategy and infers a gene with the experimental data of the other genes as input for the mathematical model.

So far, no publications are available.

Skeletalizing

This method is for comparison only. It was inspired by an inference strategy of Tominaga et al. and introduces a special threshold value t_skel, which determines the minimum absolute value of model parameters. Each parameter less then t_skel (absolutely) is forced to 0 and is fixed until the end of the inference process. It is used as a benchmark function to compare algorithms like the bitset inference against it with our bitset method resulting in better networks in less fitness evaluations.

The original implementation of this algorithm can be found in [16].

4.4 Evolutionary Algorithms

Evolutionary Algorithms have proved to be a powerful tool for solving complex optimization problems. Three main types of evolutionary algorithms have evolved during the last 30 years: Genetic Algorithms (GA), mainly developed by J.H. Holland [4], Evolutionary Strategies (ES), developed by I. Rechenberg [9] and H.-P. Schwefel [11] and Genetic Programming (GP) by J.R. Koza [6]. Each of these uses different representations of the data and different operators working on them. They are, however, inspired by the same principles of natural evolution. Evolutionary Algorithms are a member of a family of stochastic search techniques that mimic the natural evolution as proposed by Charles Darwin of mutation and selection. Evolutionary Algorithms are one of the research area of the Computer Architecture group at the University of Tübingen (see Evolutionary Algorithms for more details).

4.5 Genetic Algorithms

Genetic Algorithms (GA) imitate the evolutionary processes with emphasis on genotype based operators (genotype/phenotype dualism). The GA works on a population of artificial chromosomes, referred to as individuals. Each individual is represented by a string of L bits. Each segment of this string corresponds to a variable of the optimizing problem in a binary encoded form. The population is evolved in the optimization process mainly by crossover operations. This operation recombines the bit strings of individuals in the population with a certain probability Pc. Mutation is secondarily in most applications of a GA. It is responsible to ensure that some bits are changed, thus allowing the GA to explore the complete search space even if necessary alleles are temporarily lost due to convergence.

4.6 Evolution Strategy

The second type of Evolutionary Algorithms is the Evolution Strategy (ES). ES differ from GAs mainly in respect to the representation of solutions and the selection operators. They mainly rely on sophisticated mutation operators, smaller population sizes and an increased selection pressure. The selection of the individuals forming a population is deterministic, as in contrast to GAs, where a stochastic method is used. In case of the (μ, λ)-ES selection strategy, the μ best individuals from a population of μ offsprings are selected to create the next population. An alternative implementation is the $(\mu + \lambda)$-strategy, which selects the μ best individuals from the population of the λ offsprings joined with the old population of μ parents.

4.7 Genetic Programming

The idea of Genetic Programming (GP) is to introduce a new data type to evolve and optimize computer programs. In the first experiments, the computer programming language LISP was used to encode problems. For example, the algebraic term $(x - 1) - x^3$ is represented in LISP as:

```
( - ( - x 1 ) (* x ( * x x ) ) )
```

The GP representation for the problems that have to be encoded are trees. These tree structure use a directed acyclic graph with functions as nodes and terminals as leaves. The execution order is given by evaluating the left child before evaluating the right child.

Most commonly, GP programs use the following operators:

- Arithmetic operators: +, -, *, /, sin, cos, exp, ...
- Boolean operators: AND, OR, XOR, NOT, NAND
- Problem specific operators: Max, Min, Variance,...

With GP, Evolutionary Algorithms are able to model dynamic systems using arbitrary differential equations because with the presented encoding, any equation can be created.

4.8 Memetic Algorithm

The combination of an Evolutionary Algorithm with a local search heuristic is called Memetic Algorithm (MA) [8]. MAs are inspired by Dawkin's [2] notion of a meme. A meme is a "cultural gene" and in contrast to genes, memes are usually adapted by the people who transmit them before they are passed to the next generation.

From the optimization point of view, it is argued that the success of an MA is due to the tradeoff between the exploration abilities of the underlying EA and the exploitation abilities of the local searchers used. This means that during variation, the balance between disruption and information preservation is very important: on the one hand the escape of local optima must be guaranteed, but on the other hand disrupting too much may cause the loss of important information gained in the previous generation. Memetic Algorithms combine the features of EA to explore the solution space of combinatorial problems and the ability of exploiting of local search heuristics.

Chapter 5

Menu Descriptions

5.1 The File Menu

5.1.1 Open

Open files in JCell. There are three different types of input files. The first is a data file, containing gene expression data or metabolic concentrations. Second, a topology can be imported to be used in the inference process to incorporate known biological dependencies. And third, a reference model can be provided, which is used as the true system
JCell can read files of the following types:

- Affymetrix: Expression data files exported by Affymetrix Microarray Suite
- ASCII: Tab-delimited ASCII files containing gene expression data
- XML: Expression Data formatted in XML (SGML, JCell)

5.1.2 Export

Data Saves the time course expression data. The user can choose between two following file formats. Both file types can be read in again.

- ASCII: Tab-delimited ASCII files
- XML. Expression Data formatted in XML

Variables Exports the parameter of the model representing the regulatory network. Like for the data, the user has the choice between XML and plain ASCII. The model type and the dependencies are additionally to the parameters exported to the file.

Results This menu item exports the statistics of the inference process, i.e. the computation time, fitness error, key configuration values, and distance to the true system if specified.

5.1.3 Quit

Quits this JCell session.

5.2 The Data Menu

5.2.1 Create Artificial Data

Opens the window for data creation. Here, the user is able to create artificial time course data for a selected model, depending on the specified maximum connectivity, dimensionality, ...
The user has the choice to

- a) create a new model
  To create a model, the user has to provide some parameters, which are used to build a model. This model is then evaluated. for this, it is simulated the number of time steps given by the user and tested for stability, i.e. whether the dynamics of the system are chaotic or not. In case that a new model is to be created, several parameters can be adjusted. First, the type of the mathematical model has to be specified. Currently, there are five models implemented. Secondly, the dimensionality (number of genes in the system) and the cardinality (maximum number of connections) can be chosen together with the maximum number of different runs, i.e. how many systems are evaluated. Further on, the user has to enter the number of time steps to simulate and the time step size, i.e. the time difference between two sampling points.

- b) import a reference model
  In the case of importing an existing model, the user can specify a file by browsing the file system. The knockout-checkbox determines whether to create a time series with corresponding knock-out time series. This forces JCell to simulate the model knockout each gene and therefore creating $(n+1)$ different data files.

5.3 The Topology Menu

5.3.1 Edit Graph

Currently, only a rudimentary graph editor is available to model gene dependencies. However, it is powerful enough to capture all relevant biological interactions between

genes. With the editor, the user can model the influences on the synthesis and on the degradation of a gene, symbolized by the two lines linked to the circles representing a gene. For each edge between genes, the user can specify the type of the interaction, for example activating, inhibiting or unknown. In the last case, the algorithm tries to determine the relationship.

5.3.2 Map Expression

Data With this function, gene expression data or metabolic concentrations can be mapped onto pathways that have been either created with the editor or imported from databases. While mapping the genes, JCell tries to find corresponding genes in the data set together with its counterpart in the pathway. If the mapping is successful, the data set is assigned to the pathway and the inference process can rely on a known topology.

5.3.3 Query Database

To incorporate known biological information into the inference process, the user can import pathways from databases. There a few databases worldwide, which provide information about pathways and interactions among system components like gene-gene interactions.

KEGG

For the KEGG database we have implemented an interface, which enables the user to browse the data in KEGG and to select a specific pathway. The import shows the list of all available pathways for the selected organism (homo sapiens). There is a toolbar on the right of the pathway-window to show the graph of the selected pathway and to list all genes participating in this system. After importing a pathway, the graph is shown in the JCell GUI, with all possible inputs highlighted. With importing a pathway, the mapping function becomes available. If a data set is mapped onto the imported pathway, only the successfully mapped genes are highlighted.

5.4 The Analysis Menu

5.4.1 Inference

This is the main method of JCell. This menu items starts the inference process. It will become available if at least one data file was loaded or a data set was created. The inference process depends on the data files, the Configuration, which can be loaded or entered via the Inference Settings (see Inference Settings). Additionally, a Topology – describing the logical dependencies between genes – or a Reference

Model, which represents the true model, i.e. having the correct parameters, can be loaded to be incorporated in the inference computation.

The Topology is used for building a structure of the regulatory system. Dependencies between genes can be modeled either as activating, inhibiting or unknown. In the later case, the algorithm tries to infer the relationship. By using a Topology, researchers can input known information about the system that they want to infer.

The Reference Model is just for verification purposes. It is not used in the inference computation, but after the process has terminated to calculate statistical values, for example quality of the solution, positive and negative trues and false, respectively, and so on.

5.4.2 Exploration

After the inference process has terminated, this method enables the user to explore the fitness landscape of the solution space. The user has to specify two dimensions of the solution space to be displayed in a three-dimensional graph. For this method to work, MatLab has to be installed. The exploration method performs a grid search on all possible combinations of the two selected dimensions and displays them together with the achieved fitness value. This is interesting to learn more about the properties of the underlying problem.

5.5 The Preferences Menu

The options of JCell are divided between the general settings of the application and the options to be used in the inference process.

5.5.1 Program Settings

General options

The first part of the program settings is shown in the figure below. The user can specify the maximum number of CPUs to use for the inference computation. Further on, this option window lets the user enter the details about the network connection like the proxy settings or whether the inference is run on the local system or on a remote server.

GUI options

In the second part, so far only the font size of the GUI can be adjusted. The user can choose between font size 24 or 32.

5.5.2 Inference Settings

Inference Strategy The inference settings are more important. Here, the user specifies the type of algorithm to use in the inference process. Currently, seven different strategies are implemented in JCell:

- Standard inference,
- Separated inference,
- BitSet inference,
- Virtual Knockouts,
- Singleton inference,
- Skeletalizing inference

A detailed description of each strategy can be found in section Inference Strategies.

Processor Settings The next option panel shows the configuration for the parameter determination algorithms. There are several methods to solve the inference process. The first type are *Evolutionary Algorithms*, which are stochastic optimization techniques that mimic the natural evolution process of repeated mutation and selection as proposed by Charles Darwin. They are inspired by the same principles of natural evolution and have proved to be a powerful tool for solving complex optimization problems and in particular combinatorial problems. For details see section Evolutionary Algorithms. The second type are *direct heuristics*, which can be selected if they are appropriate for the current model.

Problem Settings The next configuration window lists global and local problem settings. Here, one can set the maximum number of fitness evaluations, the number of multi-runs for statistical purposes, the fitness and penalty functions to use, the number of hidden metabolites in the system, and the boundaries for the parameter values of the model, if they are known. The local options are only available, if an inference strategy is selected, which provides a local search... Additional or hidden metabolites are useful, if there are additional components in the system, which play intermediate roles or cannot be measured and have therefore be treated separately. The boundaries of the parameter values represents the range, which is used by the inference processor, for example an Evolutionary Algorithm, to limit the possible values of the optimization variables. Otherwise, the combinatorial problem would take infinitive time or is not likely to find an optimum.

Model Settings The user can specify the type of mathematical model for the inference process in this configuration window. The figure below shows the settings for the global and the local search. Local search options are only available, if the inference strategy allows for a local search.
Currently, there are six mathematical models to simulate regulatory networks implemented:

- Random boolean networks (RBN),
- Weight matrices,
- pseudolinear weight matrices,
- S-Systems,
- H-Systems, and
- arbitrary differential equations.

Detailed descriptions can be found in section Mathematical Models.

Solver Settings For simulation, the model has to be integrated. There are two solver implemented in JCell:

- Simple Euler-Cauchy solving
- Runge-Kutta fourth order

For both solver types, integration step size can be adjusted. Additionally, an enhancement for the Runge-Kutta can be selected, which regulates the step size adaptively.

Get Token

This method is only useful for researchers working on remote computers, which require to obtain tokens. The Get Token command obtains a token and updates it regularly.

5.6 The Help Menu

5.6.1 About...

The About menu item shows the credits of the applications.

Chapter 6

Command Line Options

6.0.2 General Syntax

The program JCell has a command line option to run with or without scripts. The general syntax is:

```
java -jar jcell.jar {-option parameter} {-...}
```

The following command starts JCell in interactive mode, displaying the GUI on the current display:

```
java -jar jcell.jar
```

6.0.3 Command Line Options and Parameter

-HELP

This option displays the syntax and to states some comments on the usage of JCell.

-NOGUI

Using the [-NOGUI] option, JCell will be started without displaying the GUI.

-KEEPALIVE

This option forces the application to ask for login information to keep a AFS-token alive.

-CPU n

This option specifies the maximum number of cpus to be used.

-EMAIL

This option forces JCell to send an email containing the most important information after finishing the inference process.

-EXPORT

The [-EXPORT] option specifies whether JCell outputs the results to a file.

-OUT file

This file name will be used, if JCell outputs results.

-CONFIG file

With this option, a user can specify a file containing configuration details for the inference process.

-TOPOLOGY file

This option can be used to include topological information into the inference. This makes it possible to include known biological information.

-REFERENCE file

This option specifies a file with information about the "true" system. This feature can be used to evaluate the efficiency of the implemented algorithms.

-DATA files

files is a list of data files, which are to be incorporated in the computation. This could be, for example, different knockout data files.

-FORMAT format

The [-FORMAT] option determines the format of the output files. At this moment the following formats are supported: plain ASCII text and XML.

Acknowledgments

This work was supported by the National Genome Research Network (NGFN) of the Federal Ministry of Education and Research in Germany under contract number 0313323.

Bibliography

[1] T. Chen, H. L. He, and G. M. Church. Modeling gene expression with differential equations. In *Proceedings of the Pacific Symposium on Biocomputing*, 1999.

[2] R. Dawkins. *The selfish Gene*. Oxford University Press, 1976.

[3] P. D'haeseleer, X. Wen, S. Fuhrman, and R. Somogyi. Linear modeling of mRNA expression levels during CNS development and injury. In *Proceedings of the Pacific Symposium on Biocomputing*, volume 4, pages 41–52, 1999.

[4] J. H. Holland. *Adaption in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Systems*. The University Press of Michigan, Ann Arbor, 1975.

[5] S. A. Kauffman. *The Origins of Order*. Oxford University Press, New York, 1993.

[6] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[7] B. Lewin. *Genes VII*. Oxford University Press, 2000.

[8] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Toward memetic algorithms. Technical Report C3P Report 826, California Institute of Technology, 1989.

[9] I. Rechenberg. *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, Stuttgart, 1973.

[10] M. A. Savageau. 20 years of S-systems. In E. Voit, editor, *Canonical Nonlinear Modeling. S-systems Approach to Understand Complexity*, pages 1–44, New York, 1991. Van Nostrand Reinhold.

[11] H.-P. Schwefel. *Numerical optimization of computer models*. John Wiley and Sons Ltd, 1981.

[12] C. Spieth, F. Streichert, N. Speer, and A. Zell. Iteratively inferring gene regulatory networks with virtual knockout experiments. In *Proceedings of the 2nd European Workshop on Evolutionary Bioinformatics (EvoWorkshops 2004)*, volume 3005 of *LNCS*, pages 102–111, 2004.

[13] C. Spieth, F. Streichert, N. Speer, and A. Zell. Optimizing topology and parameters of gene regulatory network models from time-series experiments. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2004)*, volume 3102 (Part I) of *LNCS*, pages 461–470, 2004.

[14] C. Spieth, F. Streichert, N. Speer, and A. Zell. Utilizing an island model for ea to preserve solution diversity for inferring gene regulatory networks. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2004)*, pages 146–151, 2004.

[15] D. Thieffry and R. Thomas. Qualitative analysis of gene networks. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 77–87, 1998.

[16] D. Tominaga, N. Kog, and M. Okamoto. Efficient numeral optimization technique based on genetic algorithm for inverse problem. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 00)*, pages 251–258, 2000.

[17] D. Weaver, C. Workman, and G. Stormo. Modeling regulatory networks with weight matrices. In *Proceedings of the Pacific Symposium on Biocomputing*, volume 4, pages 112–123, 1999.

[18] A. Wuensche. Genomic regulation modeled as a network with basins of attraction. In *Proceedings of the Pacific Symposium on Biocomputing*, volume 3, pages 89–102, 1998.