Information Visualization Techniques for Metabolic Engineering

Dissertation

zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften (Dr. rer. nat)

dem
Fachbereich Mathematik und Informatik der
Philipps-Universität Marburg
vorgelegt von

Ermir Qeli

aus Berat, Albanien

Marburg/Lahn 2007

Vom Fachbereich Mathematik und Informatik der
Philipps-Universität Marburg als Dissertation am
29. Januar 2007 angenommen.

Erstgutachter: Prof. Dr. Bernd Freisleben, Philipps-Universität Marburg

Zweitgutachter: Prof. Dr. Wolfgang Wiechert, Universität Siegen

Tag der mündlichen Prüfung: 5. Februar 2007

Acknowledgements

The accomplishment of this dissertation would not have been possible without the contribution of many individuals, to whom I want to express my appreciation and gratitude.

First of all, I want to gratefully acknowledge my supervisor, Prof. Dr. Bernd Freisleben, for his persistent support and encouragement. His academic guidance and thorough insights helped me to find my own way in the world of research.

I wish to express my gratitude to Prof. Dr. Wolfgang Wiechert for the fruitful technical discussions we had and for the insights he provided to me in the field of metabolic engineering and systems biology.

My acknowledgement also goes to Prof. Dr. Bernhard Seeger and Prof. Dr. Eyke Hüllermeier for serving in my dissertation committee.

The research work presented in this thesis was financially supported by the Deutsche Forschungsgemeinschaft (SPP 1063, Teilprojekt FR 791/8-1) and by Deutscher Akademischer Austauschdienst (DAAD, Stability Pact for Southeastern Europe). In this context, I would like to thank Prof. Dr. Mira Mezini and Dr. Jochen W. Münch for their support.

For the fruitful cooperation between the University of Marburg, the University of Siegen and the Research Center Jülich, I want to thank my colleagues Aljoscha Wahl and Stephan Noack for the discussions we had and their ideas and comments with respect to the visualization tools MetVis and MatVis. Marc Haunschild is acknowledged for his work on building the simulation tool MMT2. Dr. Ralf Takors and his successor, Dr. Marco Oldiges, are acknowledged for their support in the broader context of the DFG project.

I want to thank all the people at the Department of Mathematics and Computer Science of the University of Marburg for their support during my time here. In particular, I want to mention Dr. Axel Schröder and Alexander Markowetz. I want to thank Mrs. Mechthild Keßler for being there when I

needed her.

Special thanks go to my family, my parents Ismail and Feride, my sister Nilda and my brother Albi, for their extensive support. They provided me with the necessary backing and kept me in high spirits when it was needed.

Last but not least I want to thank Julinda Gllavata for her precious help and wide-ranging support. She had the patience to listen to my viewpoints, discuss issues with me and convince me when I was on the wrong track.

Abstract

The main purpose of metabolic engineering is the modification of biological systems towards specific goals using genetic manipulations. For this purpose, models are built that describe the stationary and dynamic behaviour of biochemical reaction networks inside a biological cell. Based on these models, simulations are carried out with the intention to understand the cell's behaviour. The modeling process leads to the generation of large amounts of data, both during the modeling itself and after the simulation of the created models. The manual interpretation is almost impossible; consequently, appropriate techniques for supporting the analysis and visualization of these data are needed.

The purpose of this thesis is to investigate visualization and data mining techniques to support the metabolic modeling process. The work presented in this thesis is divided into several tracks:

- *Visualization of metabolic networks and the associated simulation data.*

  Novel visualization techniques will be presented, which allow the visual exploration of metabolic network dynamics, beyond static snapshots of the simulated data plots. Node-link representations of the metabolic network are animated using the time series of metabolite concentrations and reaction rates. In this way, bottlenecks and active parts of metabolic networks can be distinguished. Additionally, 3D visualization techniques for metabolic networks are explored for cross-free drawing of the networks in 3D visualization space. Steerable drawing of metabolic networks is also investigated. In contrast to other approaches for drawing metabolic networks, user guided drawing of the networks allows the creation of high quality drawings by including user feedback in the drawing process.

- *Comparison of XML/SBML files.*

  SBML (Systems Biology Markup Language) has become ubiquitous in metabolic modeling, serving the storage and exchange of models in XML format. Generally, the modeling process is an iterative task where the next generation model is a further development of the current model, resulting in a family of models stored in SBML format. The SBML format, however, includes a great deal of information, from the structure of the biochemical network to parameters of the model or measured data. Consequently, the CustX-Diff algorithm for a customizable comparison of XML files will be introduced. By customizing the comparison process through the specification of XPath expressions, an adaptable change detection process is enabled. Thus, the comparison process can be focused on specific parts of a XML/SBML document, e.g. on the structure of a metabolic network.

- *Visual exploration of time-varying sensitivity matrices.*

  Sensitivity analysis is a special method used in simulation to analyze the sensitivity of a model with respect to its parameters. The results of sensitivity analysis of a metabolic network are large time-varying matrices, which need to be properly visualized. However, the visualization of time-varying high-dimensional data is a challenging problem. For this purpose, an extensible framework is proposed, consisting of existing and novel visualization methods, which allow the visual exploration of time-varying sensitivity matrices. Tabular visualization techniques, such as the reorderable matrix, are developed further, and algorithms for their reordering are discussed. Existing and novel techniques for exploring proximity data, both in matrix form and projected using multi-dimensional scaling (MDS), are also discussed. Information visualization paradigms such as focus+context based distortion and overview+details are proposed to enhance such techniques.

- *Cluster ensembles for analyzing time-varying sensitivity matrices.*

  A novel relationship-based cluster ensemble, which relies on the accumulation of the evolving pairwise similarities of objects (i.e. parameters) will be proposed, as a robust and efficient method for clustering time-varying high-dimensional data. The time-dependent similarities, obtained from the fuzzy partitions created during the fuzzy clustering process, are aggregated, and the final clustering result is derived from this aggregation.

Zusammenfassung

Metabolic Engineering beabsichtigt die zielgerichtete Modifikation biologischer Systeme mittels genetischer Manipulationen. In diesem Kontext werden Modelle gebaut, die sowohl das stationäre, als auch das dynamische Verhalten der biochemischen Reaktionen innerhalb einer biologischen Zelle beschreiben. Auf dieser Basis werden Simulationen durchgeführt, mit dem Ziel, das Verhalten einer Zelle zu verstehen. Der Modellierungsprozess ist mit der Erzeugung großer Datenmengen, sowohl während der Modellierung, als auch nach der Simulation der erstellten Modelle, verknüpft. Da eine manuelle Auswertung nahezu unmöglich ist, werden Techniken zur Analyse und Visualisierung solcher Daten benötigt.

Die vorliegende Dissertation behandelt geeignete Ansätze der Visualisierung und des Data Minings zur Unterstützung des metabolischen Modellierungsprozesses. Die Arbeit liefert Beiträge zu folgenden Problemstellungen:

- *Visualisierung metabolischer Netzwerke sowie der zugehörigen Simulationsdaten.*

  Neuartige Visualisierungstechniken, die die visuelle Exploration des Simulationsverlaufs metabolischer Netzwerke jenseits von statischen Schnappschüssen erlauben, werden vorgestellt. Knoten-Kanten Darstellungen von metabolischen Netzwerken werden durch Benutzung der jeweiligen Zeitreihen der Metabolitenkonzentrationen und Reaktionsraten animiert. Engpässe und aktive Teile eines Netzwerks werden leichter identifizierbar. 3D Visualisierungsmethoden, die das Zeichnen von Netzwerken ohne Überschneidungen von Kanten ermöglichen, werden untersucht. Das Konzept des gesteuerten Zeichnens metabolischer Netzwerke wird eingeführt. Im Gegensatz zu anderen Ansätzen wird ein vom Benutzer steuerbarer Ansatz zum Zeichnen metabolischer Net-

zwerke laut biochemischer Anforderungen mittels Einbindung von Benutzerrückmeldungen vorgestellt.

- *Vergleich von XML/SBML Dateien.*

  SBML (Systems Biology Markup Language), ein XML-Format zur Speicherung und zum Austausch von biochemischen Modellen, ist allgegenwärtig in der metabolischen Modellierung geworden. Die metabolische Modellierung funktioniert üblicherweise als ein iterativer Prozess, bei dem neue Modelle aus vorhergehenden Modellen abgeleitet werden. Somit entstehen Familien von Modellen, die im SBML-Format gespeichert werden. Die abgespeicherten Daten enthalten eine große Menge an Informationen, von der Struktur eines Netzwerkes bis zu den Parametern eines Modells oder entsprechenden Messdaten. Zur Reduktion der Datenmenge wird der CustX-Diff Algorithmus zum gesteuerten Vergleich von XML-Dokumenten eingeführt. Durch Spezifikation von XPath-Ausdrücken wird ein anpassbarer Differenzprozess ermöglicht, der eine Fokussierung des Vergleichs auf bestimmte Teile von XML/SBML-Dokumenten erlaubt.

- *Visuelle Exploration von zeitabhängigen Sensitivitätsmatrizen.*

  Sensitivitätsanalyse wird desöfteren in der Modellierung genutzt, um die Sensitivität eines Modells in Bezug auf seine Parameter zu untersuchen. Die Ergebnisse der Sensitivitätsanalyse eines metabolischen Modells sind große zeitabhängige Matrizen, die analysiert bzw. visualisiert werden sollen. Die Visualisierung hochdimensionaler Datensätze ist ein anspruchsvolles Problem. Daher wird ein erweiterbares System, bestehend aus vorhandenen und neuartigen Visualisierungstechniken zur visuellen Exploration von zeitabhängigen Sensitivitätsmatrizen, vorgestellt. Tabellarische Visualisierungsansätze wie die sogenannte *Reorderable Matrix* werden weiterentwickelt, und Algorithmen für deren Sortierung werden diskutiert. Techniken zur Visualisierung von Proximitätsdaten, sowohl in Matrizenform, als auch projeziert anhand der Methode der Multidimensionalen Skalierung (MDS) werden erforscht. Informationsvisualisierungsparadigmen wie die *focus+context* basierte Verzerrung, sowie *overview+details* Methoden werden eingesetzt, um die vorgeschlagenen Visualisierungstechniken zu verbessern.

- *Cluster-Ensembles zur Analyse von zeitabhängigen Sensitivitätsmatrizen.* Neuartige verwandtschaft-basierte Cluster-Ensembles, die sich auf die Akkumulation der enstehenden paarweisen Ähnlichkeiten zwischen Objekten stützen, werden zur Clusterung von zeitabhängigen hochdimensionalen Datensätzen vorgeschlagen. Die zeitabhängigen Ähnlichkeiten,

die aus den unscharfen Partitionen des Fuzzy-Clustering Prozesses entstehen, werden aggregiert, und die daraus resultierende Proximitätsmatrix wird zur Erzeugung des endgültigen Clusterergebnisses verwendet.

Contents

List of Figures

List of Tables

List of Algorithms

Table of Abbreviations

Biochemical Substances

Abbreviation	Meaning
6PG	6-Phosphogluconate
AcCoA	Acetyl-Coenzyme A
ADP	Adenosindiphosphate
AMP	Adenosinmonophosphate
ATP	Adenosintriphosphate
BM	Biomass
DAHP	3-Deoxy-D- Arabino-Heptulosonate-7-Phosphate
DHAP	Dihydroxyacetonphosphate (Glyceronphosphate)
DHQ	3-Dehydroquinate
DHS	3-Dehydroshikimate
E4P	Erythrose-4-phosphate
EPSP	5-Enolpyruvyl Shikimate-3-Phosphate
F6P	Fructose-6-phospate
FBP	Fructose- 1,6-bisphosphate
G6P	Glucose-6-phosphate
GAP	Glyceraldehyd-3-phosphate
LEU	L-Leucine
L-Phe	L-Phenylalanine
NAD/ NADH	Nicotinamide-Adenine-Dinucleotide
NADP/ NADPH	Nicotinamide-Adenine-Dinucleotide-Phosphate

Abbreviation	Meaning
PEP	Phosphoenolpyruvate
PPP	Pentose-Phosphate-Pathway
Pyr	Pyruvate
R5P/ RIBU5P	Ribose-5-Phosphate
S3P	Shikimate-3- Phosphate
S7P	Sedo-heptulose-7-Phosphate
SHIK	Shikimate
SRE	Stimulus-Response-Experiment
TCA	Citric Acid Cycle
VAL	L-Valine
X5P/ XYL5P	Xylulose-5-Phosphate

Biological Terms

Abbreviation	Meaning
C. glutamicum	Corynebacterium Glutamicum
E. coli	Escherischa Coli
In-Silico	Simulated experiments
In-Vitro	Experiments under artificial conditions in a test tube
In-Vivo	Experiments with living cell/system
LC-MS	Liquid Chromatography Mass Spectrometry
S. cerevisiae	Saccharomyces Cerevisiae

Computer Science Terms

Abbreviation	Meaning
CSV	Comma Separated Values
GUI	Graphical User Interface
JNI	Java Native Interface
ODE/ PDE	Ordinary/ Partial Differential Equation
M3L	Metabolic Modeling Markup Language
MatVis	Matrix Visualizer
MetVis	Metabolic Visualizer
MM	Michaelis-Menten Equation
MMT	Metabolic Modeling Tool
RMI	Remote Method Invocation
SBML	Systems Biology Markup Language
SSQ	Sum of Squares
UML	Unified Modeling Language
XML	Extensible Markup Language

1
Introduction

1.1 Metabolic Engineering

Metabolism has been a research subject since ancient times. The old approach to understand the functioning of the metabolism was *trial-and-error*, and most of the current knowledge in this field is based on centuries of experience with such an approach. However, the trial-and-error approach does not provide an insight into the underlying functionality of the metabolism, thus resulting in limitations regarding new treatments or gaining new knowledge. This limitation shifted the attention of scientists to the idea of searching for laws and rules that govern the metabolism. The first approaches in modeling metabolism fall into the *reductionism* category. Reductionism is a top-down approach which proceeds in the following way: to understand an organism, one should understand its parts, and to understand its parts one would have to understand the cells and their components until one achieves the elementary components already understood. However, this approach is insufficient since there is a strong difference between the organism and its chemical components inside a test tube; in the same way there is a big difference between a cake and its raw components. From reductionism there was then a paradigm shift to *reconstructionism*, which apart from studying components of the system separately, integrates the isolated parts together in order to analyze their interactions with the environment. In the context of integrated biochemical systems, models of *biochemical i.e. metabolic networks* play an important role. Models are used to address one of the central questions in metabolic networks, namely to identify which parts of a complex network are important and how they interact with each other. By building a quantitative model

which describes the biochemical processes occurring inside the cell, it is possible to deduce the optimal conditions for maximizing the productivity of the cell.

An important concept in this context is *Metabolic Engineering*, which is defined as the *"guided improvement of product formation or cellular properties through the modification of specific biochemical reactions or introduction of new ones by using recombinant DNA technology"* [SS93]. Thus, using specific genetic manipulations, it is possible to modify the metabolism of a certain organism, mainly a microorganism, for the purpose of its optimization. For achieving this purpose, a thorough study of different metabolic processes inside the cell is needed. One possibility to study this problem is by using mathematical models, which describe the timely evolution of cell dynamics. Mathematical modeling of the cell works as an iterative process, where the next generation model builds on the previous one. This process is accompanied with generation and interpretation of large quantities of data. Their interpretation is strongly related to visualization, and because of the abstract nature of the data it is related to *Information Visualization* (InfoVis) [CMS99].

The next section briefly introduces the field of information visualization, whereas the theoretical background in metabolic engineering used in this thesis will be described in Chapter 2.

1.2 Information Visualization

Information is essential to make good decisions. In our everyday life, it comes in a variety of forms; from a mere table of numbers or a collection of texts to well-structured drawings of underground connections.

If we look for the term *information* in the Merriam-Webster Online Dictionary (`www.m-w.com/`) we will find the following definitions:

1. the communication or reception of knowledge or intelligence

2. **a** (1) : knowledge obtained from investigation, study, or instruction (2) : INTELLIGENCE, NEWS (3) : FACTS, DATA **b** : the attribute inherent in and communicated by one of two or more alternative sequences or arrangements of something (as nucleotides in DNA or binary digits in a computer program) that produce specific effects **c** (1) : a signal or character (as in a communication system or computer) representing data (2) : something (as a message, experimental data, or a picture) which justifies change in a construct (as a plan or theory) that represents physical or mental experience or another construct **d** : a quantitative measure of the content of information; specifically : a

numerical quantity that measures the uncertainty in the outcome of an experiment to be performed.

From the computer science point of view, there are two aspects of information that occur in the literature: the concept of information in information theory (item 2.2.d of the definition above) which deals with the uncertainty factor contained in a certain sequence and the concept of information related to knowledge (item 2.1 of the definition). The latter is interesting for us in the context of *information visualization*. But what is information visualization?

Card et al. [CMS99] define *visualization* as **the use of computer supported, interactive, visual representations of data to amplify cognition**. This definition contains the essence of visualization, which is its purpose to amplify the understanding of the underlying object of study. In analogy to computation, whose purpose is insight and not numbers [Ham87], the purpose of visualization is insight and not pictures [CMS99].

The definition of information visualization is derived from the definition of visualization as *the use of computer supported, interactive, visual representations of* ***abstract*** *data to amplify cognition*. Thus, the abstract nature of data is what makes information visualization different from related fields such as scientific visualization where data often has a clear visual representation. The idea can be illustrated easily with two small examples. The first one comes from the application field of this thesis, namely visualization of multidimensional data in the form of matrices.

The data in Figure 1.1(a) present a matrix with real values. Suppose we want to search for similar columns/rows or just for nearly equal values. We would have to traverse the values several times and possibly the results of our search would not be so encouraging. If we convert the values into colors like in Figure 1.1(b), the understanding of the data is easier. The data is abstract in the context of information visualization because it does not possess a clear mental model to be used as a visual representative.

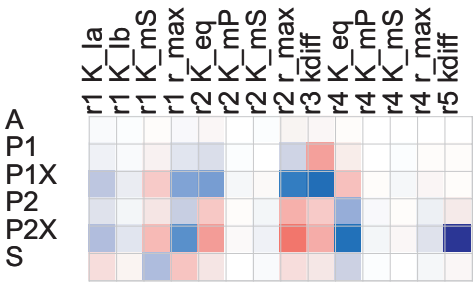
The second example in Figure 1.2 shows an example from graph drawing, where the network of protein-protein interactions is visualized [JMBO01]. The nodes represent proteins and the edges possible interaction between them. The various colors used in this visualization signify the effect of removing the corresponding protein.

1.3 Motivation

Modeling and simulation play an important role in metabolic engineering. Different models of metabolic pathways are built to describe the metabolism of a cell. Simulators operating on these models generate results which are then compared to *in vitro* experimental data and to *in vivo* data if they

	r1 K_Ia	r1 K_Ib	r1 K_mS	r1 r_max	r2 K_eq	r2 K_mP	r2 K_mS	r2 r_max	r3 kdiff	r4 K_eq	r4 K_mP	r4 K_mS	r4 r_max	r5 kdiff
A	0.007	0.002	-0.01	0.013	-0.032	-0.001	0.002	-0.043	-0.031	-0.016	-0.001	0.001	-0.002	-0.003
P1	0.031	0.009	-0.046	0.059	0.072	0.003	-0.005	0.101	-0.384	-0.066	-0.003	0.004	-0.01	-0.012
P1X	0.156	0.045	-0.237	0.299	0.318	0.014	-0.024	0.432	0.479	-0.268	-0.012	0.016	-0.037	-0.017
P2	0.064	0.02	-0.097	0.124	-0.241	-0.011	0.018	-0.329	-0.231	0.257	0.012	-0.016	0.036	-0.074
P2X	0.193	0.055	-0.294	0.371	-0.397	-0.018	0.03	-0.535	-0.343	0.472	0.02	-0.028	0.063	0.979
S	-0.132	-0.039	0.2	-0.253	-0.098	-0.004	0.007	-0.133	-0.091	0.108	0.005	-0.007	0.015	-0.03

(a) The raw matrix data



(b) Its colored visualization

Figure 1.1: An example of visualization of multidimensional data using the reorderable matrix. The red-white-blue spectrum is used to convert numerical values into colors.

exist, in order to select the model which describes these data best. In this context, *in vivo* data is data obtained from a living organism, *in vitro* data represents data obtained from experiments in a test tube, whereas *in silico* data means data generated in simulations. However, the selection of the right model describing a certain metabolic network does not only require reliable simulation results but also their correct interpretation.

Figure 1.3 represents the steps followed in metabolic modeling in the broader context of our project. The experimental data, which in our case is represented by rapid sampling experimental data, associated with the biological know-how is used to set up an initial model. After the initial setup, a simulator tailored to the model is built by code generation. Based upon simulation results, the model is then improved further. For this purpose, sensitivity analysis, a technique used to separate the important parameters of the model from the less significant ones is employed. Typically, the latter are discarded to arrive at a simpler model. The process is iterated to obtain the best model with respect to the reproduction of measured data and model complexity.

Visualization is very helpful in the process of modeling and simulation and is used extensively not only to support the modeling process but also to explore the data generated from the simulations.

In the following, the problems that we discuss in this dissertation will be

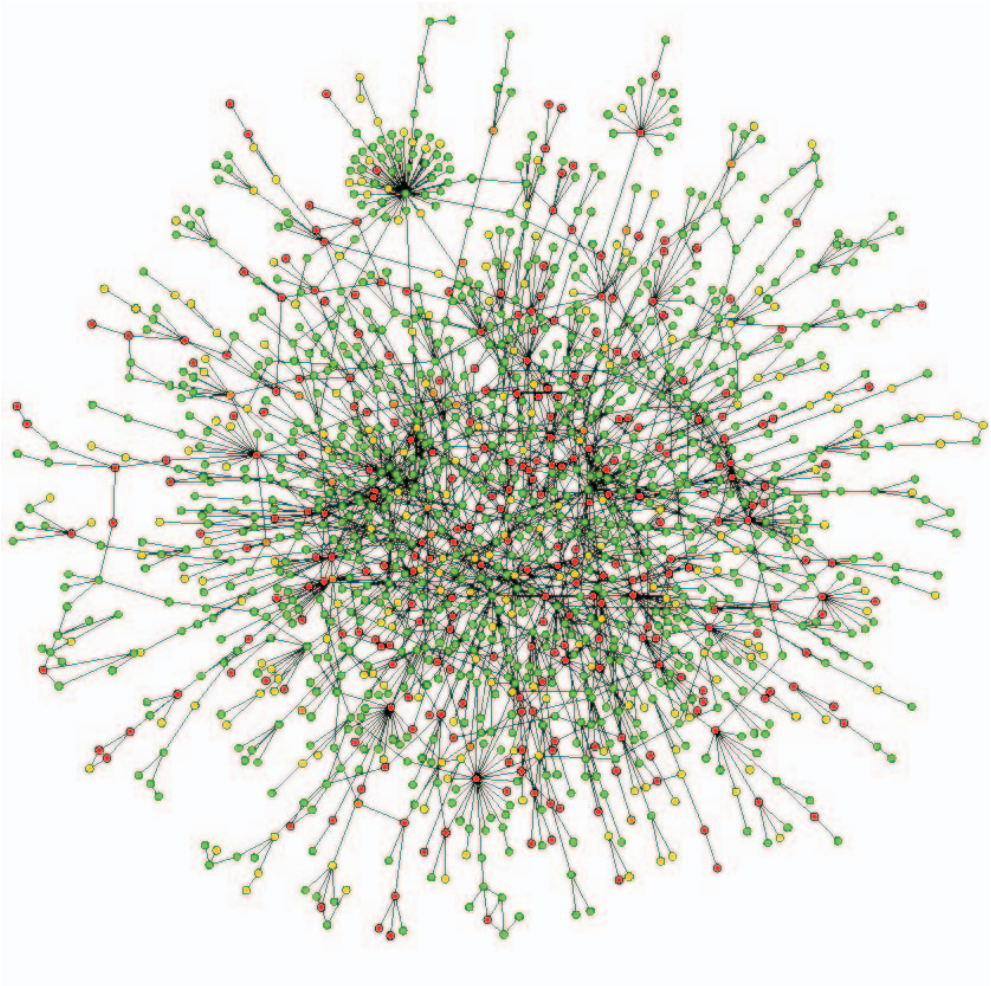


Figure 1.2: Drawing of protein-protein interaction networks (taken from [JMBO01]). The nodes representing proteins, are colored according to the effect of their removal with red being lethal; green, non-lethal; orange, slow-growth and yellow unknown.

described.

1.3.1 Visualization of Metabolic Networks

Metabolic network models usually encode a graph structure and parameters. The latter are used to generate simulations of the metabolic network. The results of simulation include time-series data, describing the timely evolution of the network dynamics. A challenging problem in this context is the integrated visualization of these time-series data with the graph representation of the metabolic network. Furthermore, the automatic drawing of metabolic pathways according to established biochemical conventions pos-

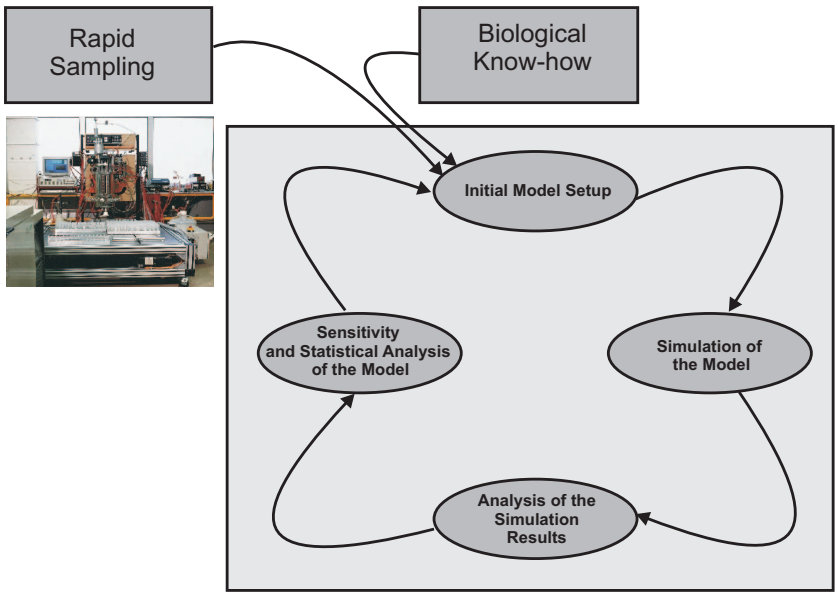


Figure 1.3: The steps involved in the modeling process. Biological know-how coupled with experimental data is included in the iterative process of metabolic modeling.

sesses a greater scale of difficulty compared to the traditional graph drawing problem. Consequently, efficient solutions to these problems are proposed in Chapter 4.

1.3.2 Visual Exploration of Time-Varying Sensitivity Matrices

Sensitivity matrices represent simulation-generated data, which describe how strong the output of the metabolic model simulation is affected by changes in its parameters. The visualization of time-varying sensitivity matrices is helpful to find and reduce redundancies in the parameters of a model. Several visualization techniques, including dimension reduction techniques such as multidimensional scaling [YH38, Mar79] and the Sammon mapping [Sam69], and tabular visualization techniques based on the reorderable matrix concept [Ber81] have been extended to allow the visualization of time-varying sensitivity matrices (see Chapter 6).

1.3.3 Clustering of Time-Varying Data

The aim of clustering algorithms is to partition a data set into groups of similar objects. Existing cluster algorithms deal with static data or low di-

mensional streaming data. The problem of clustering time-varying data, i.e. evolving sensitivity matrices, is the focus of Chapter 7.

1.3.4 Comparison of Semistructured Files

Metabolic models are typically saved as XML files containing all the above mentioned information such as structure, parameters, experimental data and so on. An important problem is change detection on these XML files. Since the techniques for comparing XML files are generic in nature and do not take into consideration the semantics of the format, a customizable change detection algorithm is proposed in Chapter 5.

1.4 Project Framework

The techniques introduced in this thesis were developed in the framework of a large project in the area of metabolic engineering and systems biology. This project was financed by the DFG (Deutsche Forschungsgemeinschaft) under the name *Modeling of metabolic networks based on in-vivo-, in-vitro-, and in-silico data*, as part of the DFG Priority Programme SPP 1063: *Informatics methods for the analysis and interpretation of large genomics data sets*. The work was performed in the Distributed Systems Group at the University of Marburg, Germany in cooperation with two other working groups, the IBT (Institute for Biotechnology), Research Center Jülich, Germany and the Institute of Systems Engineering, University of Siegen, Germany.

1.5 Contributions of this Thesis

The goal of this dissertation is to improve the understanding and interpretation of data generated during the metabolic modeling process. This data is abstract in nature and ranges from graphs representing the structure of metabolic networks to complex high-dimensional time-varying matrices describing the sensitivity of the model with respect to its parameters. The ideas utilized in this dissertation are derived from machine learning, clustering, classification, graph theory, graph drawing, computer graphics and vision and human-computer interaction. The contributions of this dissertation are as follows:

- Design and implementation of MetVis, a software tool that allows visualization and animation of metabolic networks with associated time series representing the timely evolution of the simulations.

- Development of an interactive graph drawing algorithm tailored to metabolic networks, which includes user feedback to enhance the drawing process.
- Design and implementation of a prototypical 3D visualization tool, for interactive exploration of metabolic networks in three dimensions with reduced crossing of edges.
- Development of novel algorithms for customizable change detection and similarity measurement for XML files representing metabolic models.
- Design and implementation of MatVis, a tool for interactive visualization of time-varying matrices. Such matrices are high-dimensional, with 10-100 attributes and dynamically evolving for 50-1000 time points. This tool supports:
  - Reorderable matrices [Ber81], which provide means for visualizing high-dimensional data in a tabular form.
  - Dimension reductioning techniques such as Multidimensional Scaling [Mar79] and Sammon Mapping [Sam69].
  - Interactive visualization of clustering results.
  - Visualization of cluster evolution for the evolving objects.
- Development of a novel fuzzy clustering ensemble algorithm for clustering time-varying data.

Although the evaluation of techniques presented in this dissertation is focused on metabolic modeling data, they are useful in a broader context. For example, the idea of customizable change detection algorithms to compare metabolic models saved as XML files can also be applied to compare other types of XML formats. In the same way, the fuzzy clustering ensemble proposed in Chapter 7 is also applicable to any time-varying multidimensional data.

In the framework of the research performed in this thesis, the following papers have been published.

1. E. Qeli, B. Freisleben, D. Degenring, A. Wahl and W. Wiechert. "MetVis: A Tool for Designing and Animating Metabolic Networks", *in Proceedings of European Simulation and Modelling Conference*, Naples, Italy, pp. 333-338, EUROSIS Press, 2003
2. E. Qeli, W. Wiechert and B. Freisleben. "Visualization of Sensitivity Matrices Generated During Simulations of Metabolic Network Models", *in Proceedings of the IASTED International Conference on Applied Simulation and Modelling*, Rhodes, Greece, pp. 583-589, Acta Press, 2004

3. E. Qeli, W. Wiechert and B. Freisleben. "Visualizing Time-Varying Matrices Using Multidimensional Scaling and Reorderable Matrices", *in Proceedings of the International Conference on Information Visualization* (IV'04), London, UK, pp. 561-567, IEEE Press, 2004

4. M. Haunschild, A. Wahl, E. von Lieres, E. Qeli, B. Freisleben, R. Takors and W. Wiechert. "MMT 2: Supporting the Modeling Process for Rapid Sampling Experiments", *in Metabolic Engineering V*, Lake Tahoe, USA, 2004

5. E. Qeli, W. Wiechert and B. Freisleben. "3D Visualization and Animation of Metabolic Networks", *in Proceedings of the European Simulation Multiconference*, Magdeburg, Germany, pp. 258-262, SCS Publishing House, 2004

6. E. Qeli, W. Wiechert and B. Freisleben. "Visual Exploration of Time-Varying Matrices", *in Proceedings of the International Conference on Information Visualization* (IV'05), London, UK, pp. 889-895, IEEE Press, 2005

7. E. Qeli, W. Wiechert and B. Freisleben. "The Time-Dependent Reorderable Matrix Method for Visualizing Evolving Tabular Data", *in Proceedings of the IST/SPIE Conference on Visualization and Data Analysis* VDA'05, San Jose, USA, pp. 199-207, IST/SPIE Press, 2005

8. W. Wiechert, M. Haunschild, M. Weitzel, K. Nöh, E. von Lieres, A. Wahl, E. Qeli and B. Freisleben. "Grid Computing for Systems Biology", *in Grid Computing / (eds.): T. Barth, A. Schüll*, pp. 98-133, Vieweg-Verlag, 2006

9. S. Noack, A. Wahl, M. Haunschild, E. Qeli, B. Freisleben and W. Wiechert. "Visualizing Regulatory Interdependencies and Parameter Sensitivities in Biochemical Network Models", *in MATHMOD 5th Vienna Symposium on Mathematical Modelling / (eds.): I. Troch, F. Breitenecker*, Vienna, Austria, (ARGESIM report: 30,1: pp. 17; Volume: 30,2: Full Papers-CD), ARGESIM Verlag, 2006

10. W. Wiechert, M. Haunschild, A. Wahl, E. Qeli, B. Freisleben and M. Oldiges. "Model Families as a Tool for Identifying Regulatory Mechanisms in Intracellular Biochemical Networks", *in Proceedings of International Conference on Systems Biology* ICSB'05, Boston, USA, 2005

11. M. Oldiges, S. Noack, A. Wahl, E. Qeli, B. Freisleben and W. Wiechert. "From Enzyme Kinetics to Metabolic Network Modeling - Visualization Tool for Enhanced Kinetic Analysis of Biochemical Network Models",

in Engineering in Life Sciences (Vol. 6), pp. 155-162, WILEY-VCH Verlag, 2006

12. C. Görg, M. Pohl, E. Qeli and K. Xu. "Visual Representations", *in Human Centered Visualization Environments* Book Chapter, Springer LNCS, to appear, 2007

13. J. Gllavata, E. Qeli and B. Freisleben. "Holistic Comparison of Text Images for Content-Based Retrieval", *in Proc. of the IEEE Int'l Symposium on Multimedia* (ISM'06), San Diego, USA, pp. 299-306, IEEE Press, 2006

14. J. Gllavata, E. Qeli and B. Freisleben. "Detecting Text in Videos Using Fuzzy Clustering Ensembles", *in Proc. of the IEEE Int'l Symposium on Multimedia* (ISM'06), San Diego, USA, pp. 283-290, IEEE Press, 2006

15. E. Qeli and B. Freisleben. "Filtering XML Documents Using XPath Expressions and Aspect-Oriented Programming", *in Proceedings of ACM Symposium on Document Engineering* (DocEng'06), Amsterdam, The Netherlands, pp. 85-87, ACM Press, 2006

16. E. Qeli, J. Gllavata and B. Freisleben. "Customizable Detection of Changes for XML Documents Using XPath Expressions", *in Proceedings of ACM Symposium on Document Engineering* (DocEng'06), Amsterdam, The Netherlands, pp. 88-90, ACM Press, 2006

1.6 Organization of this Thesis

The remainder of this thesis consists of nine chapters.

Chapter 2 introduces the fundamentals of metabolic engineering, with a short introduction to related biological concepts.

Chapter 3 is concerned with the foundations of information visualization. Different aspects of information visualization ranging from cognitive issues to algorithmic solutions are overviewed.

Chapter 4 presents techniques and ideas for visualizing metabolic networks and their timely evolution. MetVis is presented as a tool for accomplishing these purposes. 2D and 3D visualization are discussed and interactive drawing of metabolic networks is introduced.

Chapter 5 describes a novel algorithm which allows customizable comparison of metabolic networks saved as XML files.

Chapter 6 is focused on visualization techniques for time-varying matrices. Techniques such as Multidimensional Scaling, Reorderable Matrix, Cluster

Membership Visualization are introduced or extended. Algorithms for reordering matrices, both locally and globally, are presented.

Chapter 7 proposes a fuzzy clustering ensemble approach for clustering of time-varying multidimensional data. Evaluations are demonstrated on synthetic data sets and sensitivity matrices.

Chapter 8 presents extensions to visualization techniques presented in Chapter 6 for dealing with large-scale data sets. Information visualization paradigms such as distortion, overview+detail and filtering as well a tiled-displays approach are discussed.

Chapter 9 concludes this thesis with a summary of achievements. Furthermore, potential areas for future research are discussed.

2
Foundations of Metabolic Engineering

2.1 Introduction

This chapter introduces the foundations of metabolic engineering. First, a short overview of biological concepts will be given, followed by an introduction to the modeling approach in metabolic engineering. Furthermore, *sensitivity analysis*, which is very important for some parts of this thesis, will be described in detail.

2.2 Biological Background

Life is exceptionally diverse. Many organisms consist of a single cell whereas others consist of billions of cells. Organisms can be found in different environments regardless of the conditions existing in that environment. Diversity in general is translated also to diversity in the molecular level. However, despite this diversity, the same mechanisms and processes are present in all organisms.

The first important fact is that all living things are made of cells which could be abstracted as chemical containers where strictly controlled sequences of reactions occur, which transform a certain substance to another one. The substances that take part in these different reactions and the genetic machinery that controls the production and the interaction of these substances are more or less the same for all livings, from bacteria to humans.

2.2.1 Eukaryotes and Prokaryotes

Based on the structure of their consisting cell(s), creatures are divided in two groups, eukaryotes and prokaryotes. Eukaryotic creatures consist of cells which contain a nucleus; the part of the cell containing the genetic material. Furthermore, other specialized cellular areas, called organelles, are part of eukaryotic cells. Mitochondria and chloroplasts are examples of organelles; chloroplasts are found in plants and their purpose is to catch the sunlight energy whereas mitochondria produce energy i.e. adenosintriphosphate (ATP) for the cell. All multicellular organisms are eukaryotes and many single celled organisms such as yeasts are eukaryotes, too. Figure 2.1(a) shows a schematic view of an eukaryotic cell with its components.

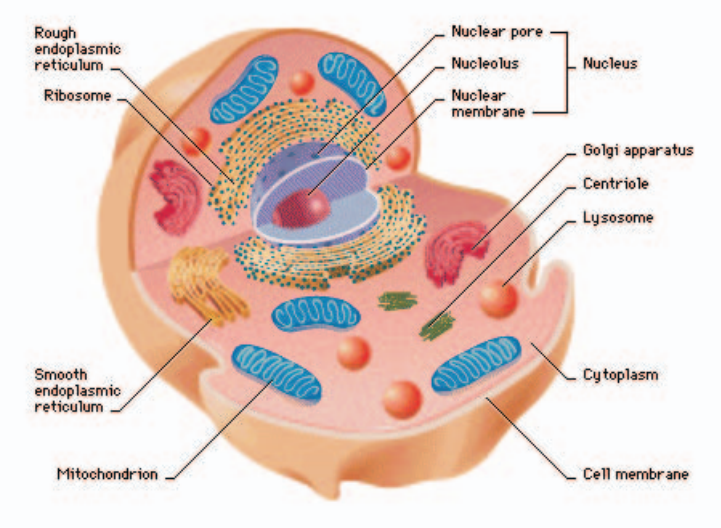
Prokaryotes are distinguished from eukaryotes by the fact that they lack a cell nucleus. Two major groups are distinguished in prokaryotes: Archaea and Bacteria.

Archaea are a recently discovered class of organism, which lives in extreme conditions such as environments with extremely high temperatures or highly alkaline or acid properties. They were first considered to belong to the same group as bacteria in the universal phylogenetic tree, which was later dismissed by Woese [Woese]. Although at the structural level they are prokaryotes, at the genetic level they have more similarities with eukaryotes.

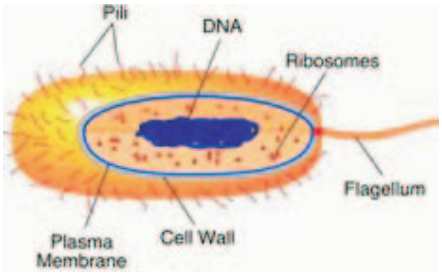
2.2.2 Bacteria

Bacteria are single-celled organisms, which are ubiquitous in soil and water or live in symbiosis with other organisms. There are millions of them everywhere on this page or in the air we are breathing. Bacteria reproduce themselves asexually by cell division and their most important activity is to produce more bacteria. Natural selection generally favors bacteria based on the ability of reproducing as fast as possible when enough food supplies are available [AJL+02, Hun93]. To move, bacteria use flagella or flagella-like structures (see Figure 2.1(b)). Bacteria come in a variety of different shapes such as rod-shaped, sphere or helix-shaped and although shape is not considered a defining factor in the classification of bacteria, their name is often derived from the shape e.g. Bacillus, Streptococcus or Staphylococcus. Bacteria are far more diverse than mammals or insects. For example, the genetic distance between E. coli and Thermus aquaticus is greater than the distance between humans and oak trees.

Bacteria are both harmful and useful to animals and humans. Different diseases and infections in humans are caused by bacteria. Furthermore, bacteria can also cause different diseases in plants. On the other side, bacteria are useful for transforming wastes into environment friendly substances. Bacteria have also been successfully used to clean oil spills. Combined with yeasts,



(a) An Eukaryotic cell (image taken from Microsoft© Encarta© Encyclopedia, `http://encarta.msn.com/media_461540224_761568585_-1_1/Animal_Cell.html`)



(b) A Prokaryotic cell (image taken from `http://nai.nasa.gov/library/images/news_articles/94_1.jpg`)

Figure 2.1: Schematic views of Eukaryotic and Prokaryotic cells.

bacteria are also used to produce fermented foods such as cheese or yogurt. Another useful aspect of bacteria is their ability to live as symbiants and help humans or other organisms, for example, in the digestion process.

2.2.3 The Organism Escherichia coli

Escherichia coli (usually abbreviated to E. coli) is one of the main species of bacteria that live in the lower intestines of warm-blooded animals (including birds and mammals) and are necessary for the proper digestion of food. Its

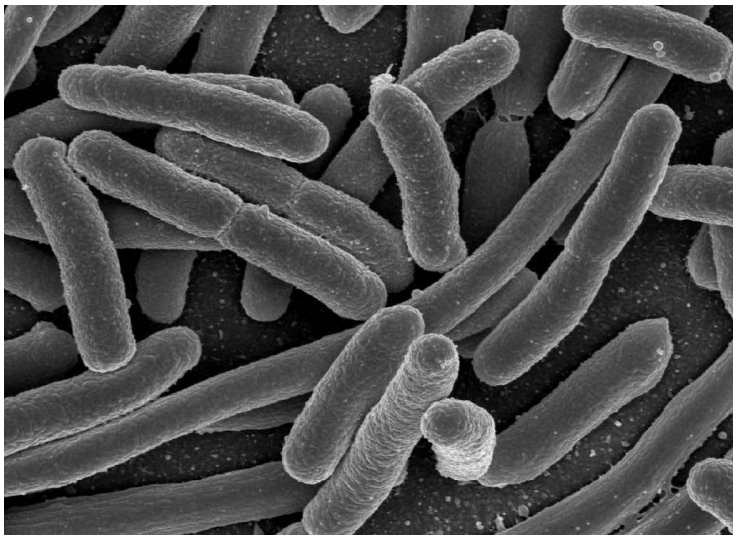


Figure 2.2: E.coli colony (taken from `http://www3.niaid.nih.gov/NR/rdonlyres/49477C30-0513-47BE-88FC-17974CB1F952/0/e_coli.jpg`)

presence in groundwater is a common indicator of pollution. The name comes from its discoverer, Theodor Escherich.

E. coli is successfully used in industry in the production of insulin, tryptophan and ethanol [VBP93]. E. coli has the ability to survive outside its native living environment and to reproduce itself quickly. In normal conditions, it reproduces itself by division every 45 minutes and in optimal conditions in 37 degrees Celsius with a pH value between 6.5 and 7.3 every 22 minutes. In these conditions, a colony consisting of more than 1 billion cells can be obtained from a single cell in only 20 hours. E.coli has different sensors which signal the presence of chemicals in its environment. By rotating its flagella, E. coli can move away from toxic substances or in the direction of food, if needed. Populations of E. coli cells with particular characteristics are known as strains, such as, for example, strains living in the intestine of humans or those living in birds. Some of those strains could be harmful to the host organism.

Because of its ubiquity and the available knowledge regarding its genetic structure and the mechanisms underlying its metabolism, E. coli is frequently studied in cellular biology. Several databases in the Internet have published information regarding the sequencing of the genome of E. coli [eco].

The complex mechanisms underlying the functioning of E. coli cells are the focus of many studies. However, in the context of this work, only the metabolism of these cells is important.

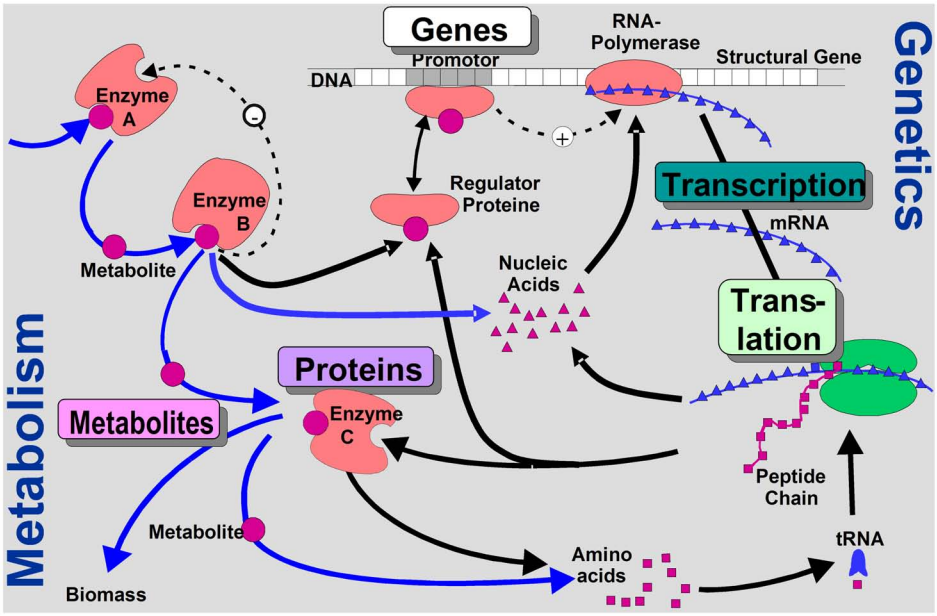


Figure 2.3: Schematic representation of the major processes taking place in a living cell

2.2.4 Processes Inside the Cell

Based on the reductionism approach (see Section 1.1), researchers have studied the elementary components of living cells such as DNA, RNA, ribosome, proteins, metabolites, etc. These components, however, do not function alone; they interact together in different processes like in gene translation or transcription, enzyme catalysis or enzyme regulation. These interactions are represented visually in Figure 2.3 [LDS99]. The DNA stores the genetic information regarding the synthesis of proteins. In the process of transcription, messenger RNA (mRNA) copies are made of bases along one strand of DNA which is obtained in the process of RNA polymerase. The information contained in this mRNA is used to create the respective amino acid sequence of the polypeptide chain during translation. These polypeptide chains fold and combine into three dimensional structures of proteins. Therefore, control of the cellular machinery requires tight regulation of the transcription and translation processes. This regulation is accomplished by the promoters, namely nucleotide sequences, which themselves are controlled by regulatory proteins.

The process of activation of genes in the production of proteins is called

expression. Considering that some proteins are always needed inside the cell, they are always expressed, while for other proteins their gene expression changes according to their need. Proteins have different functions in the organism:

- Regulatory proteins which are divided into hormones and transcription factors. Hormones such as insulin regulate the metabolism of cells, whereas transcription factors bind to target genes and regulate the gene expression by preventing or activating the transcription of the respective proteins.

- Transport proteins are responsible for the transportation of substances, e.g. Hemoglobin transports the oxygen in blood.

- Storage proteins which serve as reservoirs of nutrients like ovalbumin (egg white).

- Contractile and motile proteins that allow cell and tissue movement, such as actin and myosin in muscles.

- Structural proteins, such as collagen, which is the structural protein for bone connective tissues.

- Protective proteins, such as immunoglobulin or antibodies, which are the proteins of the immune response.

- Exotic proteins that are found in certain plants or other organisms for special purposes, such as glue proteins in mussels.

- And last but not least, enzymes which are catalysts that interact with the reactants in a particular reaction and increase the reaction rate of that reaction up to 20 million times.

  Enzymes are biological catalysts, meaning that they are produced or derived from some living organism and are not changed or destroyed by the chemical reaction that they accelerate. These catalysts are specific in nature to the type of reaction they can catalyze. Each enzyme can act to catalyze only very specific chemical reactions and only with very specific substances. An enzyme can be described as a key, which unlocks complex compounds based on its structure and shape. The substrate i.e. the compound or substance which takes part in the reaction, can undergo the change only after it is coupled with the enzyme. After that, the enzyme is released and can be used again. The role of enzymes is crucial because in normal conditions (i.e. body temperature), certain processes such as the oxidation of glucose, which is very important for producing energy for plants or animals, proceeds at a very slow

rate. Thus, enzymes allow reactions that are necessary to sustain life to proceed relatively quickly at the normal environmental temperatures.

2.2.5 Metabolism and Metabolic Networks

From a chemical perspective, living organisms are open thermodynamic systems far from equilibrium. Furthermore, cellular constituents are in a chemically more reduced state than the matter that surrounds them. Nevertheless, cells must somehow extract energy from the environment and use this energy to maintain the chemical activities that occur inside the living organism. The sum of all these processes constitutes the metabolism of the living organism. On the other side, metabolism consists of two major realms: the degradative pathways of catabolism and the biosynthetic pathways of anabolism (see [Fel97]). A pathway is a sequence of reactions, which transform a certain substance to an end product. This sequence of reactions is controlled by enzymes, which were described in the previous section. Catabolic pathways result in the conversion of complex molecules into simple compounds releasing energy, whereas anabolic pathways use the released energy to convert simple compounds into complex biomolecules. Usually, the pathway for the biosynthesis of a biomolecule is inherently more complex than the pathway for its degradation, because anabolic pathways must be coupled to many other exergonic reactions, i.e. reactions that generate energy. For example, the conversion of phosphoenolpyruvate (PEP) to pyruvate proceeds in a single reaction during glycolysis, whereas the reversal proceeds in two stages during glyconeogenesis. The whole metabolic pathways inside a cell constitute the *Metabolic Network* of the cell. Figure 2.4 shows the entire metabolic network of an E. coli cell. The points indicate the metabolites (substances) inside the cell, whereas lines indicate reactions i.e. transformations from one substance to the other.

2.2.6 Experimental Basis for the Study of Metabolic Networks

Modern developments in measurement techniques have made it possible to explore the intracellular properties of the cell *in vivo*, meaning in the living cell. The data generated from these automatized experiments is then coupled with the biochemical knowledge for building a basis of the modelers work. In this context, the research is concentrated in several fields:

- **Genomics** is concerned with the genome, i.e. the genes of an organism. One of its major applications is *Comparative Genomics* with the

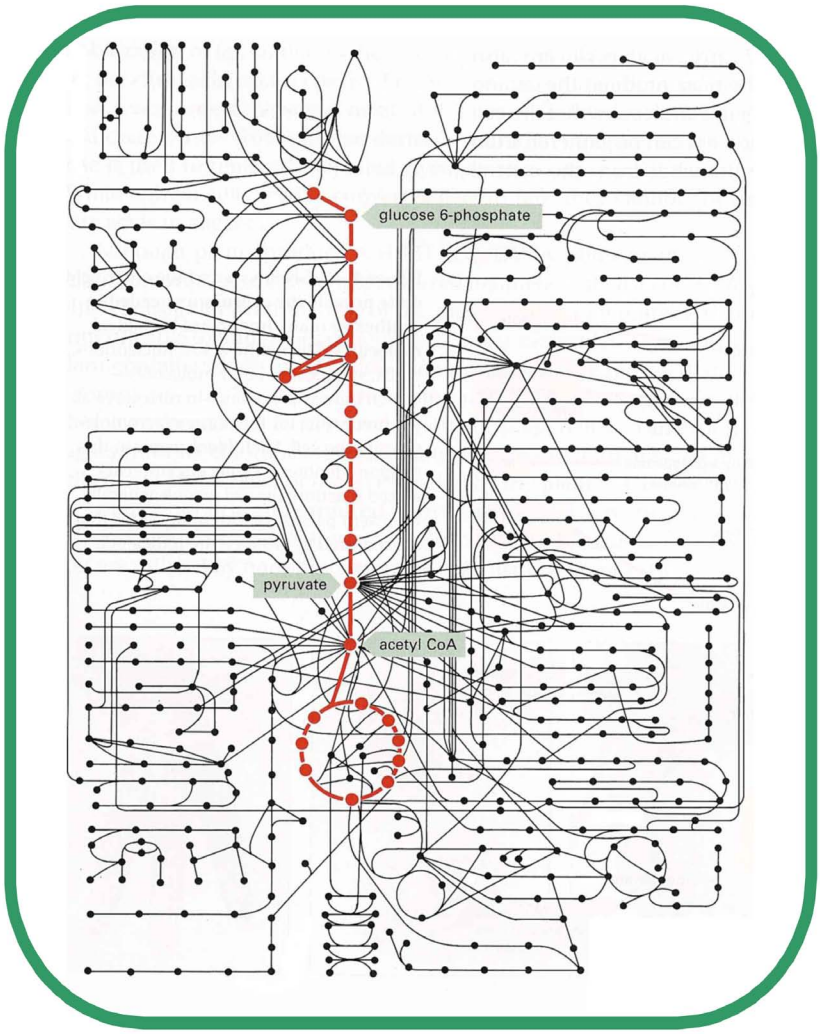


Figure 2.4: Metabolic network of the E. coli. Nodes represent metabolites, whereas edges represent reactions. The highlighted part represents the central metabolism.

purpose of clearing different phylogenetic relationships between organisms.

- **Transcriptomics** is concerned with the gene expression under different experimental conditions. Various methods are used for this purpose, but one of the most popular methods uses *DNA Microarrays*. The main idea here is to study the concentration of mRNA in a cell under different physiological conditions, obtaining thus large matrices containing information about different genes in different experimental conditions. These data is further processed using various statistical methods.

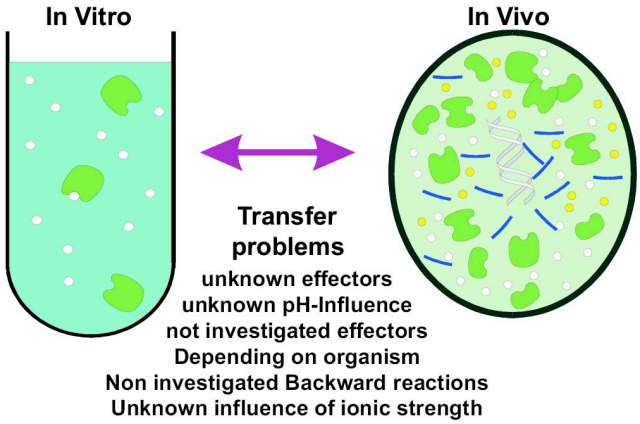


Figure 2.5: Difference between in Vivo and in Vitro approaches

- **Proteomics** is concerned with the study of proteins and their interactions, thus searching to deduce information about the phenotype of the cell.

- **Metabolomics** and **Fluxomics** are concerned with the concentrations of metabolites or the rate of intracellular reactions, respectively, by using LC-MS (Liquid Chromatography Mass-Spectrometry) methods in case of metabolites or NMR with C^{13} marking in case of intracellular fluxes.

For exploring intracellular properties of the cells, two types of experiments are performed: *In Vitro* and *In Vivo Experiments*. The basic idea of the in vitro approach (experiments performed in the test-tube) is to study the interaction of the isolated enzymes with the substances which are inside the cell. However, the knowledge gained from *in vitro* experiments should be validated further by using *in vivo* experiments i.e. in the living cell (similar to the reductionism approach, components of the cell in this case enzymes, are studied separately first, and later it is sought to find their place in the bigger picture, i.e. in the whole cell). The level of enzymes studied *in vitro* is much lower than *in vivo*. Furthermore, the effect of other substances, which could possibly affect the activity of enzymes, cannot be studied in *in vitro* experiments.

Figure 2.5 illustrates the difference between the in vivo and in vitro approaches. Kitano [Kit02] writes in his overview paper about systems biology that, "To understand biology at the system level, we must examine the structure and dynamics of cellular and organismal function, rather than the characteristics of isolated parts of a cell or organism.".

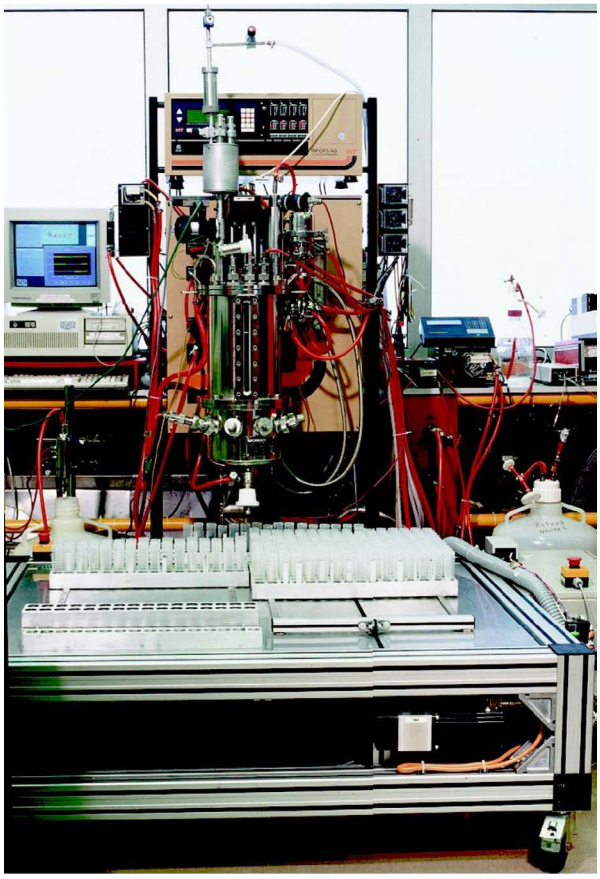


Figure 2.6: The equipment used in the Research Center Jülich for rapid sampling experiments

Rapid Sampling and Substrate-Pulse Experiments [HFTW05] are broadly used approaches for carrying out in vivo experiments. The first concept, rapid sampling, is related to the time frame used to sample the cell contents. One assumption used in this case is that there is no cell reproduction i.e. division during the sampling time, thus restricting the time frame used for the sampling to avoid these cell divisions. A short time frame means that the metabolism of the cell has a constant activity, thus being not very interesting to study. To avoid this situation, substrate-pulse experiments are used. The metabolism of the cell is affected externally by overfeeding the cells with the help of a substrate pulse. After the substrate pulse, a time frame of several seconds remains to sample the cells in order to measure the concentration of metabolites. However, special equipment is needed to successfully implement these types of experiments. Figure 2.6 shows the equipment used for rapid sampling at the Research Center Jülich, whereas Figure 2.7 shows

the process of performing rapid sampling experiments. Cells are cultivated earlier in a bioreactor using feed batch culture. To continue its normal life, the cells in the bioreactor consume the glucose until it is fully exhausted. At this moment, the metabolism of the cell comes in a quasi-stationary state. The sampling process begins five seconds before a glucose pulse is given and continues another 35 seconds after the pulse has been given. Samples are taken every 220 ms, resulting in 20 samples from the steady state and 144 samples after the pulse. The samples are then inserted in reactant glasses filled with methanol at -50 °C, stopping instantly every cell activity. These reactant glasses are later stored at -28 °C.

The whole procedure is time-consuming: all the test tubes must be analyzed and the metabolites inside them must be quantified. This takes from several months until two years of time. The metabolites are separated from the methanol solution and the enzymes (so that they do not interact anymore) using several centrifugal processes. The final concentration of metabolites is determined using LC-MS techniques. Imprecision occurs although latest technical possibilities are used. Furthermore, metabolites having the same weight, such as G6P and F6P, cannot be distinguished from each other, a fact that should be kept in mind during the later use of quantized data.

2.3 Modeling and Simulation of Metabolic Networks

High-throughput devices have produced a tremendous amount of data about the molecular mechanisms of living cells. This data comes either in the form of experimental data, as described in Subsection 2.2.6 where data about the metabolite concentrations inside the cell is obtained, or in the form of data related to genes and proteins accessible in public databases. However, all these data separated from each other are like parts of a puzzle. By inserting them in appropriate mathematical models, more knowledge can be gained as the systems level about the functioning of the cell. In this way, complex mathematical models are created, which in contrast to the complex biological system (e.g. the metabolism of the cell), possess an unambiguous meaning and thus could be used to facilitate the understanding of metabolism. On the other hand, the existing complexity of metabolism is transferred also to a certain degree to the model, making the model dependent on other techniques related to simulation such as *sensitivity analysis* and optimization [Wie02].

However, optimistic views such as those expressed by the authors of E-CELL [THT+99] predict that cellular processes may be understood by *in silico* experiments, whereas more skeptical views such as those presented by Westerhoff [Wes01] have their remarks in this issue [WT04].

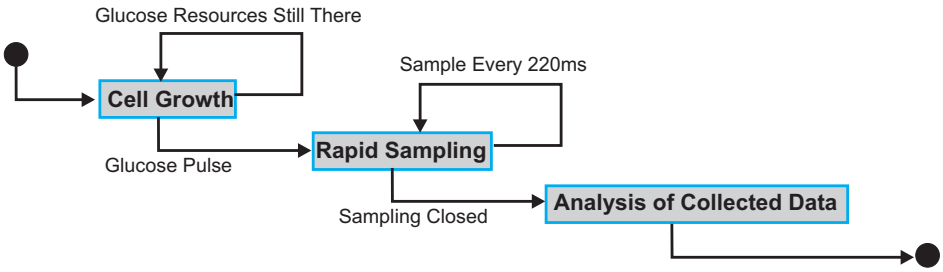


Figure 2.7: Description of the process of rapid sampling

- In contrast to discrete data, e.g. data related to the genome which (almost) never changes, the physiological conditions are continuous and depend on time. On the other hand, in vivo conditions cannot be reconstructed in a test tube, meaning that in vitro data are not equal to in vivo data.
- In vivo data sampled from whole cell experiments do not give any information about specific units of the cellular system.
- Some cellular processes such as protein synthesis are still unknown. There is a lack of information about active enzymes, and thus the maximal rate of specific reactions $vmax$ is unknown.
- The in vivo data collected and published in databases is often gathered under different experimental conditions, making their combination questionable.

For several reasons, one of the commonly used approaches to start the modeling of cells is by using only the central metabolism of the cell [Wie02], which is represented in Figure 2.8.

- Many, but not all, major intracellular metabolite concentrations can be measured by different mass spectrometry (MS) methods. By performing non-stationary pulse experiments and using rapid sampling, the time course of the metabolic response can be precisely monitored.
- In steady state, most metabolic fluxes in the central metabolism can be determined by 13C flux analysis. By repeating several stationary experiments under different physiological conditions, a portrait of metabolic regulation can be obtained.
- Enzyme activities can be roughly determined from cell extracts.
- Kinetic parameters of most enzymes (although determined in vitro) are collected in databases. At least the K_m (Michaelis-Menten constant) values are usually assumed to be reliable.

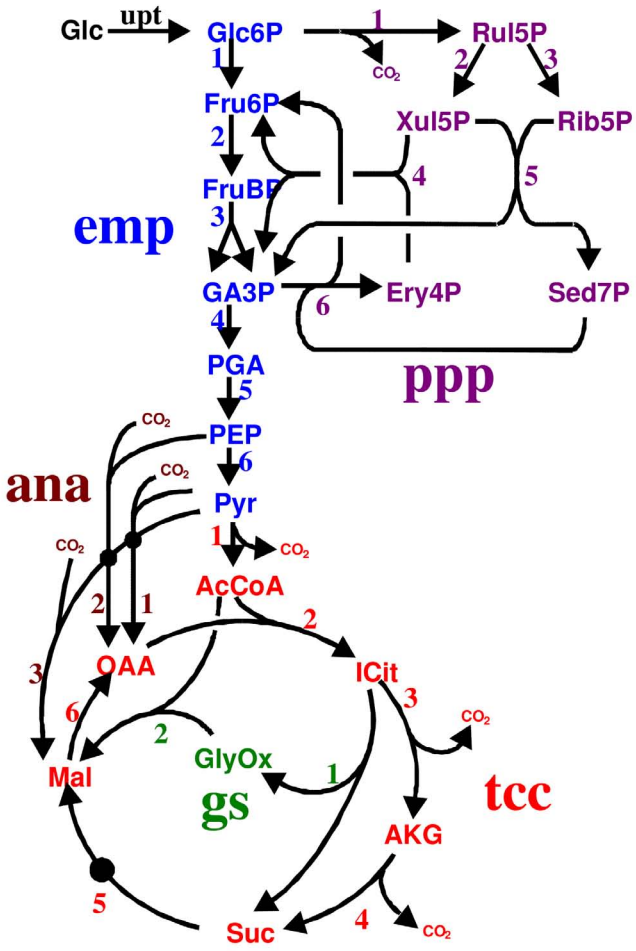


Figure 2.8: The central metabolism of the cell

2.3.1 General Metabolic Modeling Strategy

Before describing metabolic models in detail, some major assumptions about modeling cells are made:

- It is assumed that the differences in cell population regarding length, age and other factors affecting the metabolism are negligible, taking into consideration *average cells*.
- The effect of the cell cycle, i.e. the different steps between two cell divisions, to the central metabolism are also considered to be negligible.
- The metabolites are distributed uniformly throughout the cell and the stochastic effects related to them, which occur mainly when the con-

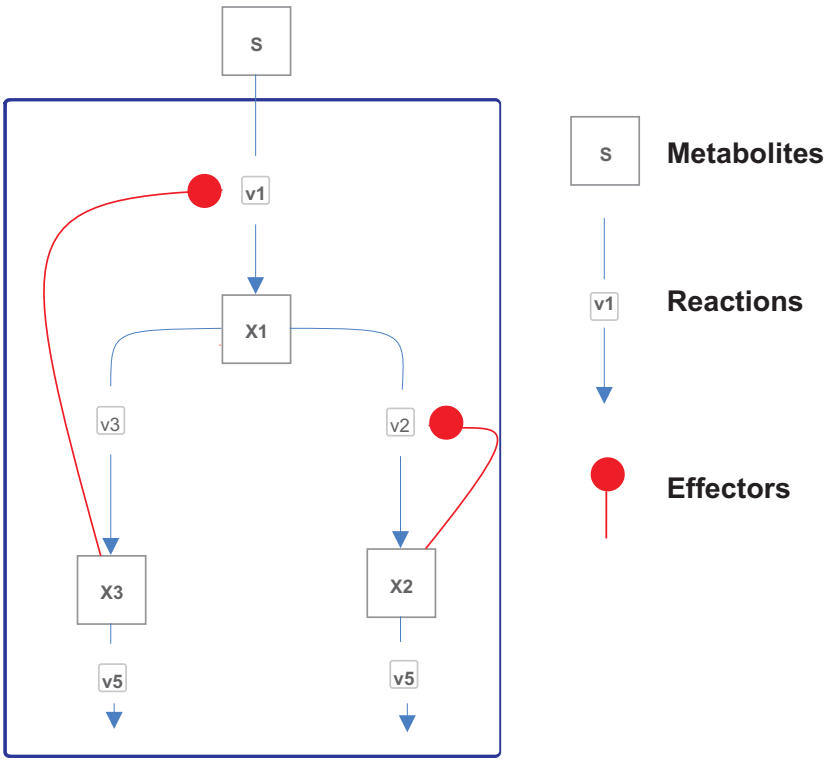


Figure 2.9: A simple branched network with inhibition

centrations of metabolites are extremely low, are neglected because the central metabolism deals with metabolites with high concentrations and reactions with fast rates. Furthermore, the diffusion process occurring around the cell membrane is supposed to run quickly.

- There are no gradients with respect to temperature, pH value, pressure or nutrients in the bioreactor due to the small size of the reactor and the short mixing times. Furthermore, the effect of gradients inside the cell itself is not considered.

- It is known that cells adapt to their environment by changing the concentration of respective enzymes, thus adjusting their metabolism to the environment. These effects are neglected in the presented models due to the short duration of the experiments of 20-40 seconds.

To illustrate the process of building a metabolic model, an example from [WT04] is adapted. Figure 2.9 shows a sample metabolic network with four metabolites and five reactions. Furthermore, two of these reactions are inhibited by their respective metabolites. From the graph theoretic point of view, a metabolic network could be modeled either as a directed hypergraph, meaning that some edges, which represent reactions could connect more than

two nodes which represent metabolites, or as a directed digraph with edges beginning at the metabolite nodes and ending at reaction nodes or beginning at reaction nodes and ending at metabolite nodes.

Stoichiometric Matrix of Metabolic Network Models. The stoichometric matrix of a metabolic model is defined similar to the adjacency matrix of normal graphs. In contrast to the adjacency matrix which shows the connectivity relationships in a graph, the stoichometric matrix shows the relationships between metabolites and reaction nodes, without forgetting the direction of the flow in the metabolic network. For this purpose, there is a positive coefficient when a metabolite is the output of the reaction and a negative sign otherwise; the respective coefficients are multiplied with their order in the respective reactions, which in this case is one. The metabolite S is an external metabolite and its trajectory is considered as known beforehand. The stoichometric matrix does not represent all the information present in the graphical representation of the metabolic network, since the effectors, i.e. inhibitors and activators, are not present. Equation 2.1 presents the stoichometrix matrix N of the simple metabolic network shown in Figure 2.9.

$$\mathbf{N} = \begin{array}{c} \\ X_1 \\ X_2 \\ X_3 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{pmatrix} +1 & -1 & -1 & 0 & 0 \\ 0 & +1 & 0 & -1 & 0 \\ 0 & 0 & +1 & 0 & -1 \end{pmatrix} \end{array} \tag{2.1}$$

A reaction named v_1 of the form $aA + bB = cC + dD$ would affect the stoichiometric matrix of the respective metabolic network as follows:

$$\mathbf{N_{imaginary}} = \begin{array}{c} \\ \vdots \\ A \\ B \\ C \\ D \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccc} \dots & v_1 & \dots \end{array} \\ \begin{pmatrix} \vdots & \vdots & \vdots \\ \dots & -a & \dots \\ \dots & -b & \dots \\ \dots & +c & \dots \\ \dots & +d & \dots \\ \vdots & \vdots & \vdots \end{pmatrix} \end{array} \tag{2.2}$$

Mathematical Formulation of the Metabolic Model. If we consider $X(t)$ as the vector of metabolite concentrations and $v(t)$ as the vector of reaction rates, then the short form of the system of differential equations

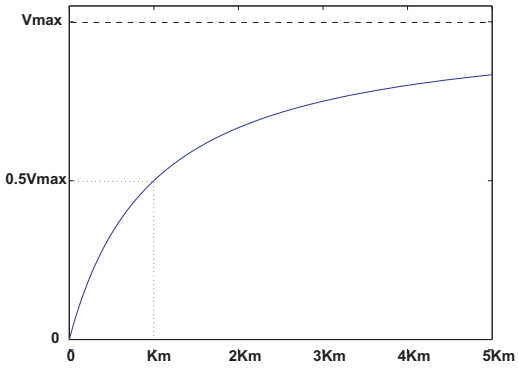


Figure 2.10: The relationship between the substrate and reaction rate according to the Michaelis-Menten equation [CB95]

describing the specific metabolic network would be:

$$\dot{\mathbf{X}} = N \cdot v \tag{2.3}$$

where $X(0) = X_0$. v is a function of a parameter set α and the vector of metabolite values including the input metabolite X and S. Thus

$$v = v(\alpha, S, X) \tag{2.4}$$

Here, *enzyme kinetics* come into play. The reactions taking part in a metabolic network are catalyzed by at least an enzyme. The rates of enzyme-catalyzed reactions have been studied since the nineteenth century. The *Michaelis-Menten* equation is one of the fundamental equations of enzyme kinetics.

$$v = \frac{V_{max} \cdot S}{K_m + S} \tag{2.5}$$

Here, V_{max} represents the maximum velocity of the reaction (although it is a limit and not a maximum in the mathematical sense) and K_m represents the Michaelis-Menten constant (see Figure 2.10). The inhibitory effects can be included in the Michaelis-Menten equation by adding one (or more) multiplicative term(s) expressing inhibition, as shown in Equation 2.6

$$v = \frac{V_{max} \cdot S}{K_m + S} \cdot \frac{K_I}{K_I + X} \tag{2.6}$$

where K_I represents the inhibition constant and X is the metabolite which has an inhibiting effect on the specific reaction. The normal and inhibited form of the Michaelis-Menten equation are only two possibilities that can be chosen from more than 33 possible kinetic laws [HFS+03].

For the network of Figure 2.9, the model after substituting the respective Michaelis-Menten equations would look like:

$$\begin{aligned} v_1 &= \frac{V_{max,1} \cdot S}{K_{m,1} + S} \cdot \frac{K_{I,1}}{K_{I,1} + X_3} \\ v_2 &= \frac{V_{max,2} \cdot X_1}{K_{m,2} + X_1} \cdot \frac{K_{I,2}}{K_{I,2} + X_2} \\ v_3 &= \frac{V_{max,3} \cdot X_1}{K_{m,2} + X_1} \\ v_4 &= \frac{V_{max,4} \cdot X_2}{K_{m,4} + X_2} \\ v_5 &= \frac{V_{max,5} \cdot X_3}{K_{m,5} + X_3} \\ \dot{X}_1 &= v_1 - v_2 - v_3 \\ \dot{X}_2 &= v_2 - v_4 \\ \dot{X}_3 &= v_3 - v_5 \end{aligned} \tag{2.7}$$

or in matrix form:

$$\begin{pmatrix} \dot{X}_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} +1 & -1 & -1 & 0 & 0 \\ 0 & +1 & 0 & -1 & 0 \\ 0 & 0 & +1 & 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} \frac{V_{max,1} \cdot S}{K_{m,1}+S} \cdot \frac{K_{I,1}}{K_{I,1}+X_3} \\ \frac{V_{max,2} \cdot X_1}{K_{m,2}+X_1} \cdot \frac{K_{I,2}}{K_{I,2}+X_2} \\ \frac{V_{max,3} \cdot X_1}{K_{m,2}+X_1} \\ \frac{V_{max,4} \cdot X_2}{K_{m,4}+X_2} \\ \frac{V_{max,5} \cdot X_3}{K_{m,5}+X_3} \end{pmatrix} \tag{2.8}$$

In practice, the modeling process is an iterative approach trying to explain the available data with the help of mathematical models. Thus, the process usually starts with a simple model and at each iteration step some new component(s) of the model such as new reaction steps are added to the old model. The operations modifying the old model fall into several categories [WT04]:

- A reaction step is added, removed or replaced by another one.
- A reaction kinetic term is extended, simplified or replaced by another one.
- The kinetic parameters are estimated by a parameter-fitting algorithm.

This procedure eventually leads to a family of models, which contain the accumulated knowledge about the biological system, as shown in Figure 2.11. In this scenario, the models M_1, M_2 and M_3 belong to the first generation

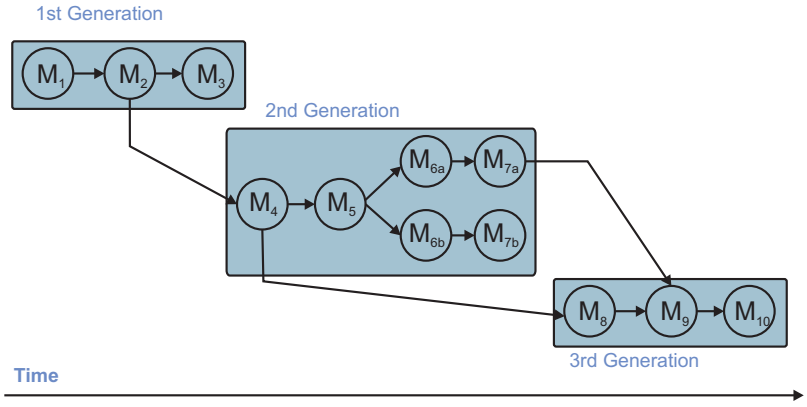


Figure 2.11: Graphical representation of a family of models (see [WT04])

of the created models and are derived from each other. On a later point of time, a new generation of models is constructed, which is derived from model M_2, developed further and then divided into two branches. Finally, the last generation of models is constructed, which uses knowledge derived from the models M_4 and M_{7a}.

Formally, this model set can be described with the set of systems of differential equations as presented in Equation 2.9.

$$\dot{X}^i = N^i \cdot v^i(\alpha^i, S^i, X^i), X^i(0) = X_0^i \tag{2.9}$$

2.3.2 Sensitivity Analysis of Metabolic Network Models

Sensitivity Analysis plays an important role for analyzing the current model and for evolving into a new one. It is widely used to analyze complex nonlinear models [CH88, SCS00]. Its purpose is to separate important parameters from less important ones, thus being a very important tool for model simplification.

Typical questions that could be asked by sensitivity analysis are:

- Which factors i.e. parameters have the biggest impact on the output of the model and should thus be investigated in detail?
- Which parameters are insignificant and could thus be omitted in a newly derived model?
- How good describes a model the considered biological system?

Sensitivity analysis is treated both in the general case of modeling [SCS00,

CH88] and in the specific case of modeling metabolic networks [WT04, HW03, IS03, MAR97, Koh02].

The sensitivity coefficients of a regression model are the partial derivatives of all computed quantities (i.e. fluxes and concentrations) with respect to all the parameters of the model (i.e. kinetic constants, substrate input concentration), which could be arranged in a Jacobian matrix. It should be pointed out that sensitivities, as it is always the case with derivatives, are just local approximations of the system behavior, i.e. they approximate the system behavior in the neighborhood of the chosen linearization point. A certain parameter may have a strong influence on the system behavior for certain values of X but little influence in other situations. The non-stationary model was described shortly in Subsection 2.3.1 and has the form:

$$\dot{\mathbf{X}} = f(X, \alpha) = N \cdot v(\alpha, S, X) \tag{2.10}$$

where $X(0) = X_0$. The local sensitivity functions for this model are defined as follows:

$$S_{i,k}(t) = \frac{\partial X_i(t, \alpha)}{\partial \alpha_k} \quad i = 1..m, k = 1..p \tag{2.11}$$

where m and p are the respective dimensions of metabolite concentrations vector X and parameters vector α. The calculation of equation 2.11 can be effected in two ways: (I) approximation via finite differences; and (II) analytical solution of sensitivity equations.

Approximation via Finite Differences is obtained by applying the central differences approach.

$$S_{i,k}(t) \approx \frac{X(t, \alpha_k + \Delta\alpha_k, \alpha_{j=1..p, j\neq k}) - X(t, \alpha_k - \Delta\alpha_k, \alpha_{j=1..p, j\neq k})}{2\Delta\alpha_k} \tag{2.12}$$

However, the accuracy of this method depends on the value of $\Delta\alpha_k$, which should be suitably small to obtain good results.

Analytical Solution of Sensitivity Equations is based on calculating the partial derivatives directly, obtaining in this way a considerably more precise solution. From Equation 2.10 we obtain:

$$\frac{\partial \dot{\mathbf{X}}}{\partial \alpha} = \frac{\partial f}{\partial X}\frac{\partial X}{\partial \alpha} + \frac{\partial f}{\partial \alpha} \tag{2.13}$$

with $\frac{\partial X}{\partial \alpha}(0) = 0$. Transforming Equation 2.13 further we obtain:

$$\frac{\partial \dot{\mathbf{X}}}{\partial \alpha} = N \cdot (\frac{\partial v}{\partial X}\frac{\partial X}{\partial \alpha} + \frac{\partial v}{\partial \alpha}) \tag{2.14}$$

Similarly, the systems of sensitivity equations with respect to extra cellular metabolites i.e. substrates, are obtained as shown in Equation 2.15 and with respect to initial values in Equation 2.16.

$$\frac{\partial \dot{\mathbf{X}}}{\partial S} = N \cdot (\frac{\partial v}{\partial X}\frac{\partial X}{\partial S} + \frac{\partial v}{\partial S}) \tag{2.15}$$

with boundary condition $\frac{\partial X}{\partial S}(0) = 0$.

$$\frac{\partial \dot{\mathbf{X}}}{\partial X_0} = N \cdot (\frac{\partial v}{\partial X}\frac{\partial X}{\partial X_0}) \tag{2.16}$$

where the boundary condition is $\frac{\partial X}{\partial X_0}(0) = I$ [WT04]. Rewriting Equation 2.14 in matrix form, we obtain:

$$\frac{\partial \dot{\mathbf{X}}}{\partial \alpha} = N \cdot (J \cdot S + F) \tag{2.17}$$

The matrices J and F contain the partial derivatives of reaction rates with respect to concentrations and parameters respectively, whereas the matrix S is the sensitivity matrix.

Returning to the example model of Equation 2.8 and Equation 2.8 we have:

$$\frac{\partial \dot{\mathbf{X}}}{\partial \alpha} = N \cdot \begin{pmatrix} \frac{\partial r_1}{\partial X_1} & \frac{\partial r_1}{\partial X_2} & \frac{\partial r_1}{\partial X_3} \\ \frac{\partial r_2}{\partial X_1} & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ \frac{\partial r_5}{\partial X_1} & \frac{\partial r_5}{\partial X_2} & \frac{\partial r_5}{\partial X_3} \end{pmatrix}$$
$$\left[\begin{pmatrix} \frac{\partial X_1}{\partial V_{max,1}} & \frac{\partial X_1}{\partial K_{m,1}} & \frac{\partial X_1}{\partial K_{I,1}} & \cdots & \frac{\partial X_1}{\partial K_{m,5}} \\ \frac{\partial X_2}{\partial V_{max,1}} & \frac{\partial X_2}{\partial K_{m,1}} & \frac{\partial X_2}{\partial K_{I,1}} & \cdots & \frac{\partial X_2}{\partial K_{m,5}} \\ \frac{\partial X_3}{\partial V_{max,1}} & \frac{\partial X_3}{\partial K_{m,1}} & \frac{\partial X_3}{\partial K_{I,1}} & \cdots & \frac{\partial X_3}{\partial K_{m,5}} \end{pmatrix} + \begin{pmatrix} \frac{\partial v_1}{\partial V_{max,1}} & \frac{\partial v_1}{\partial K_{m,1}} & \frac{\partial v_1}{\partial K_{I,1}} & \cdots & \frac{\partial v_1}{\partial K_{m,5}} \\ \frac{\partial v_2}{\partial V_{max,1}} & \frac{\partial v_2}{\partial K_{m,1}} & \frac{\partial v_2}{\partial K_{I,1}} & \cdots & \frac{\partial v_2}{\partial K_{m,5}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial v_5}{\partial V_{max,1}} & \frac{\partial v_5}{\partial K_{m,1}} & \frac{\partial v_5}{\partial K_{I,1}} & \cdots & \frac{\partial v_5}{\partial K_{m,5}} \end{pmatrix} \right] \tag{2.18}$$

where the sensitivity matrix is:

$$S = \begin{pmatrix} \frac{\partial X_1}{\partial V_{max,1}} & \frac{\partial X_1}{\partial K_{m,1}} & \frac{\partial X_1}{\partial K_{I,1}} & \cdots & \frac{\partial X_1}{\partial K_{m,5}} \\ \frac{\partial X_2}{\partial V_{max,1}} & \frac{\partial X_2}{\partial K_{m,1}} & \frac{\partial X_2}{\partial K_{I,1}} & \cdots & \frac{\partial X_2}{\partial K_{m,5}} \\ \frac{\partial X_3}{\partial V_{max,1}} & \frac{\partial X_3}{\partial K_{m,1}} & \frac{\partial X_3}{\partial K_{I,1}} & \cdots & \frac{\partial X_3}{\partial K_{m,5}} \end{pmatrix} \tag{2.19}$$

The values obtained for the sensitivity matrix are not normalized; in order to obtain comparable values, the sensitivities are normalized using the respective concentration and parameter values according to Equation 2.20:

$$S_{i,k}(t) = \frac{\partial X_i(t,\alpha)}{\partial \alpha_k} \cdot \frac{\alpha_k}{X_i(t)} \quad i = 1..m, k = 1..p \tag{2.20}$$

which is based on the fact that:

$$\frac{\partial ln(x)}{\partial ln(p)} = \frac{\partial ln(x)}{x} \cdot \frac{\partial x}{\partial ln(p)} = \frac{\partial ln(x)}{x} \cdot \frac{\partial x}{\partial p} \cdot \frac{\partial p}{\partial ln(p)} = \frac{p}{x} \cdot \frac{\partial x}{\partial p}$$

2.4 Summary

This chapter presented an introduction to the field of metabolic engineering. Basic background in cell biology as well as techniques for modeling metabolism of the cell were given. Furthermore, an introduction to the technique of sensitivity analysis, as an important step in model building is given. The output of sensitivity analysis, namely the time-varying sensitivity matrices are discussed in several chapters of this thesis.

3
Foundations of Information Visualization

3.1 Introduction

In Section 1.2, we gave a short introduction to information visualization. This section will give a deeper overview in the field of information visualization by touching different aspects ranging from psychological ones to aspects dealing with algorithmic problems occurring in this field.

3.2 Background

Visualization, when used properly, can help in the process of extracting insight from data during decision-making. Its advantages consist of the ability to rapidly interpret large quantities of data.

The challenges in this context are in enabling the user to visually approach the data so that the user can understand and perceive the data effectively, find the information the user is looking for and to provide interaction methods which allow effective communication between the user and the data.

Information visualization, as already defined in Section 1.2, deals with data that is usually abstract, high-dimensional, and structured in a complex way. Information visualization is not necessarily bound to the computer, although the usage of computers has given a boost to visualization. Indeed, simple examples of information visualization are found since the 19th century. Figure 3.1 presents one such example of visualization, showing the distribution of cholera death cases in a certain part of London during the epidemy of 1854, drawn by the epidemiologist Dr. John Snow ([Tuf83]). The death cases are

Figure 3.1: One of the early examples of information visualization (from Tufte [Tuf83])

presented as dots in the picture; from their concentration along *Broad Streed* it was found that the people living in this street had a common water pump which was the cause of the disease. Thus, although the term information visualization has been coined only fifteen years ago, its history cannot be restricted to the last fifteen years. Computer-based information visualization is the intersection point of several other fields, which appeared to be independent and specific, such as cognitive psychology, graphic representation, cartography, visual languages, semiotics, and data analysis. The common denominator of all these fields is their final purpose: Knowledge Extraction.

In the context of computer science, information visualization has the biggest intersection with computer graphics, human-computer interaction and data mining, including also basic subfields such as algorithms, data structures and alike. Cognitive and perceptual psychology offer important scientific guidance on how humans perceive information in general and visual information in particular. The focus of scientific visualization is on the visual display of spatial data associated with scientific processes, e.g. simulations. Information visualization examines developing visual metaphors for abstract data such as social networks or high dimensional data sets without a trivial visualization.

Visualization of this kind of data is difficult because no natural display is possible in contrast to e.g. volume visualization, a subfield of scientific visualization, where the objects to be visualized are approached visually via isosurfaces, representing constant data values. In scientific visualization, physical methods are often used to help make the data visible like in flow visualiza-

tion, where dye or smoke is injected, enabling an arrow based visualization of the flow. Furthermore, scientific visualization presumes that we have certain features to look for, whereas in information visualization the character of the relationships to be found is not at all obvious a priori. Thus, information visualization methods must be able to deal with data that appears to be random, but still contains valuable information.

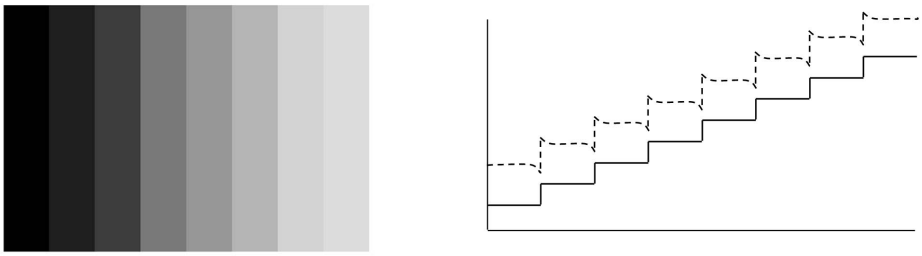
However, these two fields share with each other the idea of transforming quantitative data into meaningful sensory information for the purpose of analyzing the data by taking advantage of the human brain's ability to process a vast amount of visual information very rapidly.

3.3 Visual Perception

Human visual perception is very important in the context of information visualization. Although haptic or sonification based approaches are used to enable visualization for visually impaired people [FB99, SBG90], normal visual perception still remains the most important method of conveying visual information to the end user. In the context of this thesis, we will consider only perception issues regarding human vision since the implemented approaches function only in this context.

3.3.1 Structure of the Human Eye

The eye has an average diameter of 2.5 cm and is covered by several membranes: *cornea* and *sclera* as outer covers and *choroid* and *retina* as inner ones. The cornea consists of tough, transparent tissue and covers the anterior part of the eye and sclera is continuous with the cornea. The choroid serves as source of nutrition for the eye and lies directly below the sclera. In the anterior part, it is divided into the *ciliary body* and the *iris diaphragm*, which controls the amount of light that enters the eye. The *pupil* is the central opening of the iris. The *lens* of the eye serves as a flexible optical lens and absorbs a part of the visible light spectrum as well as infrared and ultraviolet light. In a properly focused eye, the light coming from an object outside the eye is imaged on the *retina*, where millions of receptors divided into two large groups, *cons* and *rods*, are distributed. The number of cones is between 6 and 7 million and they are located in the central retina, called *fovea* and are highly sensitive to color. Each of these cones is connected to its own nerve, making the cones important in resolving fine details. The number of rods is much larger, 75 to 150 million, and several of them are connected with a single nerve, which makes them appropriate for generating an overall picture of what is seen. The rods are not involved in the color vision. The fovea has



(a) Mach bands

(b) Illumination (continued line) and perceived brightness (dashed line)

Figure 3.2: Mach bands illustrating the differently perceived brightness of adjacent bands with gradient luminance

a diameter of about 1.5 mm and a density of cones of about 150,000 elements per mm^2, meaning that the region with the highest sensibility in the eye has about 337,000 elements. Thus, the ability of the eye to resolve detail is within the capacity of current electronic imaging sensors.

3.3.2 Image Perception

The flexibility of the eye lens controls the focus of the eye; when it is thicker, the focus is nearer, whereas when the lens becomes thinner, the focus is on distant objects. The contact with light in the retina excites the receptors which generate electrical impulses that are further processed in the brain. These nerves do not transmit any information about the amount of light falling in the retina; it is the relative amount of light, both in context of how the light in a certain part changes and how the light in a certain part differs from the neighborhood that is signaled. This fact is very important in the context of visualizations because it can cause errors in the way data is read from visualizations. Furthermore, this fact implies that light perception is nonlinear [War00].

Figure 3.2 shows the Chevreul illusion (sometimes known also under the name Mach bands). Bands with different grayscale intensities are put adjacent to each other. Although the light intensity has a staircase pattern, the perceived intensity (shown in the dashed line in Figure 3.2(b)) shows a slightly different pattern, which differs from the actual intensity especially at the borders. Figure 3.3 shows similarly that the perceived intensity of the small middle rectangle does not depend only on its intensity, which is the

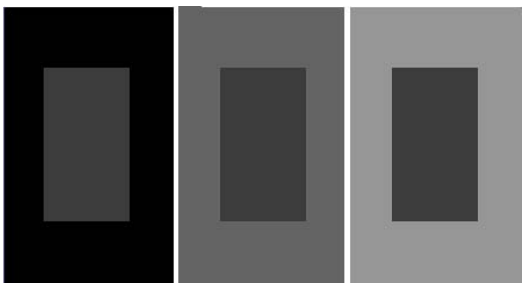


Figure 3.3: Simultaneous contrast illusion. The middle rectangles have the same intensity, but are perceived differently because of the intensity of the surrounding rectangle.

same in all three cases, but also on the surrounding background.

From the visualization point of view, these examples are important because situations where the real meaning could be misperceived due to the way the human visual system is built, should be avoided.

3.3.3 Use of Color in Visualization

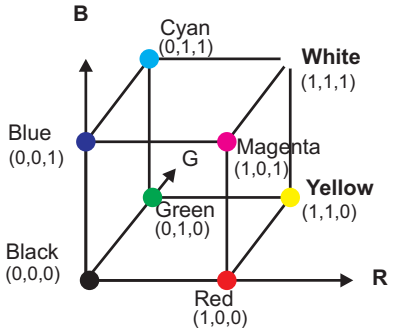
Color vision has no central role in everyday life; it does not help us to distinguish the shape of objects or track their movements. However, color usage is extremely useful in visualization.

Humans can discern thousands of color shades and intensities, compared to about only two dozen shades of gray, which makes them very interesting in the context of visualization compared to simple gray scale values.

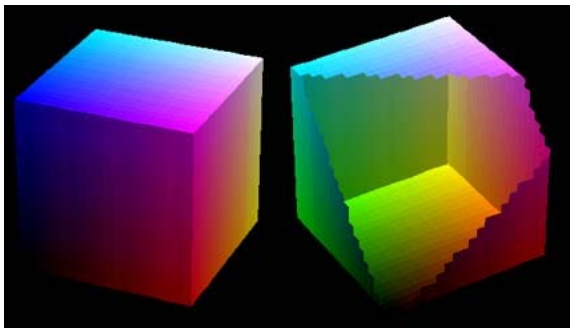
In the following paragraphs, two color models, namely the RGB and the CIE color models will be briefly explained. The described color spaces are three-dimensional, which is derived from the fact that the rods, described in Subsection 3.3.1, have three distinct color receptors.

3.3.3.1 RGB Color Model

In the RGB color model [GW01], each color appears in its primary spectral components of red, green, and blue. This model is based on a Cartesian coordinate system and is often visualized by a unit cube where each color (red-R, green-G, blue-B) is assigned to one of the three orthogonal coordinate axes in the 3D space, as illustrated in Figure 3.4(a) and in Figure 3.4(b). The values of the R, G, and B components move from 0 to 255.



(a) Schematic view of the RGB color model



(b) The colored presentation of the RGB model

Figure 3.4: Two different visualizations of the RGB color model

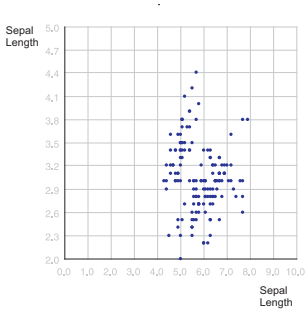
3.3.3.2 Color in the Context of Visualization

In exploratory data analysis, one way of approaching the data is to visualize it and search for relationships without any prior assumptions. For low dimensional data sets (1D, 2D or even 3D), commonly used approaches are to plot the data and then try to interpret the visualizations in the context of the problem. Thus, the perception of the visualization is of utmost importance for the right interpretation of data. However, things become difficult when the data has more than three dimensions. Color can help to extend the space of dimensions further by attaching the three basic colors, red, green and blue to another three dimensions.

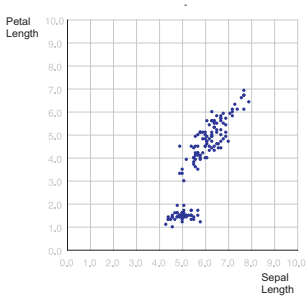
To illustrate this, let us take an example data set and show how its visualization could possibly enhanced by using color. Figure 3.5 shows six scatter plots, representing the projection of the data set into every pair of dimensions of the four dimensional Iris data set [SHM]. These six plots must be examined together in order to make hypotheses about possible groups/clusters in this data set. Figure 3.6, again represents plots of this data set into the respective pair of dimensions, but with the additional property that the two remaining features are used to determine the color of the respective point. Thus, in Figure 3.6(a), for example, the first and second dimension are used to create the two dimensional plot, the third dimension defines the quantity of red in the respective color and the fourth dimension defines the quantity of blue. Again, these six plots should be examined together, however, a single plot in Figure 3.6 with respect to a single plot of Figure 3.5 gives much more information about possible clusters in the data set, since similar points are near to each other and have similar colors.

For visualizing data sets with more than four dimensions (but definitely with equal or less than six dimensions), all three components of the RGB

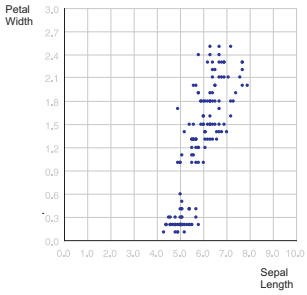
spectrum can be used. However, interpreting the data in such an approach can be difficult, because items could appear with similar color, e.g. green because they lack the red component partially and not because they have a strong green component.



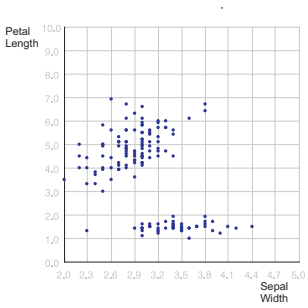
(a) 1st and 2nd feature



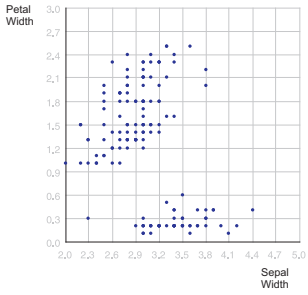
(b) 1st and 3rd feature



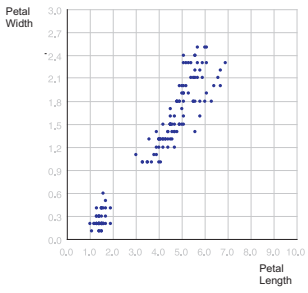
(c) 1st and 4th feature



(d) 2nd and 3rd feature



(e) 2nd and 4th feature

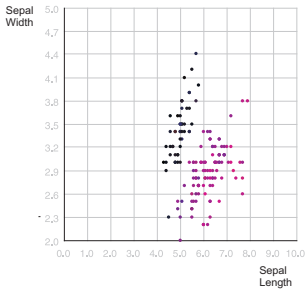


(f) 3rd and 4th feature

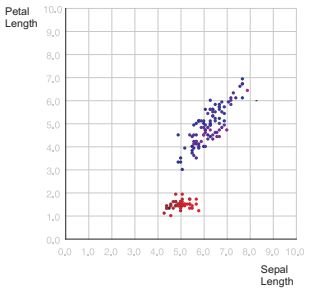
Figure 3.5: Projection plots of Iris data set

When using color in visualizations, several rules should be kept in mind:

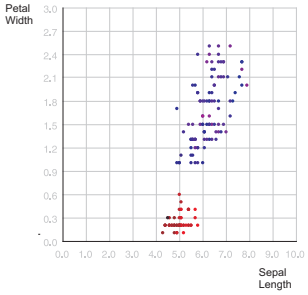
- **Good discrimination of small color differences** is possible when the colored areas are large, adjacent to each other and are viewed simul-



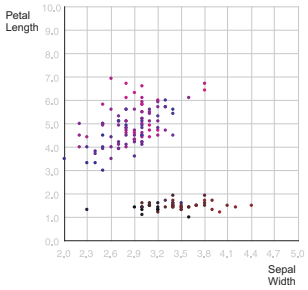
(a) 1st and 2nd feature



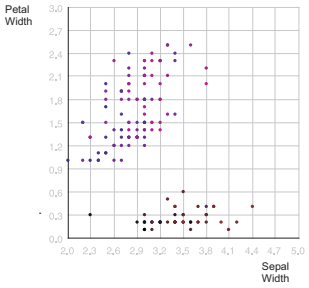
(b) 1st and 3rd feature



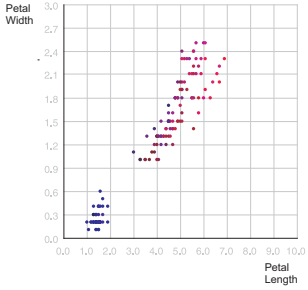
(c) 1st and 4th feature



(d) 2nd and 3rd feature



(e) 2nd and 4th feature



(f) 3rd and 4th feature

Figure 3.6: Colored projection plots of Iris data set

taneously. This is often the case in the visualization of different kind of weather, temperature or terrain maps. Deviance from these conditions should be dealt with larger color differences.

- **Size** is very important for proper color distinction. The smaller the size, the more difficult is the discrimination between colors.

- **Individual differences in color discrimination** are important in the context of visualization users with anomalous color vision. In these

cases, smaller color sets, which can be better discriminated by such people, should be used.

- **Usage of sharp edges** could also possibly help to better discriminate colors in contrast to the usage of smooth gradients. Smooth gradients, on the other side, are used to show, for example, elevations in terrain maps.

- **Conformity with cultural conventions**. Some colors have become associated with particular meanings in certain contexts and other uses could lead to false interpretations of the visualization, as for example, red, yellow, and green which are associated with the safety status in safety related visualization problems.

- **Consistency of usage**. Color should be used consistently in the same context so that the same meanings are associated with the same color, otherwise different interpretations in different contexts increase the cognitive effort and open opportunities for error.

- **Usage of neutral colors** e.g. gray backgrounds in cases where color interpretation is critical as otherwise similar colors could interfere with the process of interpretation of visualization.

3.3.4 Pre-Attentive Processing

Certain features of the visual image can be identified easily after looking briefly the image. This step, which logically occurs before the attention of the user is concentrated on the visual image, is called pre-attentive processing. One such example, generated in analogy to an example given by Ware [War00], is presented in the following. The same block of numbers is presented two times, once without any highlighting and in the second case, the digits **3** are bold whereas the rest is gray. It is clear that counting 3s in the second example is much easier than in the first.

624695333025642841625330660880934556660262497750054717370870762588592915154499211858504120907076297400814240307627367608821441631497293941734788463575751582571421496220915961443049231145104404678036740516100872620561347835181161464871150283061898420 4

624695**333**025642841625**33**066088093455666026249775005
4717**3**708707625885929151544992118585041209070762974
008142403076273676088214416**3**14972939417**3**4788463575
751582571421496220915961443049231145104404678036 74
0516100872620561**3**478**3**518116146487115028**3**0618984204

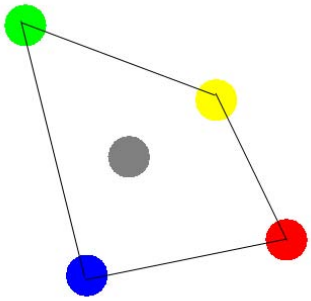
In the context of visualization, pre-attentive processing is important because good visualizations could be enhanced to make use of factors that can be processed pre-attentively. Thus, if one is interested that the user identifies instantly something on a visual display, it could be made distinct from the rest by using pre-attentively processed features. Typically, experiments are conducted in order to find out if certain features are processed pre-attentively or not. If the time taken to distinguish the target is independent of the distraction level, then the subject feature is processed pre-attentively. Table 3.1 (taken from [War00]) presents features of visual images that are processed pre-attentively.

Table 3.1: Pre-attentively processed features

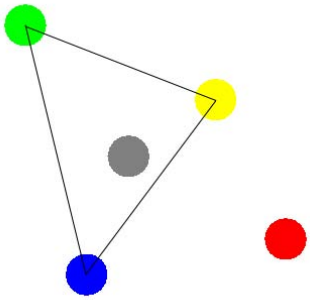
Form	**Color**	**Motion**	**Spatial Position**
Line Orientation	Hue	Flicker	2D Position
Line Length, Width	Intensity	Motion Direction	Stereoscopic Depth
Line Colinearity			Convexity/Concavity
Size			
Curvature			
Spatial Grouping			
Added Marks			

Table 3.1 also shows that color is one of the pre-attentively processed features, which is a well established fact. However, in order for a color to be distinguished pre-attentively, it should lie outside the region defined by the existing colors in a visual display in the CIELAB color space. The CIELAB color space, although not a directly displayable format, has several advantages such as its colorimetric property where similarly perceived colors are also encoded similarly, perceptually uniform i.e. color differences are perceived uniformly and it is device independent [GW01]. Thus, the *convex hull* defined by certain colors in the CIELAB space defines colors that have a certain similarity with all the predefined colors. Colors that are pre-attentively distinct from the given colors lie outside of this convex hull.

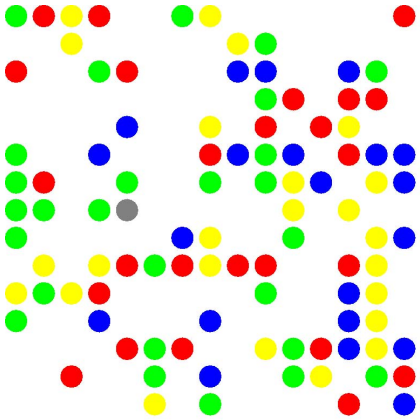
Figure 3.7 (adapted from [War00]) illustrates this idea. In Figure 3.7(a), the gray color lies within the convex hull defined by the colors blue, red,



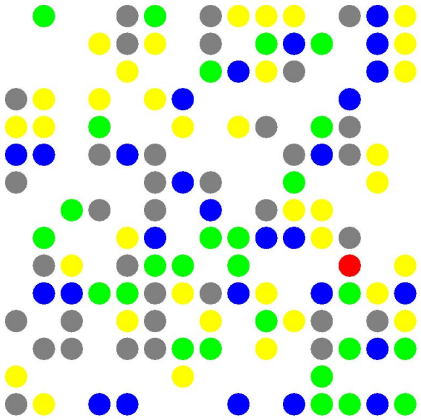
(a) Convex hull of red, green, yellow and blue with gray inside the hull



(b) Convex hull of green, yellow and blue with red outside the hull



(c) Gray is not processed pre-attentively



(d) Red is processed pre-attentively

Figure 3.7: Pre-attentive processing of colors (adapted from [War00]).

yellow and green in the CIELAB color space. For this reason, it is difficult to find the gray circle in Figure 3.7(c), whereas in Figure 3.7(b) the red color lies outside the convex hull defined by the colors blue, yellow and green in the CIELAB color space, making it easier to distinguish the red circle in Figure 3.7(d).

3.3.5 Types of Data

Data is at the source of every visualization method. However, its attributes vary from one data set to the other. Below, two taxonomies related to data attributes and data types are discussed.

3.3.5.1 Attributes of Data

One of the broadly accepted taxonomies for the classification of data scales is the one defined by Stevens [Ste46]. According to this classification, there are four categories for measuring data scales: nominal, ordinal, interval and ratio.

- **Nominal** scale is a collection of identifiers, which has the peculiarity of possessing no specific order, e.g. hair color of people. Every transformation preserves the relationships between variables of this scale.

- **Ordinal** is similar to the nominal scale, but in contrast to the nominal scale, it encompasses values that have an ordering defined in them. Examples include the income of people, which can be classified as low, medium and high, or the height of a person defined as short, medium and tall, etc.

- **Interval** scale is a further extension of the nominal scale, where the intervals between possible values of the variables are equally spaced. For example, consider three people with heights 160, 175 and 190 cm. With nominal variables, they would be short, medium and tall. In the case of nominal variables, no presumption is possible about the differences between the three persons, whereas in the case of interval scale we know that the difference between the third and second person is the same as the difference between the second and first person.

- **Ratio** scale is an extension of the interval scale, where an absolute zero is defined, as in the case of money, size etc.

A more practical division is based on three groups: Category or enumeration data like nominal data above, integer data similar to the ordinal scale and real-valued data.

3.3.5.2 Data Type Taxonomy

Shneiderman [Shn96] has defined a taxonomy of seven data types in the context of visualization:

1. **1-dimensional** or linear data types include data which could be organized in a sequential manner such as lists of strings, source code of programs, texts and so on. Each item belonging to collections of this data type could possibly have attributes that characterize it.[1]

[1]It could be argued that text data, e.g. source code is structured and might thus be classified as hierarchical or network data

2. **2-dimensional** or planar/map data includes geographic maps, plans and alike where each item in the collection covers some part of a 2D plan. This kind of data type is extensively used in geographic information systems.

3. **3-dimensional** data represents real-world objects such as molecules, the human body and buildings, which have items with volume and some potentially complex relationship with other items.

4. **Temporal** data represents time lines, which are widely used and crucial in the context of medical records or project management. This data type bears similarities to 1-dimensional data, with the distinction that we have a start and a finish time defined, and possible overlaps could occur.

5. **Multi-dimensional** data represents relational and statistical databases as tuples with n attributes (considered also as points in a n-dimensional space).

6. **Tree** data hierarchies or tree structures are collections of items with each item having a link to one parent item (except the root). Items and the links between parent and child can have multiple attributes.

7. **Network** or graph data represents data whose structure/relationships cannot be captured with hierarchical data types. In this case, links exist arbitrarily and not only between child and parent nodes.

3.3.6 Visual Variables

In his seminal work *Semiology of Graphics* [Ber83], Bertin developed a theory of what he calls visual variables. According to him, the graphic system has several visual variables: position, form, orientation, color, texture, value, and size. Since position is two-dimensional, there are eight variables to work with. Bertin also used the term retinal variables, because they can be compared effortlessly without additional cognitive processing, as if the retina were doing all the work. Figure 3.8 visually illustrates these mentioned visual variables. In modern visualization, motion is also considered a visual variable. Furthermore, the set of visual variables is a subset of the pre-attentively processed features which were listed in Table 3.1. In the context of human processing, the differences expressed by the above mentioned visual variables are distinguished perceptually without involving other processing steps, as in the case of comparing numbers.

Figure 3.8 is self-explanative, except perhaps for color and value. Value represents the luminance or the grayscale value and it is different from color.

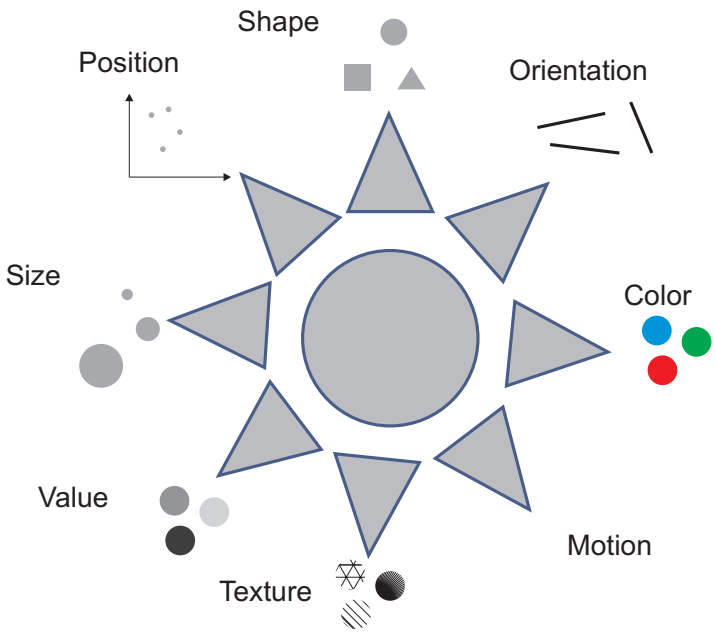


Figure 3.8: Illustration of visual variables

Although it might be argued that both represent subsets of RGB color space, value is thought to be suitable for ordinal types, whereas color is thought to be suitable for nominal/categorical data types. Thus, visual variables serve as a means of communication by encoding data in such a way as to draw distinctions between visual elements. However, depending on the attributes of data to be visualized (see Subsection 3.3.5), some of these visual variables have more representational power than others. Thus, when visualizing a file system, shape is not as efficient as color to make some files distinguishable from the others. For certain types of data, e.g. temperature, any one of these visual variables could be used: position on a scale, length of a bar, color of an indicator, or shape of an icon. The choice of a visual variable will strongly affect how the users will be able to perceive and use the displayed data.

In general, Bertin divides the perception characteristics of visual variables into four groups:

- **Associative perception** describes how much a visual variable affects the visibility of other dimensions. The size or color of an object is unaffected by its orientation, making orientation associative whereas for very small objects we can hardly distinguish their orientation or color, making the size a dissociative variable. Size and value dominate perception and are both dissociative variables.

- **Selective perception** defines how good a certain visual variable isolates particular instances from the rest. A visual variable is selective if

this process is immediate and effortless. An example is the selection of all green objects in a visual display. All visual variables are selective with the sole exception of shape. Thus, the process of picking out a triangle amidst a set of rectangles does not occur effortlessly.

- **Ordered perception** of a visual variable means that the objects should be able to be put into a ranked order based on their values. An ordered variable does not need to consult an index to determine the ranking of the objects. Position, size and value are ordered in human perception.

- **Quantitative perception** restricts the criteria of ordered perception further; it must be possible for a viewer to distinguish between two ordered values and to be able to determine the amount of difference. Position and size are quantitative in human perception.

Table 3.2 shows the seven visual variables and summarizes their characteristics, whereas Table 3.3 shows their appropriateness for being used to visualize different data types.

Table 3.2: Characteristics of visual variables

	Selective	**Associative**	**Quantitative**	**Order**
Position	yes	yes	yes	yes
Size	yes	no	yes	yes
Shape	no	yes	no	no
Value	yes	no	no	yes
Color	yes	yes	no	no
Orientation	yes	yes	no	no
Texture	yes	yes	no	no
Motion	yes	yes	no	no

3.4 Algorithmic Aspects

Algorithmic problems in the context of information visualization can be divided into two main classes: (I) problems related to computational issues for enhancing visualizations, e.g. dealing with optimizing the space on the screen for a certain visualization issue; or (II) problems related to highly computational aspects of a visualization problem, e.g. computing a low dimensional representation of a high dimensional data set. Both classes of problems are

Table 3.3: Appropriateness of visual variables for different data types with 1 lowest and 3 highest

	Categorical/ Nominal	**Ordinal**	**Numeric**
Position	3	3	3
Size	1	3	3
Shape	3	2	1
Value	1	3	3
Color	3	2	1
Orientation	3	1	1
Texture	3	1	1

very important for good visualizations; however, they are distinct from each other because the first class of problems need not be very challenging from the computational side, whereas the second one is usually computationally intensive. Below, these two aspects will be illustrated.

Visualization of high dimensional data is a difficult problem. Depending on the data to be visualized, different paths are followed. One possibility are so called dimension reduction methods, which try to approximate the high-dimensional representation of data by a low dimensional one, usually two- and sometimes three-dimensional. *Multidimensional Scaling* (MDS) [YH38, Mar79] represents a common class of methods used for dimension reduction, which tries to approximate the inter-object distances in the high-dimensional space with inter-object distances in lower-dimensional spaces. To illustrate the idea, consider Table 3.4, representing the distances between some major US cities (1=Boston, 2=NY, 3=DC, 4=Miami, 5=Chicago, 6=Seattle, 7=SF, 8=LA, 9=Denver), with the aim of reconstructing their map. Thus, we are looking for a two-dimensional configuration of the cities, which should be "close" to the true map of these cities. Different computational approaches are used to achieve this aim, and MDS includes a large group of them. Figure 3.9 represents one configuration of the cities generated using the classical MDS approach. Since MDS solutions are insensitive to rotation and translation, the solution approximates fairly well the real map. Thus, purely computational approaches as the one described above can be used to generate visual displays for exploring high dimensional data that is otherwise not visualizable using common approaches.

Let us consider another example where visual displays are not directly generated from computational frameworks but are improved significantly by their usage. To illustrate the situation, let us consider *Parallel Coordinates*, a popular multivariate visualization technique [ID90]. Each dimension in

Table 3.4: Distance between some major US cities

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	0	206	429	1504	963	2976	3095	2979	1949.0
2	206	0	233	1308	802	2815	2934	2786	1771.0
3	429	233	0	1075	671	2684	2799	2631	1616.0
4	1504	1308	1075	0	1329	3273	3053	2687	2037.0
5	963	802	671	1329	0	2013	2142	2054	996.0
6	2976	2815	2684	3273	2013	0	808	1131	1307.0
7	3095	2934	2799	3053	2142	808	0	379	1235.0
8	2979	2786	2631	2687	2054	1131	379	0	1059.0
9	1949	1771	1616	2037	996	1307	1235	1059	0.0

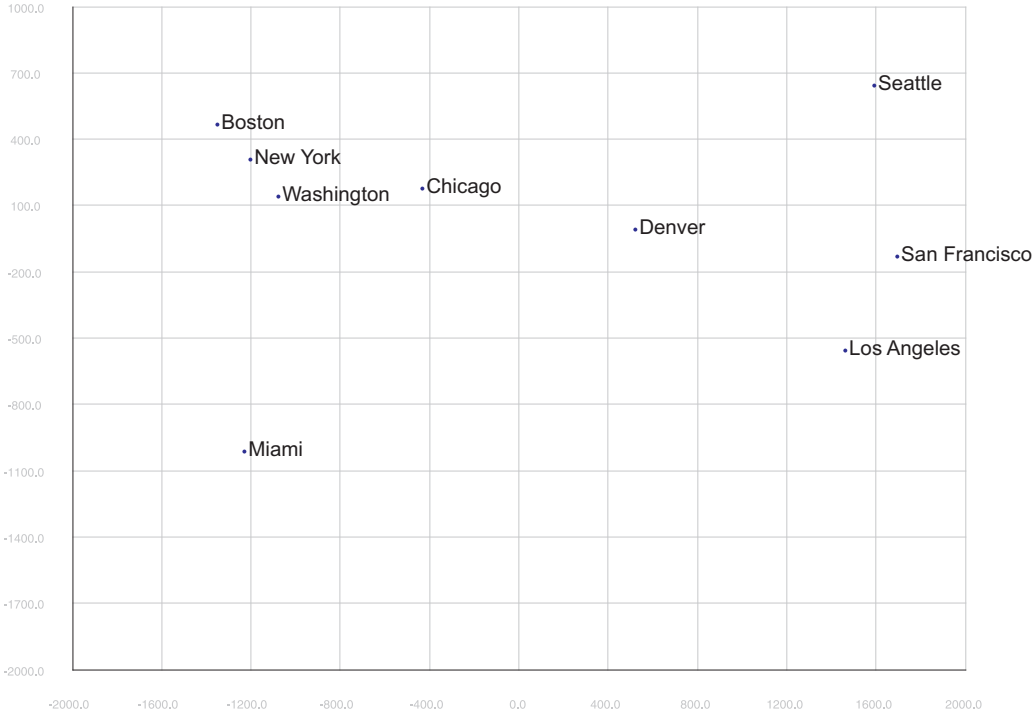


Figure 3.9: MDS representation of US cities. The MDS projection is calculated from the matrix of pairwise distances and is insensitive to translation and rotation.

this method corresponds to an axis; the axes by themselves are organized as uniformly spaced horizontal (sometimes vertical) lines. A point in the high dimensional space is represented as a line connecting points on each axis. Figure 3.10 shows the parallel coordinates visualization of the Cars

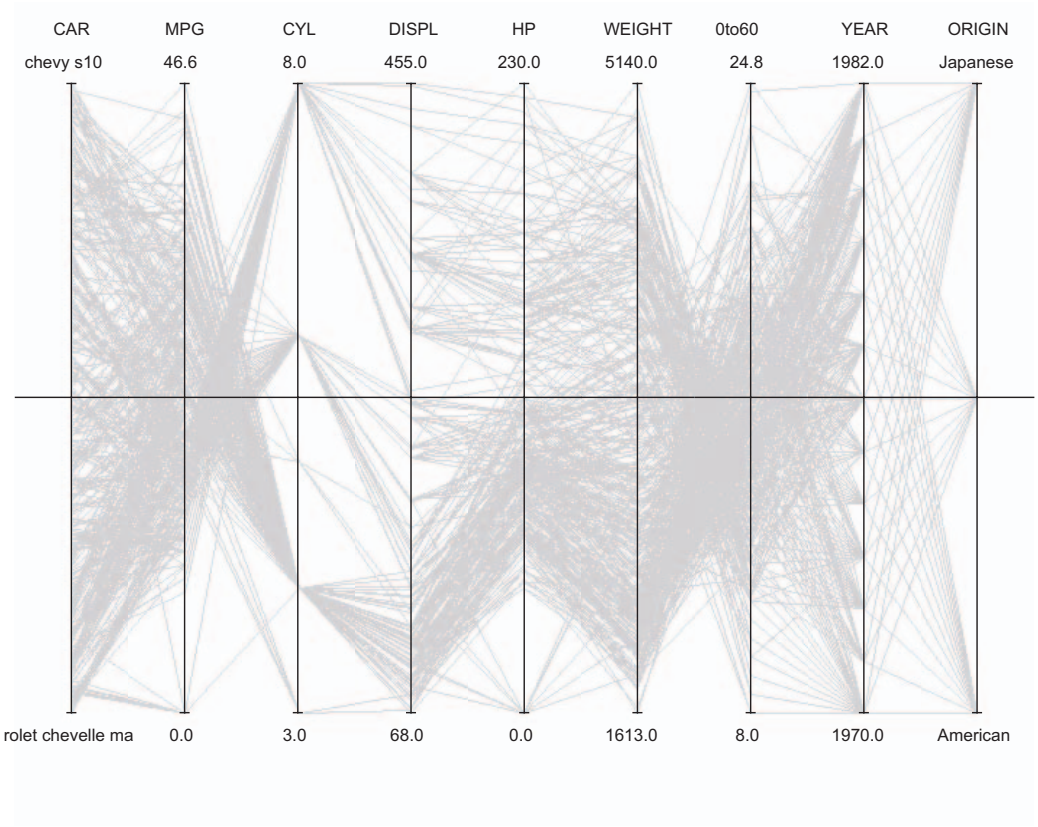


Figure 3.10: Parallel coordinates example. Each feature of the data set is mapped to one of the parallel axes and an object of the data set is represented by a line connecting the respective feature values.

data set (this data set contains information about 8 different features for 406 cars `http://lib.stat.cmu.edu/datasets/`). This method, as many other methods used in visualization of large data sets, suffers from the *clutter effect*. To reduce cluttering, different techniques such as distortion or sampling are used. However, in the case of parallel coordinates, algorithmic approaches could be used to enhance visualizations. Thus, a different order of dimensions could possibly reduce (or increase) clutter in Figure 3.10 [PWR04]. This occurs due to the fact that different axes orders create different shapes in the parallel coordinates display. The biggest difficulties occur in disclosing relationships of non-adjacent dimensions. A proper order brings more clarity, as Figure 3.11 illustrates, e.g. outliers with respect to cylinder number (third column on both displays) are easier to extract.

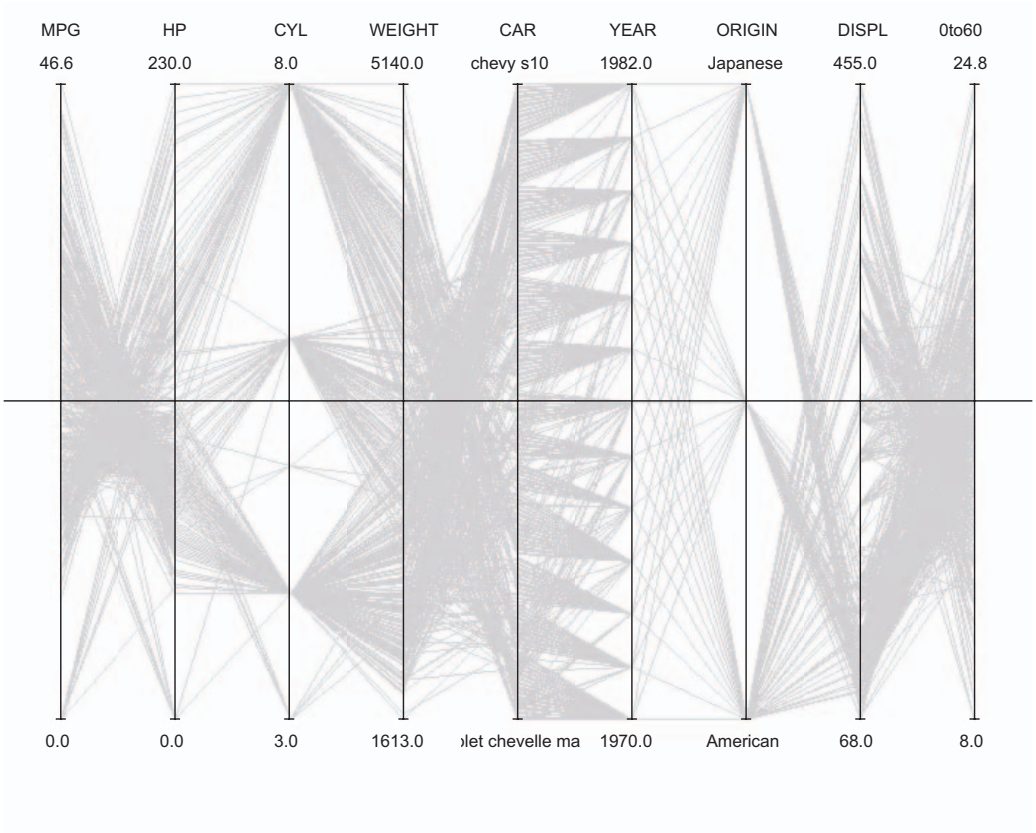


Figure 3.11: Parallel coordinates ordered. The configuration of visualization is affected by changes in the order of axes.

3.5 Visualization Stages

The visualization process consists of the following steps:

1. Collection of data to be visualized;
2. Visual display of data;
3. Human perception of the visualization.

The second step, depending on the context, could be further elaborated into preprocessing of data and visual rendering of the preprocessed data. Figure 3.12 (adapted from [War00]) illustrates the visualization process, with the above described steps.

To explain the figure, let us consider the two cases below. In the simplest case, the data set to be visualized possesses a relatively clear visual representation, such as data gathered for a certain time period for the price of a single stock. In this way, no special preprocessing is needed; the graphical engine plots the trajectory of the stock based on the data, and the human user

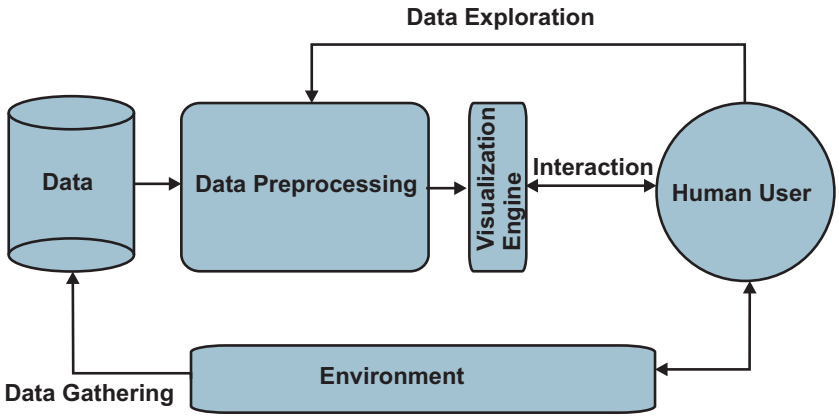


Figure 3.12: The visualization process (adapted from [War00]).

interprets the evolution of the stock based on its visualization. In a slightly more complicated case, we could consider a set of stocks in real time, and in addition to their trajectory plots, we would like to visualize any possible relationships/correlations between stock evolutions. In this case, the data gathering process is continuous, preprocessing is needed in order to calculate the correlations between stocks and they need to be properly visualized. The human user could then interact with the visualization for obtaining information about related stocks, filter a subset of the considered stock set and so on (data manipulation/data exploration steps in Figure 3.12).

Thus, the interaction between the user and the visual environment is very important in the context of information visualization. For this reason, the following section is dedicated to the explanation of the basic issues.

3.6 Interaction Issues

In some way, visualization could be considered as an interface between the human user and the data to be visualized.

In the early days of paper based visualization, interaction with visualization was nearly non-existent for technical reasons. Complicated visualizations needed to be created from scratch when changes in the data occurred and this was the only interaction loop between the user and the visualization, if it could be considered as such. An interesting example of interaction in early visualization techniques provides the interaction with the reorderable matrix method of Bertin [Ber81]. The reorderable matrix represents numerical data arranged in a matrix form by symbols with changing sizes or values. By rearranging columns or rows of the matrix, the user can thus discover interesting patterns in the data. The original version of the reorderable matrix was a paper based one. To interact with it, the reorderable matrix was cut along

the rows or columns, and the paper stripes representing them were permuted to explore the patterns obtained. In this way, a certain degree of interaction was provided for the reorderable matrix.

Computer based visualization should offer appropriate interaction abilities. Interaction techniques range from simple methods such as *linking* and *brushing* to sophisticated distortion techniques. However, leaving the differences aside, these techniques are united in a common goal: helping the user to better understand the data by enhancing the visualization.

Shneiderman [Shn96] defined what he called the visual information seeking mantra: *Overview first, zoom and filter, then details-on-demand*, which represent the most important aspects of the taxonomy he defined for visual frameworks in the context of information visualization and are tightly coupled with the data types defined in Section 3.3.5.2:

- **The overview** task for enabling the user to gain an overview of the entire collection.
- **The zoom** task to focus on items of interest.
- **The filter** task for selecting out uninteresting items.
- **The details-on-demand** task, which allows selecting an item or a group and get details when needed.
- **The relate** task for viewing relationships among items.
- **The history** task for keeping a history of actions to support undo, replay, and progressive refinement.
- **The extract** task for allowing the extraction of sub-collections and of the query parameters.

3.6.1 Interaction Techniques for Large Visualizations

Information visualization has to deal with a serious restriction: available screen space. Whereas slow algorithms perform better in faster computers or algorithms that require too much memory become feasible by extending the memory of the computer, the screen resolution has remained nearly constant in the last years. For this reason, proper interaction techniques need to be provided which can cope with this restriction. These techniques should furthermore simplify the user's tasks, for example, by providing means for finding unknown patterns in the visualization.

Interaction techniques are divided by Eick [GCE98] into three categories, namely focusing, filtering, and linking.

3.6.1.1 Focusing Techniques

Focusing techniques work by promoting certain area(s) of the visualization to the focus of the user, whereas the rest of the visualization either is not shown at all or it is visualized separately or in an integrated view in the visualization, as described below.

Inside focusing techniques, three large groups are distinguished which will be discussed in the following paragraphs.

Zooming and Panning. The basic approach to deal with large visualizations is to use zooming and panning. This approach is often used in practice; however, the user sequentially explores the information provided by the visualization. Enhancements to this approach include the ability to bookmark positions already visited in order to navigate rapidly between different positions. In general, the implementation of the ability to zoom and pan is often application specific. However, there are some approaches to provide unifying zoom and pan ability for visual displays [BM98, BMG00].

Overview+Detail. Another approach for dealing with insufficient screen size is to employ several visualization techniques, which display different views of the same document. This approach seems a little bit counterintuitive, since a resource which is already scarce, namely the screen space, is restricted. But the different views are specialized and include at least one overview display, which represents the entire visualization and a detail view, which represents the details the user needs. The two views are connected to each other by using markers that show the position of the detail view in the overview. Overview+Detail visualization techniques are ubiquitous in visualization, especially in maps, medical visualization, etc. The detail view is often a zoomed and enhanced visualization of a certain part of the overview window. Shneiderman [Shn98] has suggested that if the zooming factor exceeds a certain upper value, the two window approach should be extended to a cascade of more than two windows.

Focus+Context. Focus+Context techniques develop the ideas presented in the previous paragraph further in that they offer both a view of the whole data and a view for the detailed analysis of parts of visualization. However, the difference between the two consists of the fact that focus+context techniques aim to integrate both the details and the overview in a single visualization. The motivation for such an approach comes from the fact that when information is divided into two displays, the performance of the user degrades [Ber81]. A large group of focus+contest techniques consists of distortion-based visualization techniques. The introduced distortion in these techniques interferes with tasks requiring judgments about scale, distance,

direction, or alignment. However, it is this distortion that makes it possible to integrate both overview and details in one view.

Fisheye views are a well known distortion technique and were first introduced by Furnas [Fur86]. Originally, they were conceived as an interaction framework for filtering information based on the current point of interest of the user and found application in the browsing of structured data such as source code of programs. Nowadays, fisheye views are commonly used to introduce distortion in visualizations where the available screen size is scarce. The general metaphor used in this context is the effect observed when looking through fisheye lenses or magnifying glasses. Furthermore, instead of one focus viewers, bifocal and polyfocal viewers [LA94] are used when detailed information of more than one object is needed.

Another group of focus+context techniques makes use of non-Euclidean hyperbolic geometry, which offers more space for visualization. Although hyperbolic space is an infinite space more voluminous than Euclidean space, it can be projected into a finite volume of Euclidean space, achieving a fisheye-alike effect where the objects in the origin of hyperbolic space are displayed in the center of the normal space. Lamping et al. [LRP95] applied projection from hyperbolic space to Euclidean space to visualize large tree hierarchies, whereas Munzner [Mun97] extended the idea further to three dimensions.

Depending on the context and the tasks to be solved by the user, the above mentioned techniques have their advantages and disadvantages. Usability studies which would be appropriate in different contexts do not exist. However, in the case of large graph visualizations [SZG+96] distortion techniques are considered superior to full-zooming techniques.

3.6.1.2 Filtering Techniques

Filtering techniques are based on the simple idea of reducing the available information on the screen in order to make the visual display more understandable. The filtering process proceeds by *browsing* or *querying* the visualization as follows:

- The user filters one or some objects and wants to see all objects that possess similar properties to the one selected.
- The user selects some objects and wants to omit the selected objects (possibly omitting the objects that possess similar properties).

The results of the filtering process can be either highlighted (where the removed items are only dimmed), or the removed items are not shown at all in the visualization. Furthermore, this process can be accomplished both interactively or based on a query processing mechanism. For example, numerical data objects can be filtered based on specific ranges which can be

given either manually or selected by interacting with the visualization. For character based data, as in the case of search engines, manual interaction with a query engine is more effective.

The division between filtering techniques and focus-based methods is fuzzy; both aim to convey more relevant information to the user. However, focus techniques usually affect the visualization space, whereas filtering techniques focus on the attributes of the objects being visualized.

3.6.1.3 Linking and Brushing Techniques

There are many possibilities to visualize a data set based on its attributes, which have their advantages and their drawbacks depending on the user tasks to be performed. The idea of linking and brushing is to combine different visualization methods to overcome the shortcomings of single techniques. The different visual views need not necessarily be different visualization techniques as in the case of matrices of scatter plots (see Figure 3.6), where every pair wise projection is provided and all of them are linked with each other.

Thus, the consequences of an action carried out on one particular view of a visualization framework will be reflected in other views as well. The action could be brushing (highlighting temporarily an object or a set of objects), filtering and so on.

The decision whether to provide a set of linked views or an integrated visualization environment has been the subject of various research efforts [Rob98]. Some of the issues that are important in this context are:

- **Orthogonality/Complementarity.** The different views should complement each other with respect to the insight they give to the data, i.e. they should be "orthogonal" to each other.

- **User attention.** The drawbacks related to the attention problems since the user will be focused on several views simultaneously need to be properly considered in the case of multiple views.

- **Appropriateness.** Multiple views for different data sets and the related user tasks should be appropriately used to represent the data.

3.7 Summary

In this chapter, an overview of the field of information visualization was given. Different issues ranging from properties of the visual perception system to the interaction issues related to visualization were treated. In this way, the basic background needed for the rest of the thesis related to information

visualization was provided. More details about specific issues will be given in the respective chapters.

4
Visualization of Metabolic Networks

4.1 Introduction

Until recently, graphical representations of metabolic networks were prepared manually. A good example for these manual visualizations is the Metabolic Pathways Poster [Mic93]. Figure 4.1 shows fragments of this poster, focused on a part of glycolysis which transforms Glyceraldehyde-3-phosphate (GAP) into Pyruvate (Pyr).

General purpose drawing programs are still used in practice in visualizing metabolic networks to create static visualizations of metabolic networks. However, such approaches have their restrictions with respect to the following aspects:

- **Reuse of the work.** An existing drawing is only partially reusable in another context. For example, parts of an already drawn metabolic network which are needed in another visualization should be transferable to a new drawing.

- **Modification of the work.** A current drawing cannot be used at all if another protocol is used for visualization. Thus, if one disagrees with the used convention for visualizing metabolites, reactions or effectors, the drawing needs to be modified manually to accomplish the changes.

- **Updates of the drawing** based on data generated from simulations or residing in databases. This is probably one of the major drawbacks of static visualizations. Two scenarios are distinguished here: in the first scenario, the visualization of a metabolic network could be animated to reflect its evolution during simulation (see Subsection 4.4).

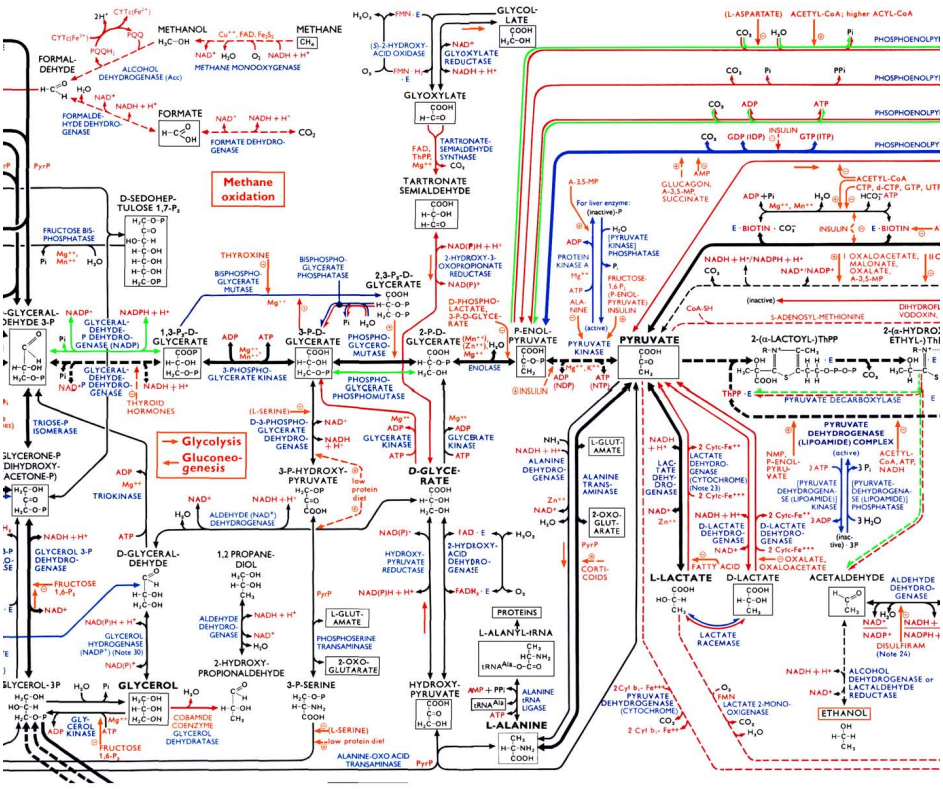


Figure 4.1: Parts of the metabolic pathways poster

In the second scenario, the visualized metabolic network could be updated/generated from data residing in databases or XML files.

In this chapter, different methods for visualizing metabolic networks are exploited. Parts of this work are published in several papers [QWF03, QWF04a, ONW+06, NWH+06].

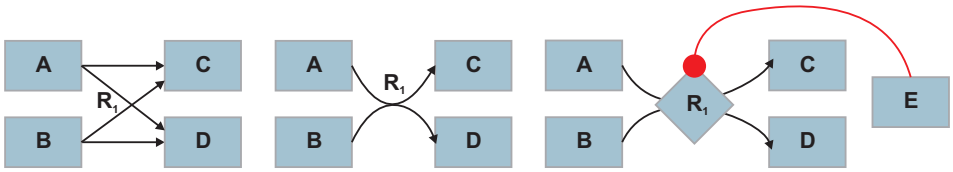
This chapter is organized as follows: Section 4.2 deals with modeling approaches for metabolic networks from the graph theoretic point of view. Section 4.3 gives a survey of existing techniques for visualizing of metabolic networks. Then, the tool *MetVis* created in the framework of this dissertation is introduced in Section 4.4. Section 4.5 discusses the visualization of the metabolic networks in 3D. Graph drawing techniques in the context of designing and visualization of metabolic networks are discussed in Section 4.6. Section 4.7 summarizes this chapter.

4.2 Graph-Theoretical Modeling of Metabolic Networks

The basic structure underlying a metabolic network is a graph or one of its derivatives. A graph G is represented by the tuple $G = (V, E)$ consisting of a finite set of nodes V and a finite set of edges E. Each edge $e \in E$ connects two nodes $u, v \in V$. The graph could be undirected, meaning that an edge between u and v also implies the existence of an edge between v and u and vice versa, or directed where this condition does not hold. A hypergraph $G_H = (V, E_H)$ consists of finite sets of nodes V and a finite set of hyper-edges E_H. Each hyper-edge $e \in E_H$ connects a set of nodes $u_1, ..., u_n \in V$. A bipartite graph $G_B = (V_1, V_2, E_B)$ consists of finite sets of nodes V_1 and V_2 and a finite set of edges E_B. Each edge $e \in E_B$ connects exactly two nodes, one node from the set V_1 with one node from the set V_2. Formally, a metabolic network can be modeled as a normal graph, digraph or hypergraph. Wagner and Fell [WF01] create two types of normal undirected graphs: a) the substrate graph $G_S = (V_S, E_S)$ where vertices represent substances/metabolites and edges represent if two substances are part of the same reaction; and b) the reaction graph $G_R = (V_R, E_R)$ where vertices represent reactions and edges represent if the reactions produce/consume the same substance. These graphs are shown to be *small world networks*, meaning that the connectivity of these networks follows a power law distribution[1]. Jeong et al. [JTA+00] analyze metabolic networks of several organisms with respect to their topology and similarities. The network is modeled as a directed graph and its scale free structure is observed in all organisms studied. Furthermore, in case of E. coli $\gamma_{in} = \gamma_{out} = 2.2$ where γ_{in} controls the probability that a metabolite in the E. coli metabolic network participates as an educt in k reactions $k^{-\gamma_{in}}$ and γ_{out} is defined similarly for a metabolite participating as a product in k reactions. Small world networks are distinguished by the relatively short paths connecting nodes in the network, and this fact is measured through the diameter of the network indicating the average of shortest paths between nodes. This diameter remains constant through all the organisms studied by Jeong et al. [JTA+00]. Furthermore, the power-law connectivity implies that some metabolites serve as hubs inside the network. Random removal of these hubs, corresponding to possible mutation in genes, does not affect the average distance between nodes, and these hub metabolites remain the same across different organisms.

In the context of visualizing metabolic networks, modeling them as hypergraphs or digraphs seems to be more reasonable since modeling them as

[1]Such networks are called also scale-free networks and the connectivity distribution follows the law $P(k) \approx k^{-\gamma}$ in contrast to random networks where this distribution is exponential, namely $P(k) \approx e^{-k}$, where k represents the degree of nodes.



(a) Simple graph (b) Hypergraph (c) Digraph

Figure 4.2: Three possible graphs representing a small metabolic network. 4.2(a) is used to analyze properties of metabolic networks such as metabolite centrality and scale-free structure. 4.2(b) and 4.2(c) are often used interchangeably.

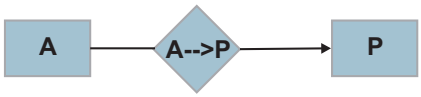
normal graphs would increase the number of crossings artificially. Hypergraphs constitute a natural way of modeling metabolic networks. They have, however, some drawbacks. First, they are not as easy to handle as normal graphs or digraphs. Second, they do not allow proper modeling of networks with effectors, where metabolites not participating in a reaction as products or educts have an inhibition or activation effect on the reaction.

Digraphs offer the same flexibility as normal graphs from the mathematical point of view. Furthermore, they allow the proper representation of effectors. Figure 4.2 illustrates the three discussed possibilities of modeling a metabolic network as a graph. Figure 4.2(a) creates four edges for one reaction, making this approach inappropriate for visualizing metabolic networks. Figure 4.2(b) represents the hypergraph approach; however it does not allow the representation of effectors as in Figure 4.2(c), where the network is modeled as a digraph.

The remainder of this work assumes that a metabolic network is modeled as a digraph, where two types of nodes exist, those representing metabolites and those representing reactions.

After having fixed an approach of modeling metabolic networks, an appropriate way of saving the data related to a metabolic network model is needed. The possibilities range from simple text files to intrinsic data structures saved in binary files. However, a good format should satisfy at least the following conditions:

- allow the easy exchange and reuse of models between users;
- be extensible to provide room for future developments;
- allow the integration of both structural i.e. graph theoretic information of a metabolic network and information related to the mechanistic model.



(a) A simple network

```
<?xml version="1.0" encoding="UTF-8" ?>
- <m3l>
  - <Model name="Simple Model">
    - <listOfModelParameters>
        <dilutionRate value="0" />
        <defaultValue flux_max="150" flux_min="10" />
        <dilutionRate value="0" />
        <resultSamplePoints begin="-0.1" end="1.0" interval="0.005" />
      </listOfModelParameters>
    - <listOfCompartments>
        <compartment name="Cytoplasma" />
      </listOfCompartments>
    - <listOfSpecies>
        <specie compartment="Cytoplasma" fixed="0" initialAmount="1" name="A" />
        <specie compartment="Cytoplasma" fixed="0" initialAmount="0.1" name="P" />
      </listOfSpecies>
    - <listOfKineticLaws>
      - <KineticLaw name="revMM">
          <description>revMM</description>
        - <                display="block" xmlns="http:/                2.org/199                thML">
        </KineticLa
      </listOfKineticLaws>
    - <listOfReactions>
      - <reaction name="A --> P" reversible="true">
        - <listOfReactants>
            <specieReference compartment="Cytoplasma" specie="A" />
          </listOfReactants>
        - <listOfProducts>
            <specieReference compartment="Cytoplasma" specie="P" />
          </listOfProducts>
        - <listOfKinetics>
          - <kineticLawReference kineticLaw="revMM">
              <parameter symbol="K_eq" value="1.6" />
              <parameter symbol="K_MS" value="0.1" />
              <parameter symbol="K_MP" value="0.05" />
              <parameter symbol="r_max" value="5.3" />
              <specieLink specie="A" symbol="cS" />
              <specieLink specie="P" symbol="cP" />
            </kineticLawReference>
          </listOfKinetics>
        </reaction>
      </listOfReactions>
      <listOfSplines />
    </Model>
</m3l>
```

(b) Fractions of its XML representation

Figure 4.3: M3L representation of a simple metabolic network

The Systems Biology Markup Language (SBML) [HFS+03], which is a description language for simulations in systems biology satisfies all these conditions. SBML is oriented towards representing biochemical networks and embodies the experience of several simulation tools in the field of metabolic modeling, thus being a perfect match for representing biochemical reaction networks in general and our models in particular. Furthermore, SBML is software-independent and XML-based, allowing great flexibility in its usage. Exploiting this flexibility, a SBML dialect has been created called M3L, which provides possibilities for distributed simulation of several model variants [HFWT02, HFTW05, Hau06].

Figure 4.3 shows a simple network with two metabolites and one reaction

and fragments of its M3L representation. Further details of SBML and M3L will be provided in Chapter 5.

4.3 Related Work

Previous work in the visualization of metabolic networks is focused on two tracks:

- *Drawing of metabolic networks/pathways*, which deals with different techniques for creating visual representations of metabolic networks based on an underlying model.
- *Visualization of metabolic networks with their associated information*, which is related to the above point, but in contrast to the former it focuses on issues related to the combined visualization of associated data, which are mainly time series, with the drawings produced in the previous step.

Below, a survey of the related work on both these fields will be given. Some technical details of the techniques will be further elaborated in Section 4.6.

4.3.1 Drawing Metabolic Networks

By representing metabolic networks graphically, researchers have the advantage of understanding the topology of these networks, which is tightly linked to their functionality. Thus, proper tools which make this topology understandable are needed, otherwise researchers will be confronted with just a list of reactions that are difficult to grasp. In general, the problem of visualizing metabolic networks can be solved in different ways. Depending on how they approach the problem of visualization, the methods can be divided into three large groups:

- Manual
- Semi-Automatic
- Automatic

Furthermore, the drawing techniques in these groups differ from each other on what kind of output they generate, and whether they are static (i.e. not modifiable) or dynamic (i.e. modifiable). Table 4.1 [2] summarizes these concepts based on the quality of the results they generate (with 3 being the highest and 1 the lowest).

[2]The estimations are based on personal experience with drawing of metabolic networks

Table 4.1: Three groups of drawing methods classified according to the quality of results

	Static Output	**Dynamic Output**
Manual Drawing	3	3
Semi-Automatic Drawing	1	3
Automatic Drawing	2	3

An example of a system in which metabolic pathways are manually drawn is KEGG (Kyoto Encyclopaedia of Genes and Genomes) [KG00]. The querying process is achieved by displaying interactive image maps which are linked to the related enzymes and pathways.

Similarly, the ExPASy Molecular Biology Server [ABH94] gives online access to the scanned version the Biochemical Pathways poster [Mic93], which is divided into rectangular pieces.

Brandenburg et al. [BJM97] have pointed out some of drawbacks of static visualizations in the context of metabolic networks, such as:

- manual update of changes;
- static behavior of the visualization (no reduction/increase of details);
- no approach to deal with novel pathways.

PathDB [Men00] offers an example of a semi-automatic approach to the drawing problem: metabolic networks stored in a database are analyzed for their topology; if they contain cycles, then the user decides if (s)he wants a hierarchical or a circular layout.

Several papers exist on dynamic graph layout algorithms for metabolic networks. They are motivated by the fact that although the aesthetics of general purpose drawing algorithms intersect with those for drawing metabolic networks, their priorities are different and new rules come into play (an additional discussion on this point follows in Section 4.6). The reasons for special purpose drawing algorithms in this context range from high connectivity of nodes (see Section 4.2) to specific cyclical structures. But the most important reason is that the end users of such visualization tools, i.e. biochemists, are used to see these networks visualized in a certain way and big deviations from their standard would definitely not help the main purpose of drawing algorithms, namely better understanding the topology of metabolic networks.

Karp et al. [KP94] developed the first divide-and-conquer technique tailored to metabolic networks, which is mainly focused on automatically drawing the networks stored in the database they built in the framework of BioCyc

project. The technique divides the networks into smaller parts with known topologies, which are then drawn using algorithms known to function well for those elementary topologies. The combination of the parts is done by using a hierarchical layout algorithm.

In BioPath [Fal02], a dynamic electronic version of the Biochemical Pathways poster [Mic93] presented earlier is realized. The visualizations it offers are generated by using a customized hierarchical layout algorithm, where nodes of different size and other restrictions approximate the established conventions regarding the drawing of biochemical networks.

The divide-and-conquer approach of Karp et al. was used also by Becker et al. in their drawing algorithm [BR01], but different from Karp et al. they use only two categories for distinguishing the topology, hierarchical and circular, and the combination of parts is done using a force-directed algorithm. The approach of Becker et al. was further extended by Wegner and Kummer [KU05]. Their approach [KU05] takes into consideration certain conventions used by biochemists such as node splitting and proper cycle identification.

Rojdestvenski [RC02, Roj03] computes 3D representations of metabolic networks in VRML (Virtual Reality Markup Language) by using a modified spring-embedding algorithm. However, it is difficult for a biologist to get accommodated with the visualization since the transformation from 2D to 3D is difficult to comprehend from a biological point of view, and the high connectivity fact is not considered at all, generating a lot of edge crossings.

4.3.2 Visualization of Metabolic Networks with Time Series Data

Computational and experimental approaches for determining the metabolome (and fluxome) offer new insights into the functionality of metabolic networks. These approaches, especially simulations of metabolic networks, lead to a great amount of data which needs to be properly interpreted in the context of the networks they are associated with. Thus, proper visualization techniques are needed to make this interpretation step easier. The common approach used to achieve this is by displaying data in tables, plots or histograms. In this way, the user can compare preselected metabolites or fluxes. Furthermore, these visualizations are quite often static snapshots, which have the same disadvantages mentioned in the context of drawing metabolic networks.

Despite the need for proper visualization tools for time series data related to metabolites and fluxes in the context of metabolic networks, there are only few approaches that treat this problem.

The solution proposed in this thesis, which will be elaborated in Section 4.4, is the first one considering this problem. Some other approaches are described below.

FluxAnalyzer [KSGG03] is a MATLAB package which offers several functionalities for analyzing the structure of metabolic networks and allows basic visualizations of simulation results through interactive flux maps. However, its functionality is restricted because the visualization proceeds by overlaying numbers as labels of the respective graph nodes, without making use of pre-attentive processing features.

Dwyer et al. [DRS04] visualize the experimental data directly in a metabolic network. This work is a continuation of [BDS03] where related metabolic networks are superimposed over each other to create a two and a half dimensional view of metabolic networks. Thus, this approach allows to overview at the same time several steps of a timely evolution in 3D. However, if the time series data representing experiments contains many time points, this approach suffers from the superimposition effect, where the visibility of layers in the end of the stack is reduced. Another drawback of the approach is that the time series data is stored in the same place where the structure of the network is stored, restricting the reuse of the same structure with different time series data.

Another tool which treats specifically this problem is SimWiz presented by Rost and Kummer [RK04]. This tool reflects the changes in concentrations of the metabolites (nodes) by changing their color or shape. The approach is extended to 3D by Wegner [Kat05] in a similar way to the approach proposed by Dwyer et al. [DRS04].

Finally, Sarayia et al. [SLN05] evaluate different approaches of visualizing graphs with associated time series data in a general context. They have studied several tasks related to overlaying of data for single time points or simultaneously for all time points, coming to the conclusion that the simultaneous overlaying is useful when judgments for all time points occur (e.g. outlier detection), whereas for local judgments the one time point approach performs better. However, they do not focus on different techniques for representing changes in time since different features, such as color, shape, size, etc. could be used to represent them.

4.4 MetVis: Metabolic Visualizer

Commonly, a great deal of attention is dedicated to building simulation tools, whereas the creation of tools to facilitate the evaluation of simulation results is somehow neglected. This, of course, does not affect the quality of simulation, but proper evaluation techniques such as visualization of time series, describing simulation results of metabolic networks makes the interpretation of the results much easier. A possibility to achieve this is by simultaneously visualizing the quantities describing the metabolic data (metabolites and fluxes) inside the metabolic network. We published the first such approach [QWF03],

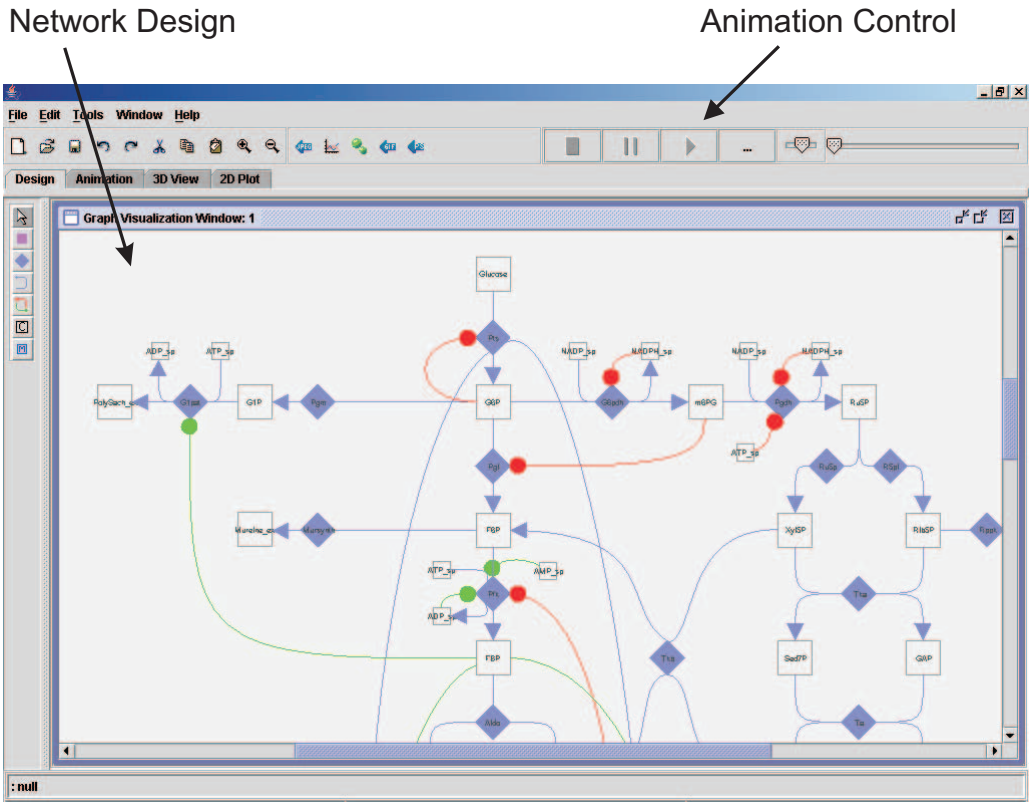


Figure 4.4: A screenshot of MetVis

called MetVis (Metabolic Visualizer), to animate metabolic networks visualizations based on data generated by simulation. This approach will be explained in detail in this section.

4.4.1 Basic Features of MetVis

When faced with the problem of simultaneously visualizing both simulation data and metabolic network structure, a good starting point is to enable the user to visualize metabolic networks in agreement with existing standards. In the framework of this dissertation, the tool MetVis was developed, which allows user friendly design of metabolic networks. Figure 4.4 shows a screenshot of MetVis, with an already designed metabolic network visualized.

4.4.1.1 Designing a Network

The desired metabolic network component (i.e. metabolite, reaction node, reaction flow, compartment and inhibitor/activator) to be drawn is simply

chosen from the toolbar offered by MetVis (Figure 4.4). A simple mouse click is required to draw the components, which are arranged in the toolbar on the left, on the screen in the case of a reaction node or a metabolite. For a reaction flow, the source metabolites, the reaction node and the destination metabolites have to be clicked in the corresponding sequence.

For reactions and metabolites, additional information should be provided. For example, the name is the most important input as it creates the logical connection with the output of a simulation. This information can be specified explicitly or taken from an already existing model.

The metabolites are represented by a square, and reactions are represented by a rhombus. Reaction and metabolite properties can be defined in a special dialog in the case when the model file will be exported or read from an existing model file. After finishing the design, the work can be saved, resulting in two XML files, one containing the model file and the other containing its graphical representation.

Basically, after this step, the user has a drawing representing the structure of the metabolic network, which satisfies the following conditions:

- representation of the metabolic network as a bipartite graph;
- consideration of effectors (inhibition/activation effects);
- easily updatable in case of improvements in the model;
- exportable in XML;
- separation of graphical information in another XML file.

4.4.1.2 Visualization of Simulation Time Series

The results of simulations are usually delivered as a CSV (Character Separated Values) structured file, containing information about the concentrations of metabolites and flows of reactions varying over time.

The dynamic metabolic behavior contained in this data is expressed visually by an animation showing changing metabolite pool sizes and changing fluxes being represented by differently filled boxes and varying arrow widths, respectively, according to data generated by a simulation.

Figure 4.5 illustrates this idea with a fictitious network. The difference in the fill volume shows the concentration of a metabolite at a specific time. The same idea applies to reactions where the width of the Bezier curves represents reaction rates changing over time. To control the animation, a VCR like toolbar is provided with play/pause/stop buttons and sliders for controlling the time and the speed the animation will run with.

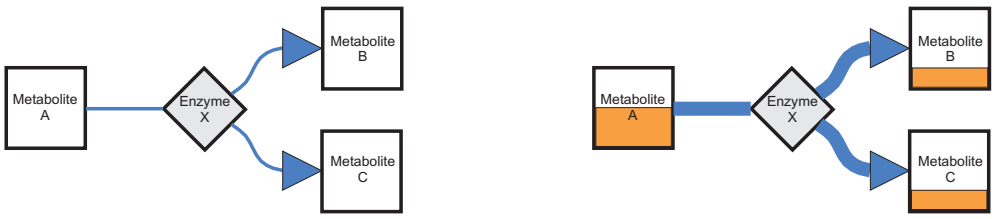


Figure 4.5: Illustration of how animation works

4.4.1.3 Scaling of Simulation Output

Before proceeding with the visualization procedure, the input values for metabolites and reactions need to be scaled to the interval [0, 1] (or [-1, 1] for effectors and reactions). This can be achieved in different ways which will be briefly explained in the following.

The motivation for the scaling is strengthened by the fact that the concentration of single metabolites can vary greatly, and with the scaling, a proper measure for visually comparing them is achieved.

Since the metabolite concentrations cannot be negative, a linear transformation to the interval [0, 1] is carried out. For reactions, which can have negative rates when the inverse flow is greater, a transformation to the interval [-1, 1] is carried out.

Two scaling methods are applied[3]:

- **Local Scaling**, where metabolite concentrations and reaction rates are scaled using the extremum values achieved during all time points in the following way:

$$C_{scaled}(M,t) = \frac{C_{raw}(M,t) - C_{raw-min}(M)}{C_{raw-max}(M) - C_{raw-min}(M)} \tag{4.1}$$

  where $C_{raw-min}(M)$ and $C_{raw-max}(M)$ represent the minimum and maximum of concentrations or rates, respectively. In this case, an empty fill value for a metabolite (maximum fill value) shows that the minimum value (maximum value) of that metabolite is obtained.

- **Global Scaling**, which is basically an extension of local scaling, used firstly by Noack [Noa05], where the scaling is done globally with respect to metabolite concentrations and reaction rates according to Formula 4.2, respectively. In contrast to the first scaling method, an empty fill

[3]Local and global scaling methods will also be encountered in later chapters, when the problem of visualizing time-varying sensitivity matrices is treated

value indicates the overall minimum is achieved.

$$C_{scaled}(M,t) = \frac{C_{raw}(M,t) - C_{global-raw-min}}{C_{global-raw-max} - C_{global-raw-min}} \tag{4.2}$$

Sometimes, it is useful to scale metabolite concentrations based also on external data. Thus, the concentration of some substances such as *AcCoA* grows continually, and by achieving a high maximum they can affect the scaled concentrations of all metabolites in the global scaling method.

4.4.2 Visualization of an E. coli Model with MetVis

The presented ideas about the visualization and animation of metabolic networks were tested in the diploma thesis of Noack [Noa05]. Figure 4.6 presents a model of E. coli, originally published in [CNRS+01], designed with MetVis. It portrays the state of the animation after the substrate pulse of glucose is given (see Subsection 2.2.6). The animation underlies the global scaling (Equation 4.2) in this case, which puts stress on global extrema, both with respect to time and metabolite concentrations or reaction rates, as for example, in the case of *Pts* in the figure. Furthermore, bottlenecks in the reactions can be detected easily in this way.

Figure 4.7 represents the same step, but normalized locally (Equation 4.1). The local scaling method is more sensitive to small changes of concentrations and rates, meaning that there are more dynamics in the animation of the network. This local scaling method is useful for observing the evolution of metabolite concentrations and reaction rates, whereas the global scaling method is useful for comparing metabolite concentrations with each other.

MetVis also provides the possibility to focus on how reaction rates in the network change during the time. To visualize the changes in this case, we make use of the "stock market visualization" metaphor, by visualizing the changes in the reaction rates with a small arrow colored in green or red, indicating an increase or decrease in the reaction rate. Furthermore, positive reaction flows are colored in blue, whereas negative ones are colored in red. Figure 4.8 presents the visualization of fluxes as described above.

4.4.3 Visualization of Models with Effectors

The examples presented until now treated models of E. coli where effectors, i.e. inhibition and activation effects, were not taken into consideration. The quantities describing the effectors need to fulfill the following two conditions in order to be considered in the context of visualization:

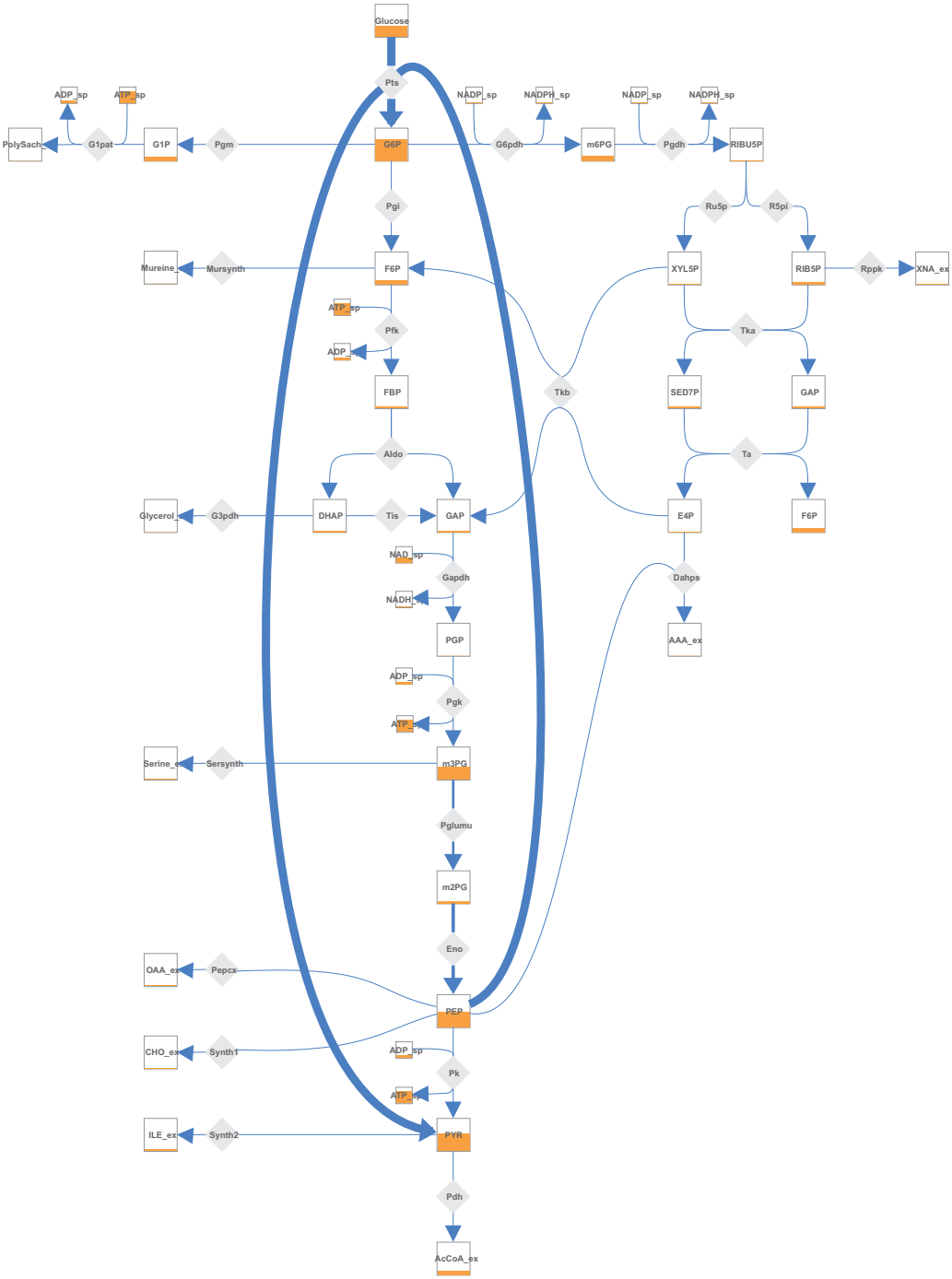


Figure 4.6: Animation of an E. coli model scaled globally. The most active reaction is the one where the substrate Glucose takes part.

- They should be restricted to a certain range to be scalable for visualization;

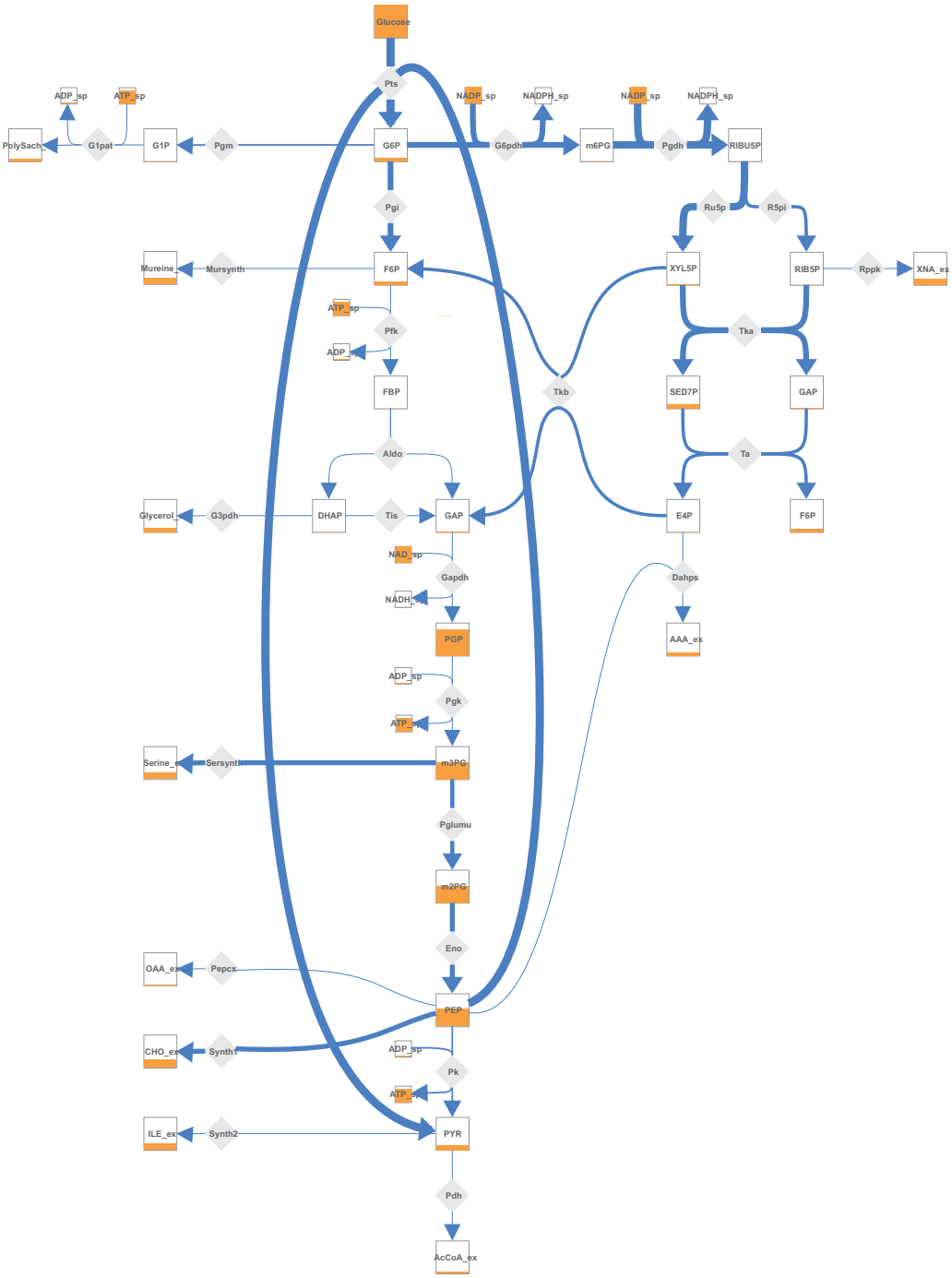


Figure 4.7: Animation of an E. coli model scaled locally. In contrast to Figure 4.6, other reactions are also stressed by the local scaling process.

- The sign, whether positive or negative, should indicate if it is an inhibition or activation effect.

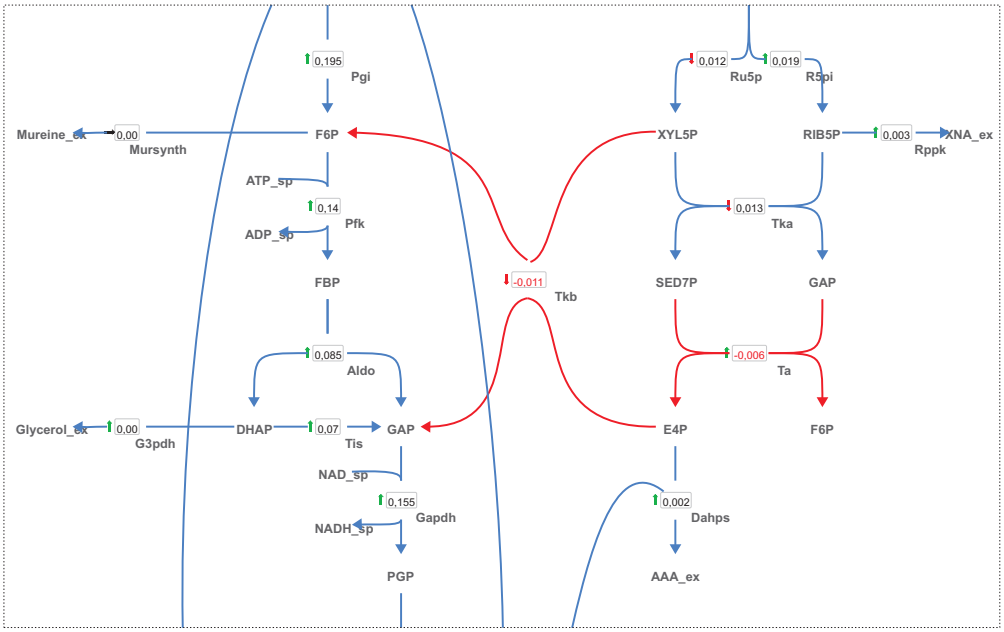


Figure 4.8: Visualization of E. coli model with flux rates stressed

4.4.3.1 Quantification of Effectors

This paragraph describes how the quantification of effectors is performed. We will discuss the two basic cases, where only activation or inhibition takes place, in the context of a Michaelis-Menten kinetic. Further details are available in [Noa05].

Quantification of One Inhibitor. The basic Michaelis-Menten equation with one inhibitor is:

$$r = \frac{r_{max} S}{K_s(1 + \frac{I}{K_I}) + S} \tag{4.3}$$

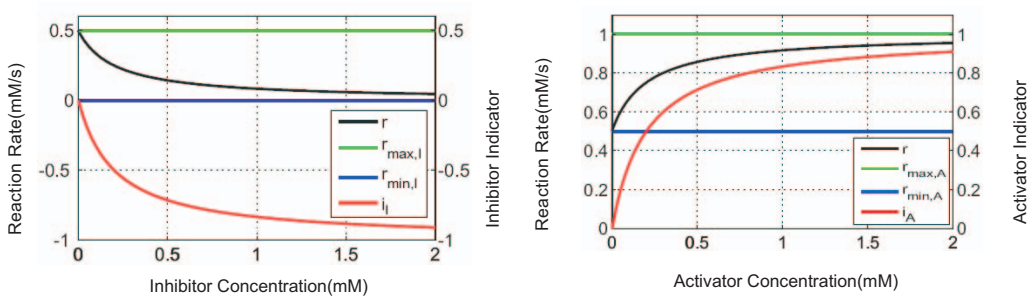
The purpose is to find a measure indicating how much a change in the inhibition level I affects the reaction rate r. This is fulfilled by taking the value of i_I, which is defined as follows:

$$i_I = \frac{r(s, I) - r_{max,I}(S)}{r_{max,I}(S) - r_{min,I}(S)} \tag{4.4}$$

where

$$r_{max,I}(S) = \sup_{0 \leq I \leq I_{max}} r(S, I) \tag{4.5}$$

indicates the reaction rate obtained when the effect of inhibition tends to



(a) Inhibitor case with S=0.1, $r_{max} = 1$, $K_S = K_I = 0.1$

(b) Activator case with S=0.1, $r_{max} = 1$, $K_S = K_A = 0.1$

Figure 4.9: Graphics of inhibition and activation indicators

zero and Equation 4.6

$$r_{min,I}(S) = \inf_{0 \leq I \leq I_{max}} r(S, I) \tag{4.6}$$

indicates the reaction rate when the effect of inhibition is maximal. Thus, the indicator of inhibition i_I takes values in the interval $[-1, 0]$. Figure 4.9(a) illustrates the quantified inhibition indicator in comparison with quantities $r_{min,I}(S)$, $r_{max,I}(S)$ and $r(S, I)$. With increasing inhibition effect I, r_I is increased (in absolute value, because it is negative) and the reaction rate tends to go to zero when r_I tends to go to the maximum.

Quantification of One Activator. The case of one activator follows analogously as denoted in Equation 4.7 where the modified Michaelis-Menten equation with one activator is presented.

$$r = \frac{r_{max} S}{K_s(1 + \frac{A}{K_A})^{-1} + S} \tag{4.7}$$

The index of activation i_A, which is defined as:

$$i_A = \frac{r(s, A) - r_{min,A}(S)}{r_{max,A}(S) - r_{min,A}(S)} \tag{4.8}$$

where

$$r_{max,A}(S) = \sup_{0 \leq A \leq A_{max}} r(S, A) \tag{4.9}$$

indicates the reaction rate obtained when the effect of inhibition tends to go to zero and

$$r_{min,A}(S) = \inf_{0 \leq A \leq A_{max}} r(S, A) \tag{4.10}$$

indicates the reaction rate when the effect of inhibition is the maximum. Thus, the indicator of activation i_A takes values in the interval $[0, 1]$. Figure 4.9(b) illustrates the quantified activation indicator, where an increasing activation effect A also increases r_A and the reaction rate tends to go towards its maximum.

4.4.3.2 Visualization of Effectors

To visualize the effectors, except edges presenting reaction flows, edges representing the inhibition or activation effect connecting metabolites with reactions (enzymes) need to be inserted into the visualization.

These connecting edges are visualized with a red circle for inhibition and a green circle for activation. The circle is placed next to the affected reaction. The changing size of these circles indicates the level of the respective activation and inhibition. Figure 4.10 presents the visualization of the E. coli network after effector related modifications have been made. The state of the network presented in the figure corresponds to the moments before the glucose pulse has been given, which is indicated by the high inhibition that *G6P* exerted on *Pts* reaction. Furthermore, *PEP* show also a high concentration, thus inhibiting the *Pfk* reaction, whereas *FBP* which activates three reactions has a low concentration. Generally, the flow inside the network is on a constant low level, because of the lack of *Glucose*.

Figure 4.11 presents the state of the E. coli network after the glucose pulse has been given. After the pulse, the activation effect *FBP* exercises on *G1pat* and *Pepcx* increases substantially, favorizing thus the production of *OAA* and the increase in consumption of *PEP*. The decrease of the concentration of the latter also decreases the inhibition effect on *Pfk*, leading to a general increase of the production of *OAA*.

4.4.4 Discussion

The above section presented different ideas for visualizing metabolic networks including their animation based on time series of simulation data. Here, some of the choices made during this process and the reasons behind them were discussed.

We begin by discussing color usage in MetVis. Although the colors are customizable, i.e. the user can select her/his own colors, the preselected colors

are chosen so that they make the interpretation of the figures easier. Thus, in concordance with other usages of these colors, inhibition is represented with red edges and activation with green ones.

To indicate the concentration level of metabolites, the visual variable size is used (see Subsection 3.3.6). However, the size of the filling level is restricted to a certain range and its minimum and maximum value is understandable in every moment because they correspond to the size of the square representing the metabolite. The same visual variable is also used for reaction flows, but without surrounding borders to indicate the minimum and maximum size because they would overload the visualization too much.

4.5 3D Visualization of Metabolic Networks

2D visualizations are very helpful for analyzing both topological and other simulation related properties of metabolic networks. However, 2D views have some restricions which could possibly be avoided by using 3D visualizations. Thus, some pathways, as for example the pentose phosphate pathway, cannot be drawn in 2D without line intersections. A much more difficult problem occurs when the metabolic cofactors like *ATP* or *NADH* are involved. They are coupled to almost all central metabolic reaction steps and induce a strong network connection resulting in many line crossings. The crossings issue can be tackled in the following ways:

- by ignoring it;
- by duplicating the metabolites so that we have fewer crossings;
- by using special purpose algorithms to reduce the number of crossings;
- by using 3D visualization techniques to displace edges in different planes.

In the case of metabolic networks, the crossing problem is complicated furthermore when the activation/inhibition effect is considered.

This section presents an approach allowing the visualization (and animation) of the evolution of a metabolic network based on generated simulation data in three dimensions (3D). This approach is an extension of MetVis to 3D.

4.5.1 Transformation from 2D to 3D

Basically, the 3D visualizations we generate are transformations of 2D visualizations into the 3D space following certain rules. The 3D view is intended to be used as a complementary part to the 2D view, allowing the elimination of

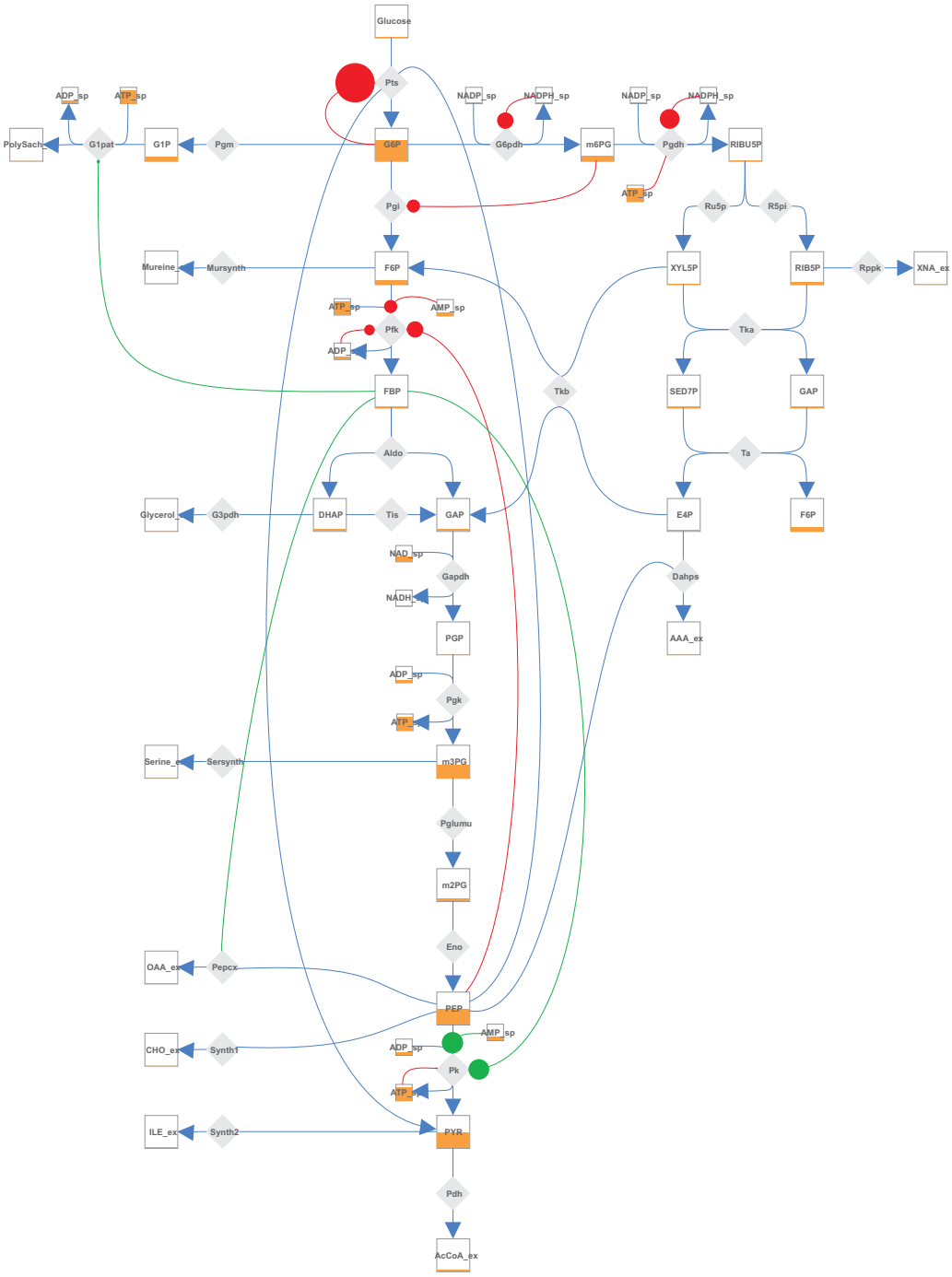


Figure 4.10: Animation of E. coli model with effectors before the Glucose pulse is given

crossings in 2D. Furthermore, the generated 3D views are similar to their 2D counterparts, without creating totally different views which do not conform

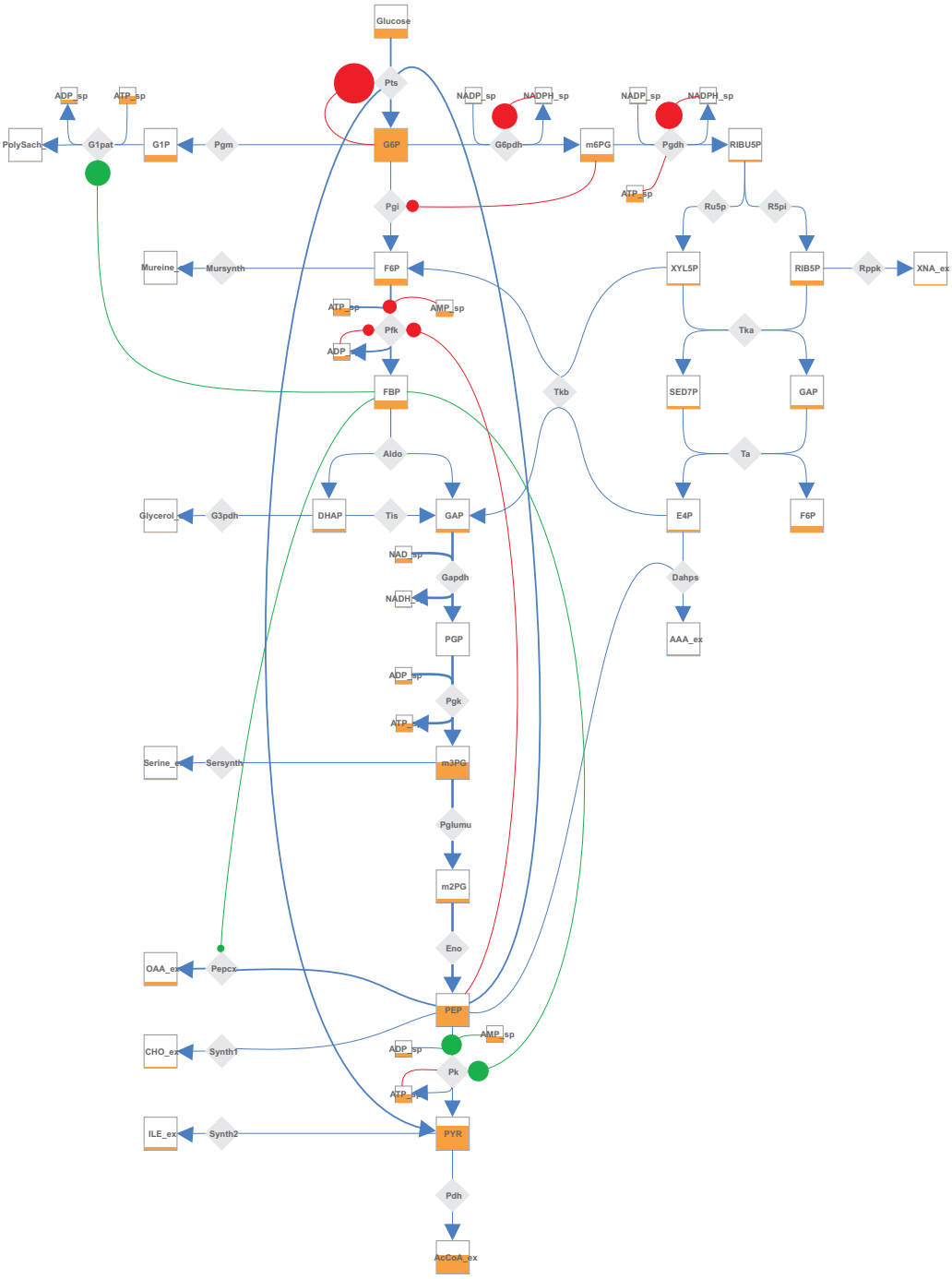


Figure 4.11: Animation of E. coli model with effectors after the Glucose pulse is given

to biochemical conventions and would confuse modelers in their work.

In the 3D visualization, metabolites are represented by 3D cubes, and

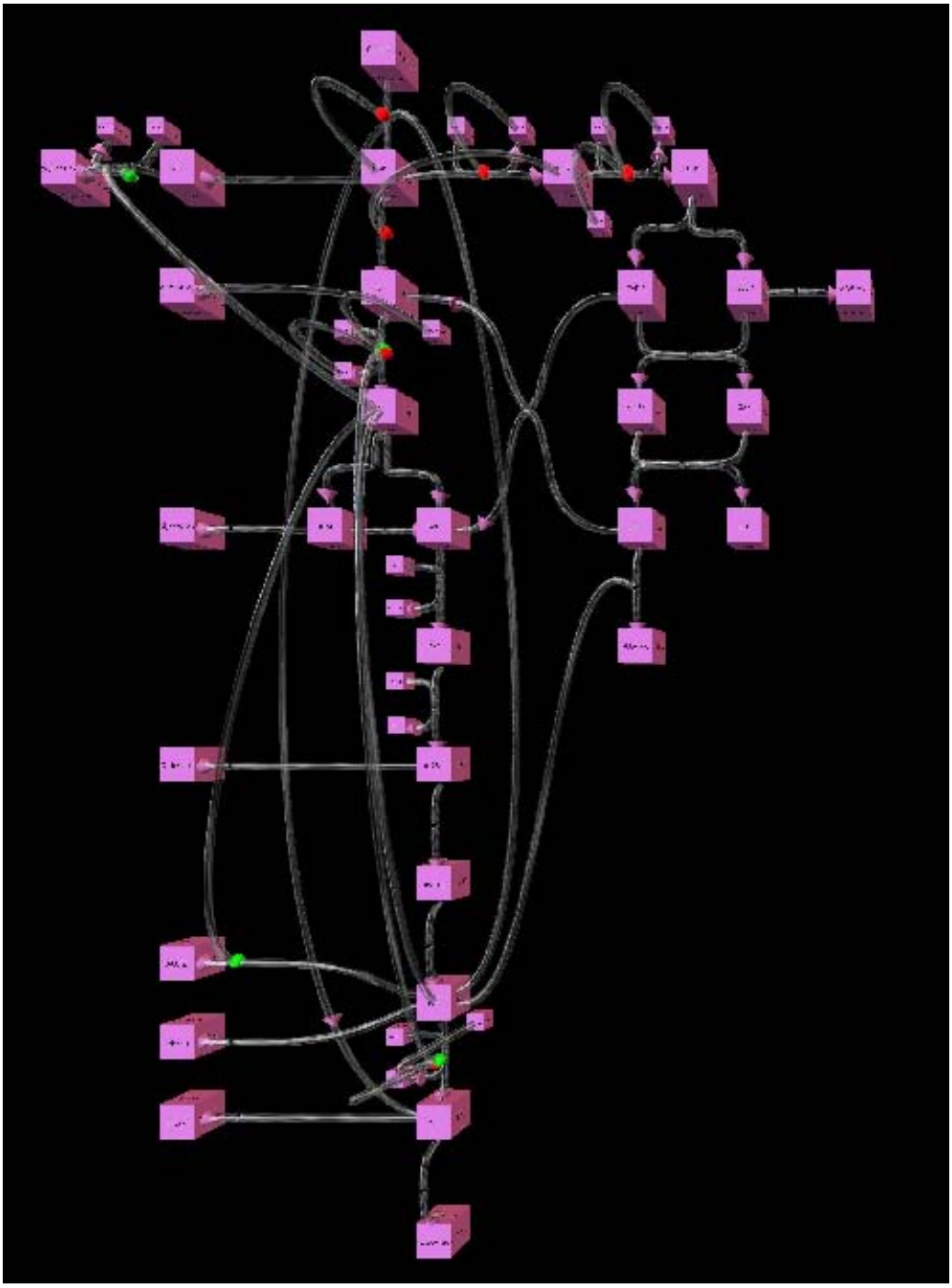


Figure 4.12: Visualization of E. coli model in 3D

edges that in 2D were represented by Bezier curves are now represented by tubes that have the shape of a three dimensional Bezier curve and with a

certain diameter. To eliminate crossings between edges, the control points lying in the middle are displaced in different z-planes.

Figure 4.12 presents the 3D visualization of the model of E. coli considered earlier in this chapter, where the intersecting edges are displaced in the z-plane. A separate z-plane is used for inhibitor and activator edges such that they do not visually affect the rest of the network, as shown in Figure 4.12.

To animate the 3D views, the raw data taken from a simulation is converted into relative percentages of the respective metabolite or flow, as described in Equations 4.1 and 4.2. Two types of objects are animated:

- The z-dimension of the *cubes* representing the metabolites varies according to the scaled concentration of the corresponding metabolite. Figure 4.13 and Figure 4.14 illustrate this effect; the substrate (the highest metabolite) has a low z-dimension before the pulse in Figure 4.13 and a high z-dimension after the pulse in Figure 4.14.

- *Tubes* represent flows for both reactions and inhibitions, and accordingly colored spheres moving within the tubes represent the material flow for the corresponding reaction. For the reaction flows, these spheres are colored with a nuance of blue, for inhibition they are colored in red and for activation they are colored in green. The speed of flow of these spheres depends on the scaled rate of the respective reaction. Depending on the point of time, cubes will show how high the concentration of the metabolite is, and the reaction flow will be faster or slower.

4.5.2 Technical Visualization Issues

The approach has been implemented completely in the Java programming language. For 3D rendering, the idx3d library [Wal00] is used. This library is similar to the well know Java3D library, but offers additional functionality for our specific case and is in pure Java allowing easy inclusion in Java applications and applets without additional libraries. It is worth mentioning that for eliminating crossings, a simple but effective algorithm was implemented to assign different z-planes to the middle control points of Bezier tubes representing edges. The nodes of the metabolic network (i.e. reactions and metabolites) are all left in plane $z = 0$. Only the inner control points of Bezier curves (i.e. edges) that connect metabolites with reactions are shifted to different z-planes. The edges are processed consecutively. In the beginning, the first edge is directly assigned to the first group of edges to be drawn in plane $z = 0$. Then, the second edge is processed; if it intersects with the first edge, a new group is created with edges to be drawn in plane $z = c$, otherwise it is inserted in the first group. The third edge is processed in the same way, if it intersects any of the edges in the first group or second group

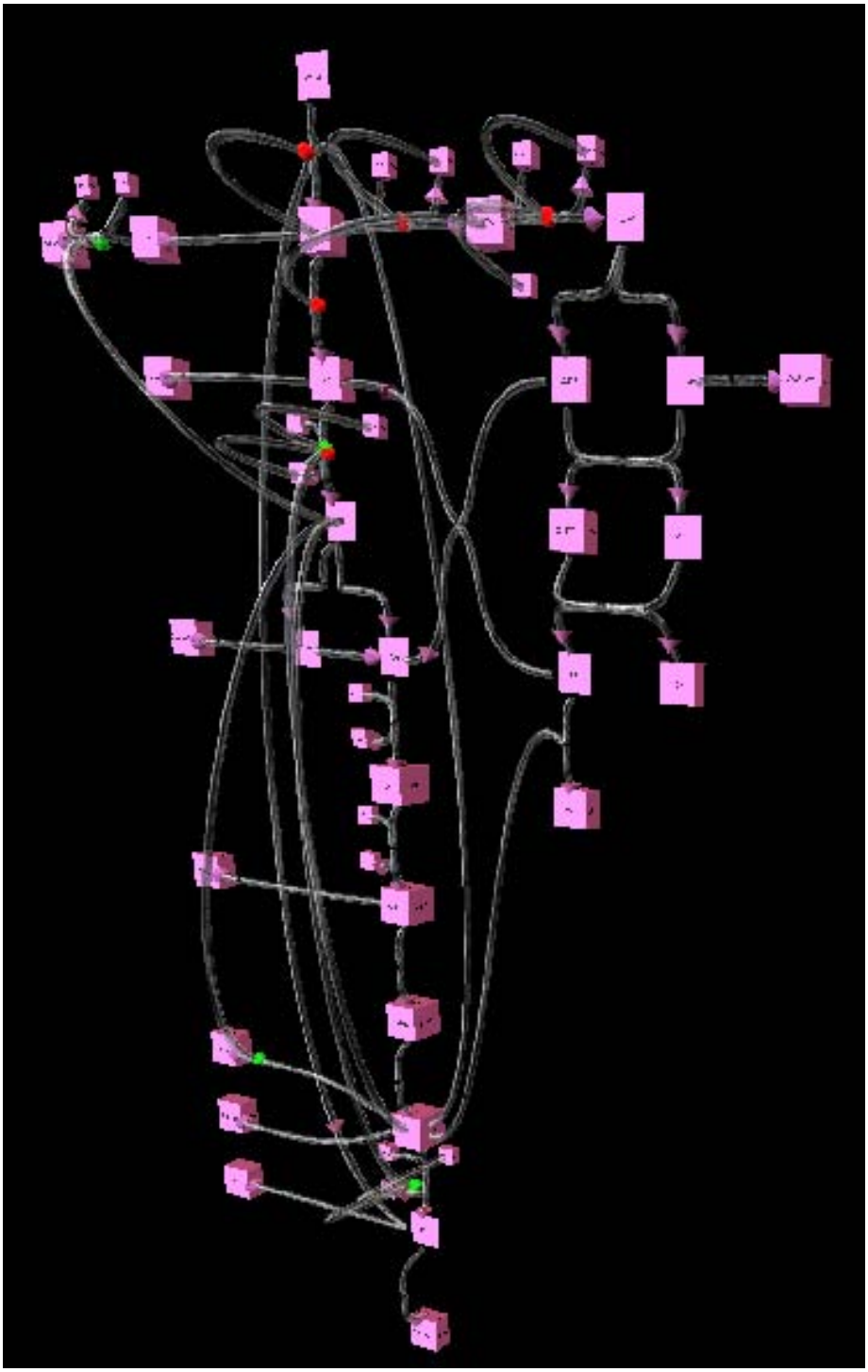


Figure 4.13: Animation of E. coli model in 3D. The point of time here is before the substrate pulse is given.

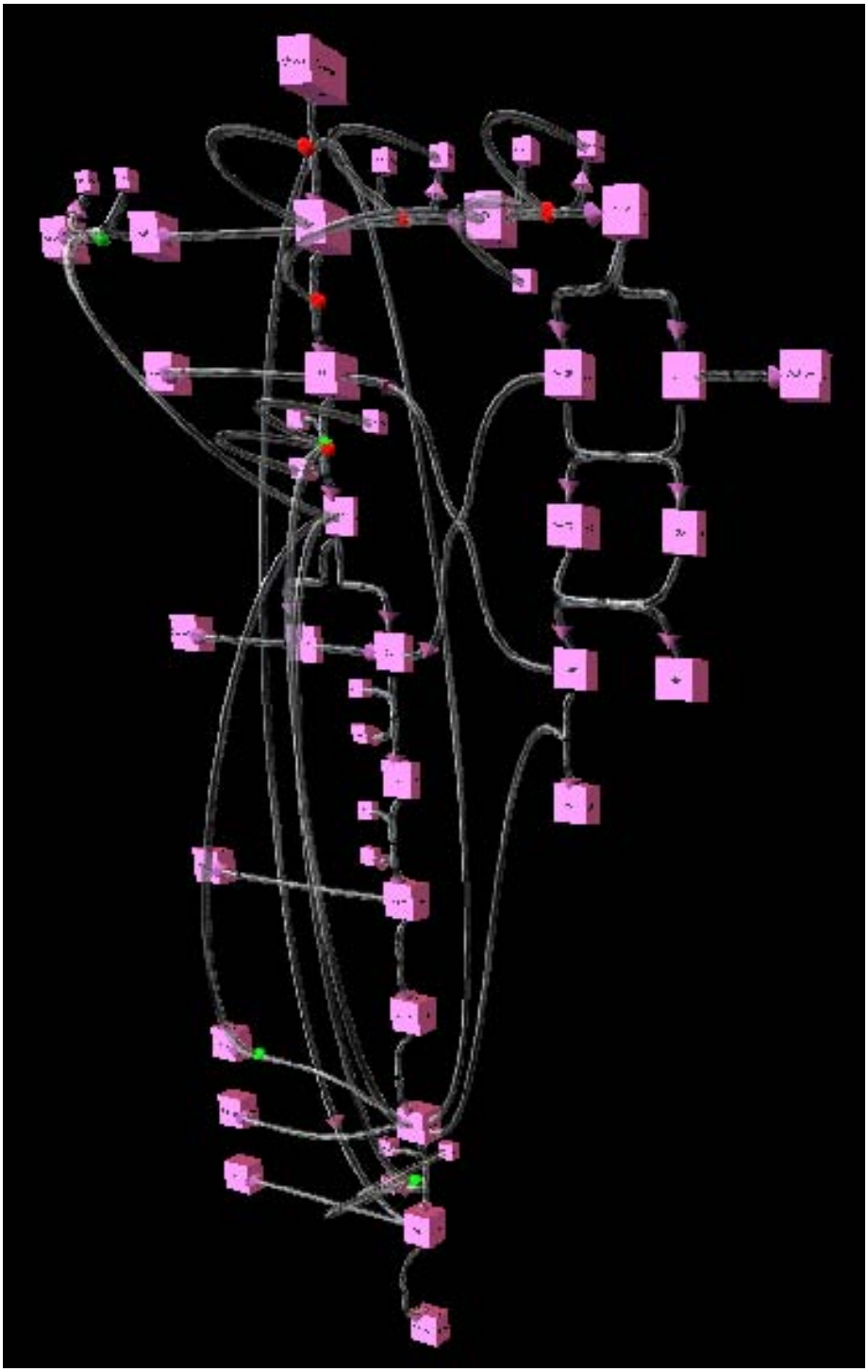


Figure 4.14: Animation of E. coli model in 3D. The point of time here is after the substrate pulse is given.

Algorithm 4.1: Z coordinate assignment algorithm

```
Input: Graph representing metabolic network
Output: Z coordinate assignments for the edges of the network
Initialize an empty array T for storing the respective levels;
Initialize an empty array Z-edges for storing the z coordinates;
for i=1 to Number_of_Edges do
    edgeLevel=0;
    currentList=getEdgesofLevel(edgeLevel);
    doesIntersect=false;
    for j=1 to currentList.length do
        if edge_i intersects edge_j then
            doesIntersect=true;
        end
    end
    if doesIntersect==false then
        T[i]=edgeLevel;
        if i mod 2==0 then
            Z[i]=edgeLevel × c;
        end
        else
            Z[i]=edgeLevel × (−c);
        end
    end
end
```

(if this group exists), then a new group is created with edges to be drawn in plane $z = -c$, otherwise it is inserted in the first group where it does not intersect with other edges. The fourth edge would be drawn in the plane $z = 2c$ if it intersects with at least one edge from every previous group, the fifth in plane $z = -2c$, and the algorithm proceeds in this way until all edges are processed. The pseudocode of the method is presented in Algorithm 4.1.

4.5.3 Discussion

This section presented our approach to visualize metabolic networks in three dimensions. In contrast to 2D visualization, this approach allows the dynamic visualization of metabolic network models without crossings, by using the third dimension to eliminate the possible crossing points.

The visual variable motion was used to indicate the change in reaction flows instead of the visual variable size in case of 2D. Thus, whereas in the 2D animation one must be concentrated to view the changes in different flows,

in 3D animation the speed of the movement of spheres makes it directly clear which part of the metabolic network is more active. For metabolite concentrations, the same visual variable, namely size, was used for visualization.

4.6 Graph Drawing Algorithms for Metabolic Networks

As discussed in Section 4.3, there are several approaches for automatic drawing of metabolic pathways and networks. This section discusses some of them in more detail and presents our approach, *Steerable Drawing of Metabolic Networks*, which unifies the best properties of automatic drawing in a highly interactive process for achieving high quality drawing of metabolic networks. In contrast to the approach presented in Section 4.4, which assumes that the drawing of a metabolic network is done in parallel to the creation of its respective model (i.e. the differential equations describing the model), this section handles a different problem, namely the drawing of the metabolic network after the model has been created without having the layout related information. Thus, the basic problem in this context is how to create a drawing of the metabolic network knowing only its structure.

4.6.1 The Automatic Drawing Problem

Conventional layout algorithms are not capable of realizing the special requirements of drawing metabolic networks. The reason is that common users of programs for visualizing metabolic networks, biologists, biochemists and so on, follow certain conventions which are either not considered during the layout process or contradict with the conventions which are usually taken into consideration during the same process. Basically, the approaches in the automatic layout of metabolic networks are divided in two groups:

- methods which support the layout process by extracting information from the respective databases.
- methods which adapt generic techniques for graph drawing for the case of metabolic networks.

The former approach is based on the assumption that somebody has drawn a specific metabolic pathway/network earlier and stored it in a database. Such an approach would be inappropriate for dynamic models, where some metabolites and reactions are removed or added as is often the case during the modeling process.

In contrast, the latter approach is mainly concerned with how to adapt the generalized graph drawing approach for metabolic networks. Drawing metabolic networks automatically is important in the context of these two problems:

- Visual exploration of large metabolic networks involving thousands of metabolites. Here, the simple details of the structure of a metabolic network are put in the background, and the user is concerned with an overview of the structure of the metabolic network, the main metabolites and reactions with the largest number of connections which serve as hubs, maximum pathway length, etc. Such questions are important especially in the context of exploring the *small world* structure of metabolic networks [WF01, JTA+00]. The drawing algorithms used in this case are general purpose algorithms, usually force-directed approaches with slight modifications.

- Visual exploration of small to medium sized metabolic networks, usually involving one or several pathways will be considered further in this section. Compared to the first problem, the user expects certain conventions regarding the display of metabolic networks to be met. These conventions will be discussed in detail in the following.

The objective of graph drawing in general is to fulfill a set of aesthetic criteria. The extent to which this criteria are fulfilled determines how good or how bad a certain algorithm performs on a certain graph. Some of the most important aesthetic criteria include:

- *minimizing size* for clarity and to fit on a certain area (of a display, paper etc.).

- *minimizing edge crossings*: edge crossings are one of the most important indicators of the readability of the drawing; the smaller their number, the better readability of the drawing.

- *minimizing edge bends*: to improve readability, some of the edges are bent in order to reduce crossings; however, good drawings make use of a minimal number of bends.

- *maximizing symmetry* in cases where symmetry is possible.

- *maximizing the minimum angle between edges leaving a node* to support drawings where outgoing/incoming edges form optimal angles in the sense that they do not occlude each other.

- *maximizing edge orthogonality*: edge orthogonality measures how much edges deviate from the vertical or horizontal direction.

Table 4.2: Importance of drawing aesthetics for metabolic networks

	Generalized GD	**Drawing of MN**
Size Minimization	++	++
Edge Crossing Minimization	++	++
Node Duplication Minimization		++
Edge Bends Minimization[4]	++	++
Symmetry Maximization	++	+
Angle Maximization	++	+
Edge Orthogonality Maximization	++	+
Node Orthogonality Maximization	++	++
Consistency of Flow Maximization	+	++
Cycle Visibility Maximization	+	++

- *maximizing node orthogonality*: node orthogonality measures how good the position of the nodes could be described by an imaginary grid.

- *maximizing consistent flow direction* in the case of directed graphs; understanding flow direction is extremely important when directed graphs are drawn.

Purchase [Pur02] has defined metrics for some of these aesthetics criteria, which can serve to determine the quality of specific drawings or used as objective functions in optimizing drawings.

However, in the context of drawing metabolic networks, only some of these aesthetics are important. Furthermore, some of these are complemented with additional aesthetics which are borrowed from the conventions used in biochemistry books for drawing metabolic networks. Thus, minimization of edge crossings is often achieved by duplication of metabolites which take part in more than one reaction. This is often the case with the so called co-metabolites such as *ATP*, *ADP*, *NADP* and *NADPH*. However, their duplication does not follow strict rules and is usually done ad-hoc during the manual drawing process. Another aesthetic criteria important in the context of drawing metabolic networks is the cycle readability. Metabolic networks often contain one or more cycles which should be clearly identified and properly drawn.

Table 4.2 shows the aesthetic criteria, including those related only to metabolic networks, along with their importance for drawing of general graphs and metabolic networks.

4.6.2 Existing Approaches for Drawing Metabolic Networks

One of the first approaches for automatically drawing metabolic networks was presented by Karp and Paley [KP94]. Their layout algorithm is composed of three main steps:

1. Determining the topology of the metabolic pathway if it is linear, branched, cyclic or complex;
2. Applying layout algorithms appropriate for the topology;
3. Assigning positions to "auxiliary" nodes with respect to the main nodes of the metabolic pathway. This step calculates the necessary space for such nodes.

The pathways are classified according to the following conditions:

1. **Circular pathways** are those pathways which are found to belong to a single circle during a cycle detection process in the graph representing the pathway.
2. **Branched pathways** are those pathways that contain no cycles and could be represented by a Directed Acyclical Graph (DAG) in their representation graph.
3. **Linear pathways** are those pathways that contain no cycles and no branches in their representation graph.
4. **Complex pathways** represent pathways whose representation graph is a combination of two or more of the situations above.

Circular, branched and linear pathways are laid out using algorithms suitable for these kind of graphs. Complex pathways are decomposed further into parts which belong to one of the basic topologies. The proposed approach focuses on models present in the EcoCyc database and produces good results on pathways with basic topologies.

A similar recipe is also followed by Becker and Rojas [BR01], but the basic topologies defined in their approach are only two, namely hierarchical and cyclic. Each reaction is represented by a hyperedge, connecting the metabolites taking part in the reaction. Since the graph drawing library used to draw the basic topologies does not support hypergraphs, dummy nodes with size zero are inserted to represent the hypergraph with a normal graph. Cycles are defined as above as parts of the metabolic pathway where each node is traversed exactly once and are detected by breaking the graph into strongly connected components. The following cases are distinguished:

[4]Bends are usually drawn as Bezier curves or B-splines

- The metabolic network contains no cycle at all, meaning that it is a DAG. Hierarchical layout algorithms are then used to lay out the metabolic network.
- The metabolic network consists of only one entire cycle, which is laid out using circular layout drawing algorithms.
- The metabolic network is complex i.e. it is composed of both hierarchical and circular parts. Then, the largest cycle is found and the procedure is repeated until we have only basic topologies, whose layout is already possible to calculate. The parts of the network obtained in this way are transformed into supernodes of a meta-graph and the resulting meta-graph is laid out using a force-directed algorithm. The supernodes are then expanded and replaced by the proper layout of the representing parts. Furthermore, metabolites which have a high degree of connectivity with a cycle are drawn inside the cycle. As a last step, some enhancements are made to reduce crossings which are induced when supernodes are expanded.

4.6.3 Steerable Drawing of Metabolic Networks

Often, the drawings created with automatic layout algorithms, although customized for metabolic networks, either do not satisfy biochemical conventions entirely or need many manual modifications to come into an acceptable state. Let us illustrate the ideas with the metabolic network representing E.coli model which we considered in Section 4.4. Basically, the network contains no cycles and is a directed acyclic digraph. However, if we feed the graph representing the network to the *dot* program of Graphviz [GN00], which is one of the best libraries for drawing hierarchical networks, then we obtain the drawing in Figure 4.15. From the figure, we see that the quality of the drawing, although the network visualized is simplified by omitting the co-metabolites (*ADP*, *ATP*, etc.), is much worse than the quality of the hand made visualization in Figure 4.6. Two main reasons affect the bad quality of automatic drawing in this case:

- The nodes *F6P* and *GAP*, which are basically the nodes with the highest degree, are duplicated.
- The drawing of two of the hierarchical parts are not branches of a tree but rather perpendicular to the main flow.

Similar problems are also encountered with networks of mixed topology, i.e. networks that contain both hierarchical and cyclical parts. Whereas the drawing of specific parts with clear structure succeeds without problems, their

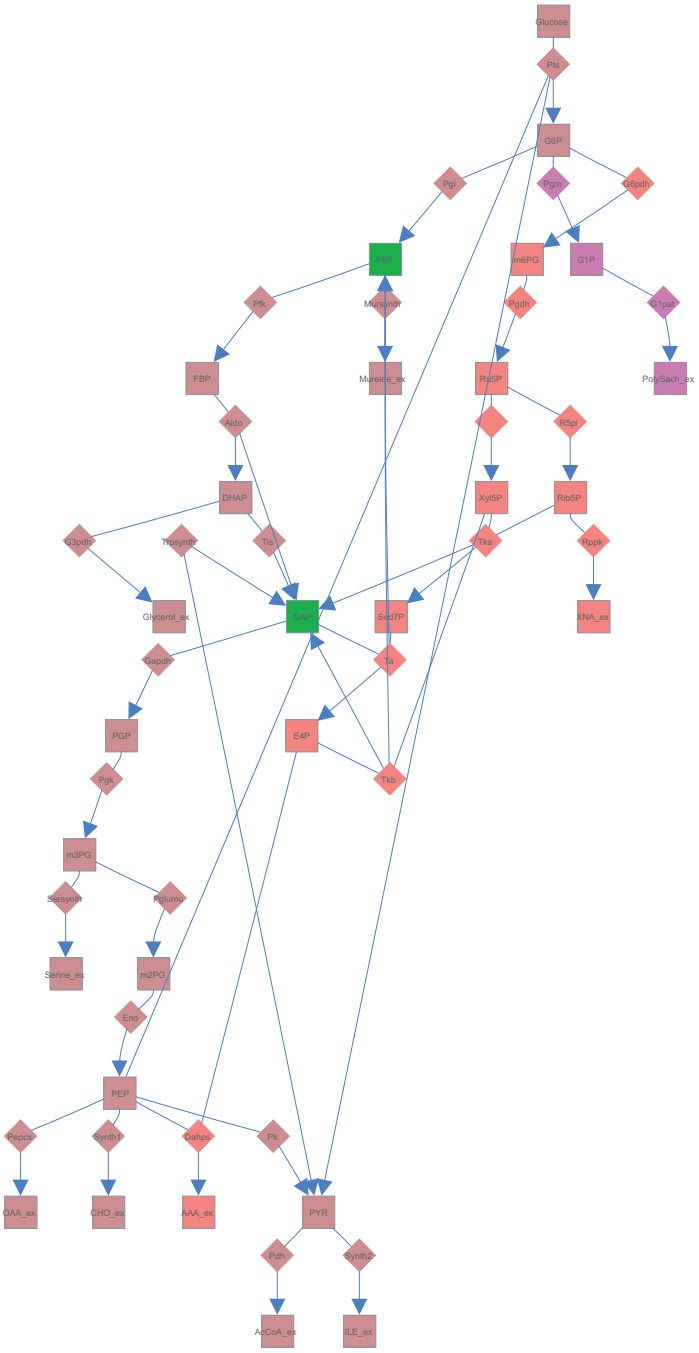


Figure 4.15: Automatic drawing of E. coli model with *dot*

Algorithm 4.2: General schema for drawing metabolic networks

Input: Graph representing metabolic network
Output: Layout of the metabolic network
1 Decompose the network into subnetworks;
2 Draw the subnetworks;
3 Combine the drawings into a single one;

aggregation in a common drawing of the whole network is a difficult process, which strongly affects the quality of the final drawing. To solve these problems, we follow a different route compared to the existing automatic drawing approaches. The drawing process is automated maximally without affecting the quality of the final visualizations. Furthermore, the highly critical steps of putting together the parts of the network already drawn is designed as an interactive process, *steerable by the user*. Thus, we focus on the quality of the visualizations by automating all the steps that do not affect this quality.

4.6.4 The Drawing Process

The fact that metabolic networks have a complex structure with cycles and hierarchical parts intermixed with each other and the requirements regarding the final layout make the drawing problem challenging. As shown above, the *divide and conquer* approach can be beneficial in this case. However, there are several issues that strongly affect the quality of the final drawings, namely the way the network is decomposed, how the separate subnetworks are drawn and their final merging. The general schema of the proposed drawing method is presented in Algorithm 4.2. Basically, the decomposing methods corresponding to step 1 of Algorithm 4.2, can be divided into two groups:

- *Graph theoretical methods* such as finding cycles, strongly connected components and weakly connected components.
- *Biological methods* such as elementary mode or extreme mode analysis.

After decomposing the network with graph theoretical methods, we achieve subnetworks with a simple structure. After elementary mode analysis, we achieve simpler subnetworks but not necessarily with basic structure, i.e. only hierarchical or only cyclical.

Step 2, which is employed after the decomposition, is either a simple application of well known drawing procedures when the structure is simple or a repetition of the same algorithm when a complex structure is still present in the network.

Step 3 is the decisive step which generates the final drawing. While merging subnetworks that have nothing or few things in common (e.g. common

Algorithm 4.3: Steerable drawing process

```
Input: Graph G representing the metabolic network
Output: Layout of the metabolic network
decompositionModes={none,SCC,WCC,Elementary Modes};
currentSubgraph=undrawnNodes(G);
while there are nodes not yet drawn do
    currentDecompositionMode=userSelection();
    for each decomposed subgraph subG do
        preview its hierarchical or force directed layout;
        if results are visually satisfiable then
            Glue the subnetwork subG to the current drawing;
            Mark subG as drawn;
        end
        else
            Repeat the whole procedure with subG as main graph;
        end
    end
end
```

nodes, connecting edges) is relatively easy, the problem becomes much more difficult when subnetworks overlap substantially.

User feedback can affect the results substantially if it is included in steps 1 and 3. Thus, our idea is to build a steerable system that allows the user to interactively select how the network will be decomposed into subnetworks, draw these subnetworks automatically and then *glue* the subnetworks together in a last step.

Algorithm 4.3 presents the steerable drawing procedure. Basically, the method works as follows. First, the user selects how the network should be decomposed. Then, each subnetwork is previewed on the top of the current visualization and its settings are adjusted. The drawn subnetwork is glued to the rest. This procedure is repeated for every part of the entire network. The subnetworks can be individual reactions (with related metabolites), strongly connected components or elementary modes.

The following paragraphs describe different ways how metabolic networks can be decomposed.

4.6.4.1 Graph Theoretical Decomposition

A common step followed when drawing metabolic networks is represented by finding the cycles present in the network. These cycles are then laid out using force directed algorithms, which generally deliver very good results with such

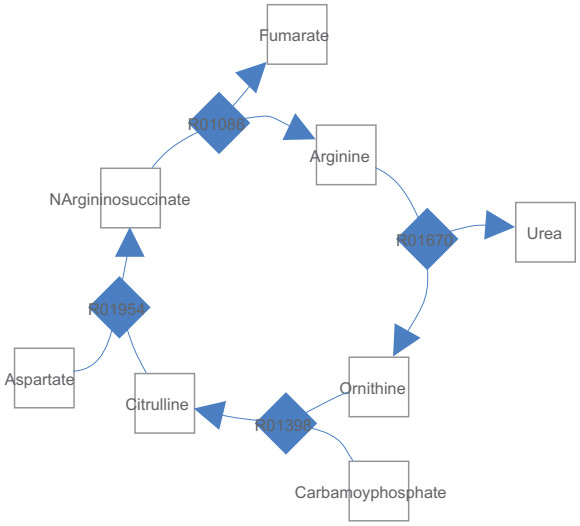


Figure 4.16: Automatic drawing of Urea Cycle with neato

structures. Figure 4.16 and Figure 4.17 represent two cycles, namely the urea cycle and tca cycle drawn automatically by using neato [GN00].

Furthermore, there are situations where cycles intersect with each other and dividing them from each other is a counterintuitive step. In this case, finding *strongly connected components*, which quite often represent more than just cycles, is more interesting because force directed algorithms deliver very good results in this case. Detection of *weakly connected components* can be used in networks where the structure graph contains more than one connected component.

4.6.4.2 Biological Decomposition

Elementary Modes. Extraction of elementary modes constitutes an important step in the context of stoichiometric network analysis. To explain what the elementary modes mean, we need to return shortly to the model of a metabolic network presented in Section 2.3.1. In steady state, metabolite concentrations are equilibrated, implying that Equation 2.3 is simplified to:

$$N \cdot v = 0 \tag{4.11}$$

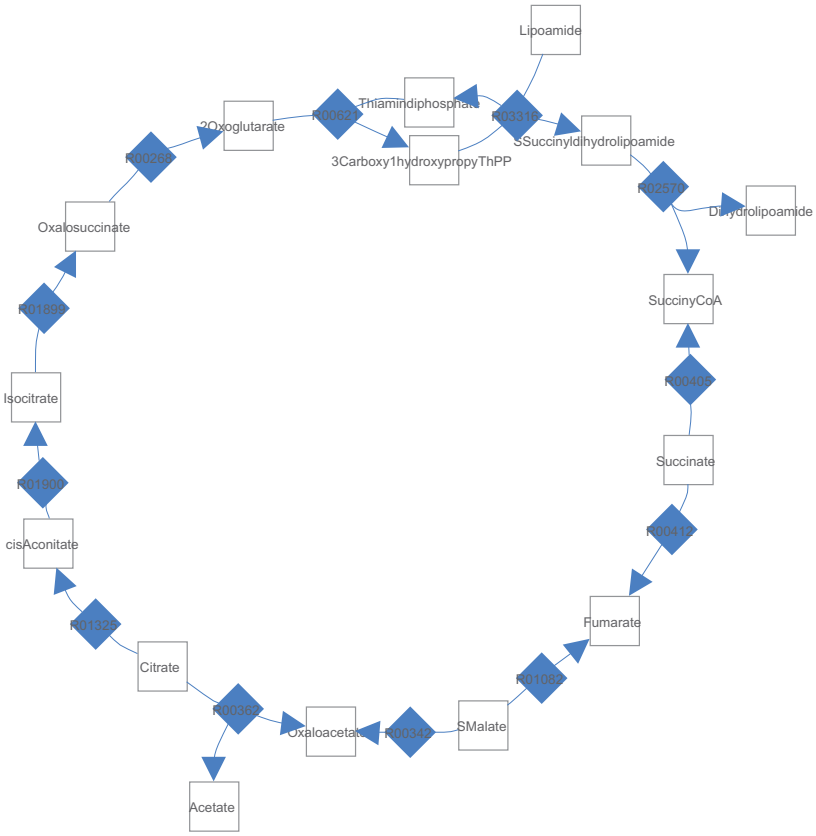


Figure 4.17: Automatic drawing of TCA Cycle with neato

To analyze the steady states of the network, the concept of elementary modes is defined, which represents minimal sets of reactions (and thus metabolic routes) which connect the inputs to the outputs of the biochemical network provided that no accumulation takes place within the network. Furthermore, they also describe the presence of internal cycles.

Basically, the elementary modes are derived from the set of vectors v satisfying equation 4.11, which define the null-space of N. For the irreversible reactions to take place, they must fulfill the condition:

$$v_{irr} \geq 0 \tag{4.12}$$

Extreme Modes. Extreme modes are detected in a similar way with the elementary modes but they present a more compact subset of feasible states.

The decomposition of the metabolic network into elementary modes results in a large number of elementary modes, which increases exponentially with the network complexity. Furthermore, these elementary modes have large intersection parts with each other. Thus, using the technique in an automatic drawing approach to decompose the metabolic network would increase rather than decrease the complexity of the problem due to issues related to the layout and merging of subnetworks with large intersections. However, in an interactive drawing process they can be used up to a certain degree to simplify the process.

4.6.4.3 The Merging Process

During the execution of Algorithm 4.3, decomposition and merging is considered as concurrently running processes. However, the drawing of subparts can generate layouts which contradict each other and thus need to be adapted in order to be merged with each other. For this purpose, we allow the user to preview the layout of the subgraph before gluing it to the rest. Furthermore, we allow the user to perform the following operations on the preview:

- moving up, down, right or left the previewed layout.
- rotating it clockwise or counter-clockwise.
- increasing or decreasing the space where the layout of the subgraph is drawn.

Figure 4.18 illustrates a merging example, where the Urea Cycle previewed in Figure 4.16 is rotated until it fits into the previous drawing of TCA Cycle in Figure 4.17. The toolbar which controls the merging is presented in Figure 4.19. The list shown in Figure 4.19 is the list of reactions which identify the parts of the network to be drawn, whereas the buttons allow the accomplishment of the operations described above.

4.6.5 Discussion

Steerability allows the user-guided drawing of metabolic networks. Automatic approaches, although tailored to metabolic pathways, deliver often results which are far from the biochemical conventions that the user is expected to see.

By allowing users to immediately preview what kind of layout hierarchical or force directed algorithms generate with a subpart of the network, it enables the user to achieve the expected results. The ability to quickly see that a

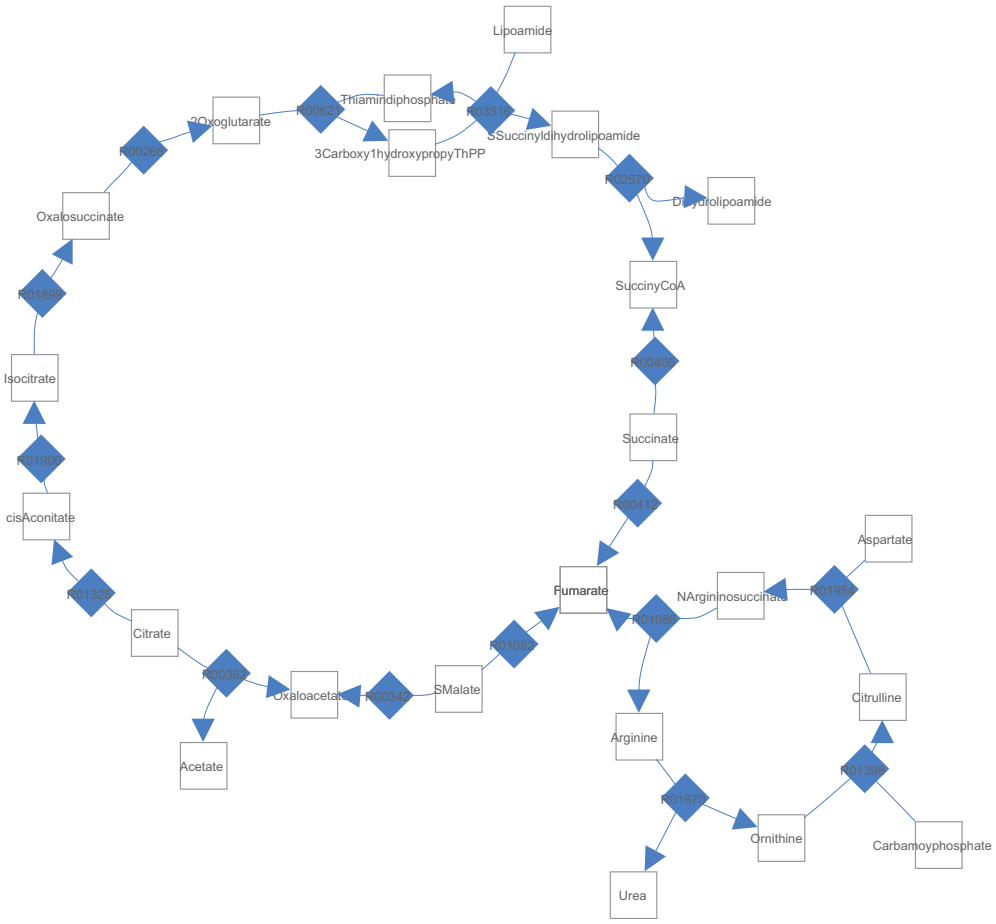


Figure 4.18: Merging of Urea and TCA Cycle. The urea cycle is rotated until its drawing fits to the drawing of TCA cycle.

layout is not promising for a certain part of the network and to abandon such layouts offers great flexibility in this process. In this way, the novelty of the approach consists of bringing steerability to the drawing process. This results in more time dedicated to the drawing process compared to the automatic drawing approaches, and also in better results of the layout. Figure 4.20 shows the network of Figure 4.15 drawn using the steerable approach. It has the same structure as the hand-drawn model in Figure 4.6; the difference consists in the drawing process. The steerable drawing approach allows a faster drawing of the network while keeping the high quality of the drawing.

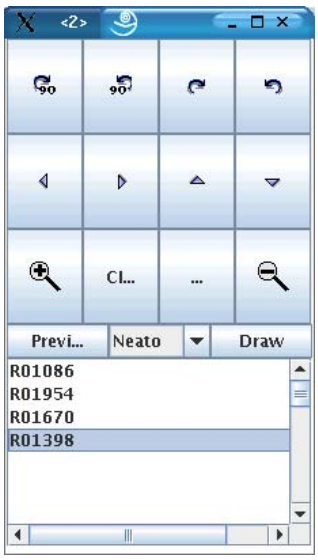


Figure 4.19: The toolbar which controls the merging process

4.7 Summary

In this chapter, we have presented MetVis, a tool which supports the design, visualization and animation of metabolic networks. In contrast to previous approaches, MetVis allows to graphically animate the dynamic evolution of a modeled cell's metabolic properties based on time series generated during the simulation. Furthermore, we provided a prototypical 3D visualization approach, which permits the drawing of complex networks in three dimensions, thus enabling the reduction of edge intersections. Finally, a steerable drawing technique was proposed for high quality drawing of metabolic pathways. By the inclusion of user feedback in the drawing process, we ensure that biochemical conventions related to layout of metabolic networks are satisfied.

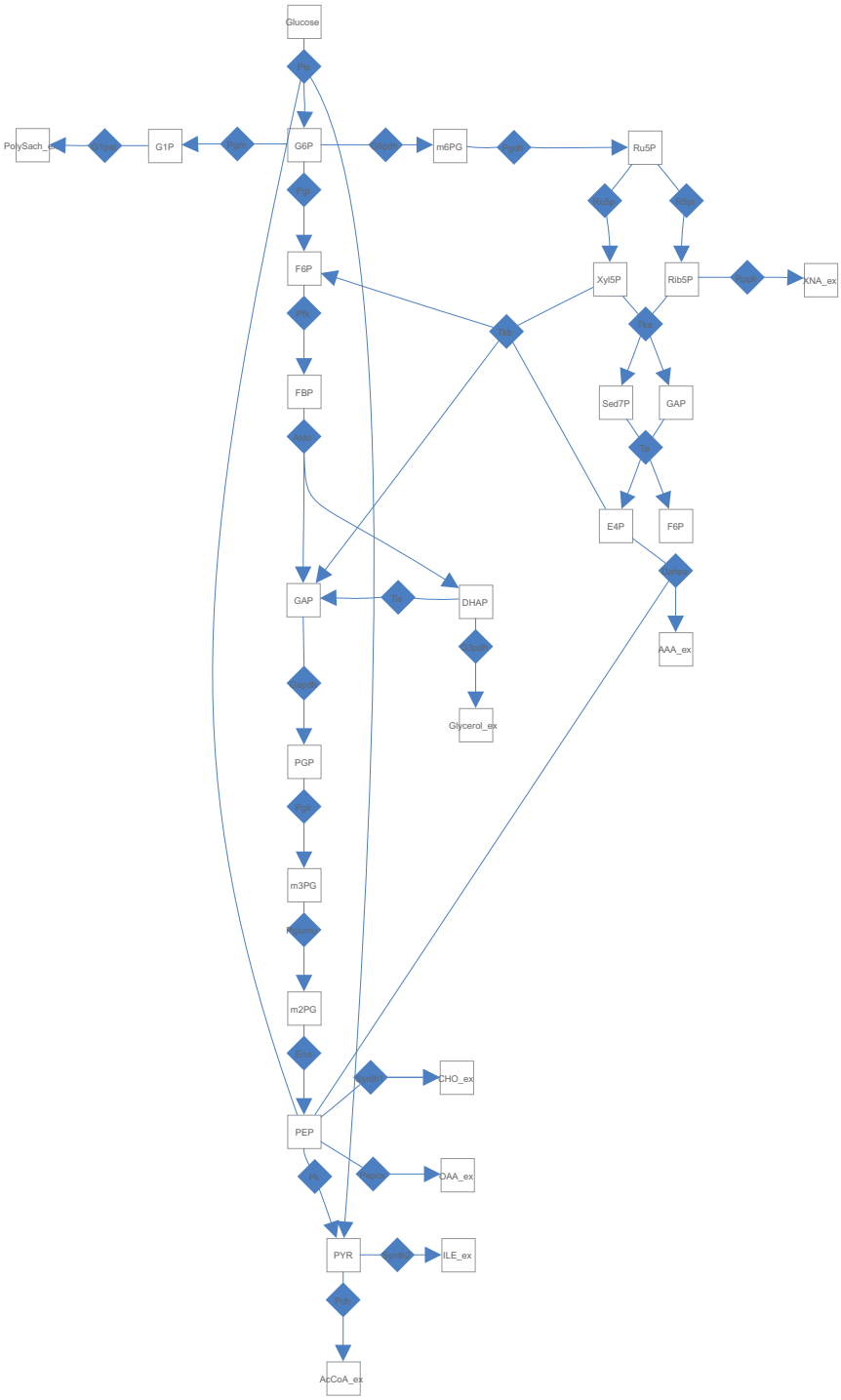


Figure 4.20: Steerable drawing of E. coli model

5
Customizable Comparison of Metabolic Network Models

5.1 Introduction

Biological data, especially data related to metabolic network modeling, is often represented as structured data stored in XML files. A common problem encountered in this context is how to structure this information to provide a basis for compatibility and reusability between a broad range of sources and tools. Furthermore, a suitable format should represent the main features of the represented objects and should serve as an exchange language between different systems. After a certain format has been fixed, a very important problem is how to detect changes in the underlying data sources. The purpose of change detection is twofold:

- To detect changes between different files in order to grasp the similarity that exists between these files.
- To track changes in a family of files which are derived from a common source.

However, the different techniques for comparing XML data are generic in their purpose and do not take into account the semantics of the underlying data. For this purpose, a novel customizable technique for detecting changes in XML files, will be proposed in this thesis. It can be adapted to different contexts and takes into consideration formalized format semantics. Then, existing techniques are used to detect exact changes in metabolic network

models stored in XML format. Parts of this work are published in [QF06, QGF06].

The remainder of this chapter is organized as follows. Section 5.2 gives a survey of existing techniques for comparing structured data, especially XML data and techniques related to change visualization in similar contexts. Section 5.3 describes the SBML format and its dialect, M3L. Section 5.4 presents a novel *extensible and customizable XML diff algorithm*, which takes into account special semantics of SBML before doing the comparison of XML files. Section 5.5 describes the evaluation of the approach. Section 5.6 concludes this chapter.

5.2 Related Work

5.2.1 Comparison of XML Documents

Approaches for comparing structured data can be divided into different groups. Methods that compute the minimum cost edit distance constitute an important group. Their drawback usually consists of the fact that their computation is expensive. Thus, other heuristics which sacrifice accuracy for speed have been developed.

In the following, an overview of the different types of algorithms used to determine document similarity is given. These techniques include tree edit distance based similarity, tag similarity, Fourier transformation based similarity, and path similarity.

5.2.1.1 Tree-Edit Based Similarity Measures

Considering that XML documents can be thought of as trees, a natural approach to the comparison problem would be to use tree-to-tree editing techniques [Sel77, Tai79] to detect changes in XML documents. The algorithms used to achieve this purpose are derivatives of the edit distance between strings, which search for the best sequence of edit operations that transform a tree into another.

The algorithms that compare trees can be divided into two groups:

- algorithms that consider XML files as ordered trees, i.e. where the order of subtrees and leaves is important;
- algorithms that consider XML files as unordered trees.

One of the first algorithms working with ordered labeled trees has been proposed by Zhang and Shasha [ZS89]. For two ordered trees T_1 and T_2, where each node has an associated label, their approach for finding an edit

script which transforms T_1 into T_2 has a time complexity of $O(|T_1| \times |T_2| \times \min\{\text{depth}(T_1), \text{leaves}(T_1)\} \times \min\{\text{depth}(T_2), \text{leaves}(T_2)\})$, where depth() returns the depth of the tree given as parameter.

Chawathe et al. [CRGMW96] formulated the change detection problem on hierarchically structured data, proposing an efficient algorithm based on the assumption that when comparing two labeled trees T_1 and T_2, any leaf in T_1 is at most matched by one leaf in T_2. The time complexity achieved in this way is of the order $O(ne + e^2)$ where n is the number of leaves and e is the edit distance between the two documents. This assumption works well for documents that do not contain duplicates, but it does not perform well for all XML documents.

Two other algorithms for change detection in ordered trees are XMLTreeDiff [CE99] and XyDiff [CAM02].

XMLTreeDiff computes hash values for the nodes of both documents using DOMHash [MTU00], reducing the size of the two trees by removing identical subtrees. Zhang and Shasha's [ZS89] algorithm is then employed to generate the difference between the two simplified trees. However, the removal of identical subtrees might conflict with the cost model employed by Zhang and Shasha's algorithm, possibly generating non-optimal results.

XyDiff is a heuristic which calculates a signature and a weight, which depends on the subtree size for every node on the two documents to be compared. Starting at the root node, the signatures are then compared with each other. If they are equal, then the nodes are matched; otherwise, the child nodes will be inserted in a priority queue and will be matched against each other in decreasing order of weight. Exact matches between subtrees are then propagated to the upper levels according to the weight of the subtree. Furthermore, XyDiff avoids the full evaluation of alternatives by using simple heuristic rules when more than one potential candidate can be matched. The overall complexity of XyDiff is O(n log n), but because it makes use of heuristic rules it cannot guarantee optimal results.

Nierman and Jagadish [NJ02] provide a dynamic programming approach for evaluating the tree edit distance between (ordered) XML documents by taking into account XML issues such as repetitions in sub-elements and optional nodes.

Handling unordered trees, on the other hand, is a much more difficult problem. Zhang et al. [ZSS92] proved that the general unordered tree-to-tree correction problem is NP-complete. Zhang et al. also proposed a polynomial-time algorithm based on a restriction that matching is only performed between nodes at the same level [Zha93]. Mh-Diff [CGM97] is a heuristic for detecting changes in unordered structured data. The edit script is found as an edge cover in a bipartite graph and the basic edit operations foresee not only the usual insert, delete or update but also operations on subtrees such as subtree copy and subtree glue. The complexity of the approach is of the

order O(n^3).

Wang et al. [WDC03] present a change detection algorithm specialized for XML documents. Their approach assumes that XML trees are unordered and it exploits XML specific features to overcome the NP-completeness of the unordered tree comparison problem.

The algorithms presented above have a common property, namely they calculate both a numerical value indicating the similarity between the documents and they also output an edit script transforming one document to the other, which is not the case for the other similarity algorithms which will be presented in the following paragraphs. To convert this similarity value into a similarity metric, this value is often normalized with the number of nodes of the tree representing the larger document.

5.2.1.2 Tag Based Similarity Measures

The presented techniques are used both for matching structures of documents and their contents. However, when only structure matching is needed, a simple method is to measure how closely the sets of tags match between documents.

Buttler [But04] defines a so called Weighted Tag Similarity as follows: Let T_1 and T_2 be the two sets of tags of documents D_1 and D_2, t_{1k} and t_{2k} members of T_1 and T_2, w_{1k} be the number of times tag t_{1k} appears in T_2 and v_{2k} be the number of times tag t_{2k} appears in T_1 and n the number of unique tags.

Then, in analogy with the *Dice Coefficient*, the similarity between documents is defined as:

$$WTS(D_1, D_2) = \frac{\sum_{k=1}^{n} 2 \times min(w_{1k}, v_{2k})}{\sum_{k=1}^{n}(w_{1k} + v_{2k})} \tag{5.1}$$

Considering that this approach takes into account only the set of tags in each document, it will be very sensitive on the tags used. Thus, if HTML data is compared in this way, where the tag set is fixed but structure may vary than the accuracy would be low.

5.2.1.3 Path Based Similarity Measures

To avoid the time complexity restrictions that tree edit based distances pose and the accuracy problems of tag based approaches, path based similarity measures are used.

Joshi et al. [JAKN03] introduced a bag of paths model for measuring the structural similarity of tree models such as DOM trees representing HTML and XML documents. Two different techniques are used in this context:

1) the first considers simple tree paths which retain all child-parent relationships ignoring sibling relationships; and 2) the second technique called *bag of XPaths* includes some sibling information into the model. The simple bag of paths model, which constructs a multiset containing all root-to-leaf paths, functions well with trees containing many levels of nesting and has certain difficulties with shallow trees that contain many similar paths. The bag of XPaths defines a Jaccard-alike similarity measure on generalized XPaths, which incorporate positional information on nodes that accounts for sibling relationships. In this way, this model better accommodates issues encountered often in XML such as repetition, optional elements and recursive elements. Both methods are used as similarity measures for clustering XML/HTML files with different structures, where the bag of XPaths model performs better than the bag of paths model.

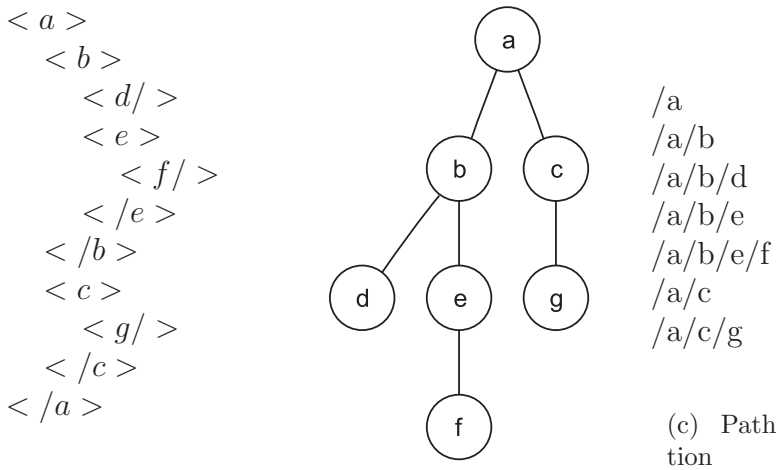
A similar idea is followed by Buttler [But04], where a shingle-based approach for comparison of semi-structured data is introduced. Shingles were introduced originally by Broder [Bro97] to compare text documents for similarity and containment. Using these sets of shingles, the resemblance and containment between two documents is defined as in Formulas 5.2 and 5.3 respectively, where S represents the set of shingles for the respective document. Furthermore, the set of shingles S obtained for a certain size w is reduced into a subset of itself, called sketch, which is further used to determine document similarity. These sketches are random samples of text in documents, which are chosen after applying random permutations to the shingles, i.e. words of the document. Two selection methods are used by Broder [Bro97], namely choosing the smallest s indices after permutation or the indices that are multiples of a parameter m. It is shown that by using these sketches, unbiased estimates of resemblance and containment are found by retaining only a small part of the document.

$$r_w(D_1, D_2) = \frac{S(D_1, w) \cap S(D_2, w)}{S(D_1, w) \cup S(D_2, w)} \tag{5.2}$$

$$c_w(D_1, D_2) = \frac{S(D_1, w) \cap S(D_2, w)}{S(D_1, w)} \tag{5.3}$$

Semi-structured documents such as XML have the advantage that they do not need to be partitioned arbitrarily into sequences of words because they can be viewed as sequence of root-to-leaf branches. In this way, a XML document is represented as a list of tokens, representing root-to-node paths for every node. Figure 5.1 illustrates how a path list is constructed from the tree representing a XML document.

To measure the similarity between two documents, their respective lists of paths can be compared. This comparison can proceed in different ways,



(a) XML document

(b) Tree representation

(c) Path representation

Figure 5.1: XML documents and its tree and path representations

namely:

- by comparing the two lists of paths as sets, measuring the ratio of their intersection over their union
- by building weighted models where different paths have different weights
- by applying the shingle technique mentioned above for the path lists

Buttler [But04] uses the last approach to measure the similarity of structures of XML documents. Instead of tokens in word documents it considers root-to-node paths which are later transformed via a hash function. Out of these hashed values either sets or bags of shingles are built, which represent the values $S(D_i, w)$ in Formula 5.2 or 5.3. The rest then proceeds like for text documents.

5.2.1.4 FFT-Based Comparison

Another approach for comparing XML documents has been introduced by Flesca et al. [FMM+02, FMMP05], where a Fourier transformation is used to compare encoded XML documents. It is based on the idea to interpret an XML document as a time series, where the values combine relevant features of the XML elements. These time series can be compared with each other as being discrete-time signals by using DFT (Discrete Fourier Transform).

The whole procedure can be summarized as follows. First, an appropriate tag encoding scheme is chosen. In the first proposal [FMM+02], a simple encoding scheme using a randomly chosen linear order for all distinct tags was used. Since both opening tags and closing tags are encoded, the technique uses a simple trick where closing tags are encoded with the opposite negative value. This simple technique means that no neighborhood relationships between nodes of XML documents are exploited. Later [FMMP05], the scheme was further enriched with two other techniques, namely pair wise tag encoding where two consecutive tags are encoded together with a numerical value and nested tag encoding which also considers the path names and not only the tags.

After an encoding function for tags has been defined, the document itself is encoded into a time series. Thus, a document D with tags $[t_0, t_1, ..., t_n]$ will be encoded as a time series $[S_0, S_1, ..., S_n]$ where

$$S_i = \gamma(t_i) \times B^{maxdepth(D)-l_{t_i}} + \sum_{t_j \in nest_d(t_i)} \gamma(t_j) \times B^{maxdepth(D)-l_{t_j}} \qquad (5.4)$$

B is usually chosen as the number of distinct symbols present in tag names so that different nesting levels do not interfere to each other's contribution and $nest_d(t_i)$ represents the ancestors of t_i. $\gamma(.)$ represents the tag encoding function, i.e. a function that assigns numerical values to the distinct tags of the XML document.

The final distance between documents D_1 and D_2 encoded as time series h_1 and h_2 is defined as:

$$dist(D_1, D_2) = (\sum_{k=1}^{M/2} (|[\widetilde{DFT}(h_1)](k)| - |[\widetilde{DFT}(h_2)](k)|)^2)^{\frac{1}{2}} \qquad (5.5)$$

where $\widetilde{DFT}$ represents an interpolation of DFT to frequencies common to both documents and $M = N_{d_i} + N_{d_j} - 1$.

This approach assumes that XML documents are ordered, and indeed the ordering affects the tag encoding process substantially.

5.2.1.5 Discussion

Buttler [But04] presents an empirical evaluation of the accuracy and performance of the different techniques explained. To compare the different metrics with each other, which sometimes is not quite easy because the metrics are not directly comparable to each other, clustering (such as k-means clustering) based on both real and synthesized data is performed based on

the assumption that the clusters that are generated by the same algorithm are comparable with each other.

For HTML pages extracted from known websites, the weighted tag metric has the best results, whereas the FFT based approach has the worst; path based approaches and TED are somewhere in the middle. The bad performance of TED is justified by the simple set of tags HTML has to offer.

In terms of speed, the TED based approach is the slowest followed by FFT based approach. Path based approaches are the fastest with the weighted tag approach being the fastest among them.

When dealing with large files, the tree edit distance becomes dramatically slow. Path based approaches are more efficient and furthermore they can be enhanced using shingle based techniques. Buttler [But04] concludes that the approach based on the Fourier transform is the slowest and the least accurate technique and the tree edit distance may not be appropriate for the purposes of clustering whereas path based approaches present efficient heuristics for measuring similarity.

It should be mentioned that whereas TED algorithms compute also a *delta* accounting for the differences between documents, the other techniques concentrate only on the computation of similarity values between documents.

5.3 SBML and M3L

This section is concerned with SBML and its dialect M3L which were briefly described in Chapter 4. XML has become a common standard for information representation and exchange in the context of biological data. These biological data range from protein and gene interaction data to metabolic pathways, and although interconnected with each other, for obvious reasons they differ in structure and scope. The next subsection will give a survey on XML standards for describing biochemical pathways.

Several databases and simulation tools in bioinformatics and especially in metabolic engineering support the exchange and processing of XML data. This XML data can be grouped under several ongoing standardization efforts such as SBML [HFS+03], PSI-MI [HMPB+04], etc. In [SL05] a comparison study for three such standards is performed.

The Proteomics Standards Initiative Molecular Interaction XML format (PSI-MI) [HMPB+04] is a proteomics standard focusing on protein-protein interactions. As such, it represents an unnatural language for storing pathways although they can be represented indirectly via interactions. Furthermore, this format is not intended to be used in simulation environments and is thus inappropriate in this context.

5.3.1 SBML

The Systems Biology Markup Language(SBML) [HFS+03] was created as a cooperation from representatives of many system and tool developers working with models of metabolic pathways. Its development is based on merging of common features of different tools such as BioSpice [Bio06], DBSolve [GHS99], E-Cell [THT+99], Gepasi [Men93], Jarnac [Sau00], StochSim [MFB98] and Virtual Cell [SL99].

Its releases are organized into levels, with level 2 being currently in use, but there is ongoing work on level 3. The number of systems supporting SBML is more than 80[1], with systems ranging from databases to simulation environments. Figure 5.2(a) shows the skeleton of a SBML document. The structure of the XML file is kept simple and can thus be used as an exchange format as well as a representation format to be formatted with simple XML editors. We will describe briefly the structure of SBML below.

Two upper hierarchies of the SBML file represent the elements identifying the type with the <sbml> tag and basic properties of the model such as the name embedded with the <model> tag. Compartment nodes, listed under the tag <listOfCompartments> represent virtual containers in which species are located. They do not necessarily correspond to structures related to the cell although this is often the case. The following fragment of SBML describes the definition of compartments:

```
<listOfCompartments>
    <compartment name="Cyt" volume="1.5" />
    <compartment name="Nuc" outside="Cyt" />
</listOfCompartments>
```

The required attribute for compartment is its unique name whereas optional attributes are its volume (default 1 unit) and outside which expresses relationships between compartments. Compartment nodes are optional in a SBML document and no compartment definition assumes that everything is located within a single compartment of unit volume. Species on the other side represent entities that take part in reactions such as molecules. Examples of species include molecules such as glucose or proteins. In the context of this thesis, specie and metabolite are used interchangeably. Below, an extract of a species definition in a sample SBML model is shown.

```
    <listOfSpecies>
    <species compartment="Cyt" initialAmount="3.48" name="G6P"/>
    <species compartment="Cyt" initialAmount="0.6" name="F6P"/>
    ...
<listOfSpecies>
```

Species elements have two required attributes, “name” and “initialAmount”

[1] http://www.sbml.org

```
1: <?xml version="1.0" encoding="UTF-8"?>
2: <sbml >
3: <model">
4: <listOfUnitDefinitions>
5: ...
6: </listOfUnitDefinitions>
7: <listOfCompartments>
8: ...
9: </listOfCompartments>
10: <listOfSpecies>
11: ...
12: </listOfSpecies>
13: <listOfParameters>
14: ...
15: </listOfParameters>
16: <listOfRules>
17: ...
18: </listOfRules>
19: <listOfReactions>
20: ...
21: </listOfReactions>
23: </model>
24: </sbml>
```

(a) SBML structure

```
1: <m3l>
2: <model name="...">
3: <listOfCompartments>
4: ...
5: </listOfCompartments>
6: <listOfSpecies>
7: ...
8: </listOfSpecies>
9: <listOfReactions>
10: ...
11: </listOfReactions>
12: <listOfKineticLaws>
13: ...
14: </listOfKineticLaws>
15: <listOfConstraints>
16: ...
17: </listOfConstraints>
18: <listOfModelParameters>
19: ...
20: </listOfModelParameters>
21: <listOfSplines>
22: ...
23: </listOfSplines>
24: <listOfMeasurements>
25: ...
26: </listOfMeasurements>
27: <listOfMeasureModels>
28: ...
29: </listOfMeasureModels>
30: </model>
31: </m3l>
```

(b) M3L structure

Figure 5.2: SBML and M3L structure

and several optional attributes, such as, for example, "compartment" which indicates the compartment within which the species is located. This attribute can be omitted only if the model does not define any compartments.

Reactions represent some transformation, transport or binding process,

typically a (bio)chemical reaction, that can change one or more species, as described below. In SBML, reactions are defined using lists of reactant species and products, their stoichiometric coefficients, and kinetic rate laws. A short example of a reaction node definition is shown below.

The reaction node has a required attribute name, used to identify it in the

```
<listOfReactions>
    <reaction name="reaction1" reversible="false">
        <listOfReactants>
            <speciesReference species="X0" stoichiometry="1"/>
        </listOfReactants>
        <listOfProducts>
            <speciesReference species="S1" stoichiometry="1"/>
        </listOfProducts>
        <kineticLaw formula="k1 * X0">
            <listOfParameters>
                <parameter name="k1" value="0"/>
            </listOfParameters>
        </kineticLaw>
    </reaction>
    ...
</listOfReactions>
```

Figure 5.3: Reaction definition in SBML

model, one optional attribute indicating whether the reaction is reversible or not (default true) and one optional attribute "fast" (default false) indicating a reaction which proceeds very fast and can thus be treated as quasi-stationary. Information about reversibility is useful in certain kinds of analyses such as elementary mode analysis. Furthermore, the reactants (sometimes called educts) and products, together with their stoichometric numbers of a reaction are identified by the proper "specieReference" nodes inside "listOfReactants" or "listOfProducts".

The optional "kineticLaw" element provides a mathematical formula describing the rate at which the reactants combine to form the products. It is optional although there is no useful default value for this element, but certain kinds of network analysis (e.g. elementary mode analysis) are still possible. The "kineticLaw" element has one required attribute, "formula", of type string, that expresses the rate of the reaction. The "kineticLaw" tag was modified in SBML level 2 in order to be able to define MathML expressions instead of strings present in level 1 for kinetic laws.

The elements of SBML introduced so far fully define the structure of a metabolic network. However, SBML offers the possibility to insert other ele-

```
<kineticLaw>
    <math xmlns="http://www.w3.org/1998/Math/MathML">
        <apply>
            <times/>
                <ci>
                    k1
                </ci>
                <ci>
                    X0
                </ci>
        </apply>
    </math>
    <listOfParameters>
        <parameter id="k1" value="0"/>
    </listOfParameters>
</kineticLaw>
```

Figure 5.4: Kinetic law definition in SBML using MathML

ments which could be helpful in different situations in simulation. The details of these other elements can be taken from the SBML user manual.

5.3.2 M3L

The simulation tool used in the framework of the project including this thesis, MMT [HFTW05], makes use of a SBML dialect called Metabolic Modeling Language (shortened M3L). The reason why it is a dialect and not pure SBML lies in the fact that SBML does not support the inclusion of rapid sampling results or other experimental data into the simulation model, which could be used as a criteria to evaluate the simulation, especially when more than one model i.e. model variants, are used.

Figure 5.2(b) shows the structure of M3L and the original structure of SBML. The elements "listOfCompartments", "listOfSpecies" and "listOfReactions" were taken from SBML. Furthermore, kinetic laws which in SBML were stored under "listOfReactions" were promoted one level higher to reduce redundancy, because they are often repeated especially in model variants. Thus, a reaction in M3L is defined as shown in Figure 5.5.

The kinetic laws are defined separately as presented, in Figure 5.6.

Constraints (see Figure 5.3.2) are defined in order to exclude variants, which are feasible but not wanted.

A further development of M3L compared to SBML is the ability to define measurements and splines, which are used later to help in the estimation of

```
<listOfReactions>
    <reaction value="v1">
        <listOfReactants>
            <specieRef compartment="X" specie="A"/>
        </listOfReactants>
        <listOfProducts>
            <specieRef compartment="X" specie="B"/>
        </listOfProducts>
        <listOfKinetics>
            <kineticLawRef kineticLaw="null">
            </kineticLawRef>
            <kineticLawRef kineticLaw="mm">
                <parameter symbol="k m" value="1"/>
                <specieLink specie="A" symbol="S0"/>
            </kineticLawRef>
            <kineticLawRef kineticLaw="mm 1inh">
                <parameter symbol="k m" value="1"/>
                <parameter symbol="k I" value="1"/>
                <specieLink specie="A" symbol="S0"/>
                <specieLink specie="C" symbol="S1"/>
            </kineticLawRef>
            <kineticLawRef kineticLaw="mm 1inh">
                <parameter symbol="k m" value="1"/>
                <parameter symbol="k I" value="1"/>
                <specieLink specie="A" symbol="S0"/>
                <specieLink specie="D" symbol="S1"/>
            </kineticLawRef>
        </listOfKinetics>
    </reaction>
</listOfReactions>
```

Figure 5.5: Reaction definition in M3L

parameters of the model or to reduce the complexity of the model. For this purpose, the element "listOfMeasurements" (Figure 5.8) is defined, which contains information about the time point, the value and the standard deviation of the respective species.

The splines are defined similarly under the element "listOfSplines" as given in Figure 5.9. Splines are mainly used to reduce the complexity of the model by inserting splines representing pre-processed measured data into the model, which serve the purpose of reducing the noise in the measured data.

Another change with respect to SBML is represented by the extended

```
1: <listOfKineticLaws>
2:     <KineticLaw name="mm" formula="v m*S0/(S0+k m)"/>
3:     <KineticLaw name="mm 1inh" formula="v m*S0/(S0+k m)*1/(1+k I)">
4:         <extends name="mm"/>
5:     </KineticLaw>
6: </listOfKineticLaws>
```

Figure 5.6: Kinetic law definition in M3L

```
1: <listOfConstraints>
2:     <constraint name="c1">
3:         (r.v1==k.null XOR r.v6==k.null)
4:     </constraint>
5: </listOfConstraints>
```

Figure 5.7: Constraints definition in M3L

```
1: <listOfMeasurements>
2:     <measurement name="mDAHP">
3:         <measure t="-3.942" value="0.0117749" stddev="6.6e-5"/>
4:         <measure t="-3.708" value="0.0471655" stddev="0.0004314"/>
5:         <measure t="-3.477" value="0.0379831" stddev="0.0003058"/>
6:         <measure t="-3.246" value="0.042373" stddev="0.0003632"/>
7:         <measure t="-3.012" value="0.0464297" stddev="0.0004205"/>
8:         <measure t="-2.781" value="0.0438399" stddev="0.0003834"/>
9:         ...
10:    </measurement>
11:    ...
12: </listOfMeasurements>
```

Figure 5.8: Measurements definition in M3L

attribute set for species in M3L (in Figure 5.10) which serve the purpose of distinguishing which metabolites are connected with measured data and which metabolites take part in ODEs.

The last modification in M3L is represented by model parameters which define global parameters such as time of simulation, boundaries for flux rates and metabolite concentrations, etc. An extract of the definition of model parameters is shown in Figure 5.11.

```
1: <listOfSplines>
2:     <spline name="spl ADP" t0="-3.942">
3:         <break tr="-3.708" degree="3">
4:             <coef c="0" value="0.119826413412"/>
5:             <coef c="1" value="0.0159524499127"/>
6:             <coef c="2" value="0"/>
7:             <coef c="3" value="0.195012665949"/>
8:         </break>
9:         ...
10:    </spline>
11:    ...
12: </listOfSplines>
```

Figure 5.9: Splines definition in M3L

```
1: <listOfSpecies>
2:     <specie compartment="Cyt" name="DAHP" initialAmount="0.04" measure="mDAHP"
min="0" max="0.2"/>
3:     <specie compartment="Cyt" name="ADP" from data="spl ADP"/>
4:     <specie compartment="Cyt" name="Phe" initialAmount="20" fixed="1"/>
5:     <specie compartment="Cyt" name="EPSP" initialAmount="0.2" min="0" max="5"/>
6: ...
7: </listOfSpecies>
```

Figure 5.10: Species definition in M3L

```
1: <listOfModelParameters>
2:     <defaultValue flux max="10" flux min="-10"/>
3:     <dilutionRate value="0"/>
4:     <resultSamplePoints begin="-3.0" end="20" interval="0.1"/>
5: <listOfModelParameters>
```

Figure 5.11: Definition of global model parameters in M3L

5.4 Customizable XML Comparison

In the following, we will motivate why a customizable XML comparison technique is needed. Then, CustX-Diff, the customizable tree edit distance for XML documents is presented, which extends the normal tree edit distance with the ability to specify XPath lists for the elements of XML documents. These XPath expressions will specify which parts of the XML trees will be compared against each other. Then, a similar approach for path based dis-

tances is presented. For both approaches, some experimental results will be given.

5.4.1 Motivation

Section 5.2 presented an overview of the techniques for comparing semi-structured data, especially XML data. However, the presented techniques focus either on the computation of a generic similarity measure (as a value) or on the calculation of edit distances between trees. The techniques are generic, i.e. they do not incorporate any previous information on the structure of the document and do not make any assumptions about the semantics of the data. However, in real data sets, important information for measuring the difference between XML files often, is mixed with unimportant one in the same XML file. Since XML elements are treated all the same in the existing approaches when measuring the difference between two documents, unimportant elements would also affect the difference, which may be an unwanted effect. To illustrate this with an example, let us consider two imaginary documents, as shown in Figure 5.12. Both represent a book with the same information up to the extract element which is different in the two documents. A XML difference algorithm would answer the question whether the two book elements are the same negatively, due to the influence of the extract element.

```
1: <books>
2:     <book>
3:         <author>
4:             Author A
5:         </author">
6:         <title>
7:             Title A
8:         </title>
9:         <extract>
10:            Extract 1
11:        </extract>
12:    </book>
13: </books>
```

(a) First document

```
1: <books>
2:     <book>
3:         <author>
4:             Author A
5:         </author">
6:         <title>
7:             Title A
8:         </title>
9:         <extract>
10:            Extract 2
11:        </extract>
12:    </book>
13: </books>
```

(b) Second document

Figure 5.12: Two XML documents

However, there are cases when we would like to simply ignore specific nodes or attributes of elements in different levels of the XML document, thus measuring a *constrained difference* between the two documents. For example, we would like to ignore the node "extract" in both documents.

To illustrate the idea further, we consider the structure of SBML files. A SBML file represents a metabolic model, i.e. it represents both the network structure of the metabolic network and the model parameters which serve to build simulations of this metabolic network. The same is valid for M3L documents. When comparing SBML files, two kinds of questions can be posed, namely are the structures of the respective metabolic networks the same or are the models represented by the SBML files the same. For the latter, we need a full comparison of SBML documents, whereas for the former a constrained XML comparison algorithm, which considers only the elements under $listOfSpecies$ and $listOfReactions$ is needed. From the illustrations above it becomes clear that two potential possibilities come into question for every element: to include or not to include the element during the difference process, thus creating filters with the to-be-included and not-to-be-included elements. After defining these filters, this difference process can proceed in three different ways:

- by customizing the tree edit distance to assign zero weight to the respective edit operations;
- by first filtering the two XML trees and then doing the comparison;
- by modifying path based distances to measure the similarity.

These techniques are illustrated in detail in the following subsections.

5.4.2 Formalization of the Problem

XML documents basically contain three kinds of nodes when represented as a DOM tree:

- **Element** nodes which are named non-leaf nodes
- **Text** nodes which are leaf nodes with a value
- **Attributes** which are leaf nodes containing both a name and a value

Comments, processing instructions and namespaces are not considered in the context of comparing XML documents. Although attributes are unordered and element nodes are ordered in the DOM specification, in many cases such as in SBML the order of element nodes is not important. For this purpose, we have extended the X-Diff algorithm presented by Wang et al. [WDC03]. Also some notation is borrowed from the same paper.

Three elementary and two composite edit operations are defined on trees, namely:

- Insert($x(name, value), y$) inserts a leaf node x as a child of y. The position is not important since the tree is unordered.
- Delete(x) deletes a leaf node x.
- Update($x, newvalue$) changes the value of the leaf x to *newvalue*.
- Insert(T_x, y) insert a tree rooted at x as child of y and represents a sequence of Insert operations.
- Delete(T_x) deletes the subtree rooted at x and represents a sequence of Delete operations.

Sequences of these operations can be used to transform a tree into another. Since there are many such sequences, the concept of minimum cost edit script is introduced. By assigning costs to each of the elementary operations (e.g. one to each of them), each sequence will have an overall weight.

Definition. E is a minimum cost edit script for transforming the tree T_x into T_y iff $\forall$ edit script E' which transforms T_x into T_y, cost(E')$\geq$cost(E). The edit distance between T_x and T_y is defined as cost(E).

Two more notions are needed before proceeding further. The signature of a node is defined as the modified path from the root to that node. In general, it has the form Signature(x) = /Name(x_1)/.../Name(x_n)/Name(x)/Type(x) where (x_1,...x_n,x) represents a path from *root* to x. For text nodes, it is reduced to /Name(x_1)/.../Name(x_n)/Type(x).

Furthermore, for each subtree, a hash value, XHash, similar to the DOMHash [MTU00] is calculated. An equal value of XHash indicates with high probability that the two compared subtrees are equal to each other.

So far, the formalities dealt with the generic edit distance between XML documents. To customize the comparison process, means for defining the parts of XML which are to be excluded from the comparison process are needed. To achieve this purpose, two techniques are used, namely:

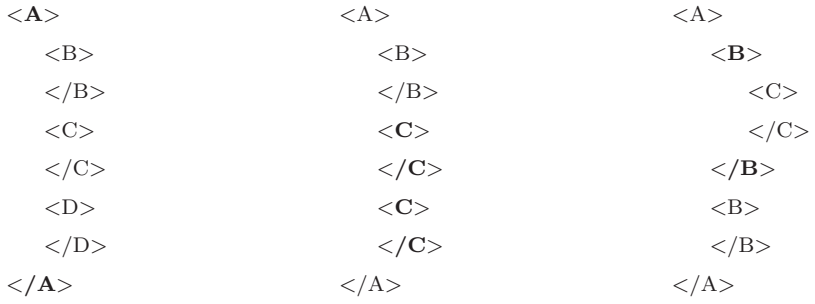
- by specifying complete paths of the parts of XML documents not to be considered
- by specifying XPath expressions for the same purpose, allowing more flexibility.

The first approach works by defining complete paths of the form $/a/b/c$, which implies that every part of the XML document whose signature starts with $/a/b/c$ will be omitted from the comparison process. This approach allows a simple but efficient way of excluding nodes from the comparison process because the verification of the fact that the signature fulfills the defined condition can be accomplished by checking if the predefined path $/a/b/c$ is prefix of the signature of the treated part of the XML document. However, more complicated expressions are beyond the capabilities of this simple approach. For this purpose, XPath expressions are defined, which specify the parts of XML document to be omitted. But by increasing the capability of the filtering expressions, we also increase the complexity of verification if a certain signature of a part of XML document is contained in a predefined XPath expression. A short XPath introduction and the containment problem for XPath expressions are presented in the following paragraphs.

5.4.2.1 Short Description of XPath

XPath is a technology which allows to address certain parts, i.e. nodes of XML documents, in a manner similar to the Unix file systems. The matching of nodes in a XML document is done via XPath expressions, whose evaluation results can be a set of nodes, a boolean variable, a number or a string. The specification of an XPath expression defines which nodes of the document are requested while the question of which algorithm should be used to find these nodes is left open. Some of the examples below illustrate the results of interpreting XPath expressions in their long and short form in simple XML documents. Figures 5.13(a) and 5.13(b) show the result of the interpretation of the simplest XPath expressions, namely the selection of nodes using an XPath expression without wild characters. Figure 5.13(c) illustrates a slightly complicated expression, which selects nodes B, descendants of A, that have a son C.

Figure 5.14(a) illustrates the selection via a XPath expression of text information contained in a certain part of the XML document. So far, the queries involved nodes on a certain level. Figure 5.14(b) illustrates XPath expressions which involve results occurring on different levels of the XML document, namely elements with the name C in every level or elements with any name, as in Figure 5.14(c). The last group of figures illustrates more complex XPath expressions. Thus, in Figure 5.15(a) all daughter nodes of the node $/A/B$ including the node itself, result from the interpretation of the XPath expression $/A/B//*$. Figure 5.15(b) and 5.15(c) show the backwards filtering ability of XPath expressions. A further functionality of XPath is represented by the ability to move not only down and up in the document tree, but also sideways, allowing to specify expressions such as the 3^{rd} B-named child of a certain element and so on.

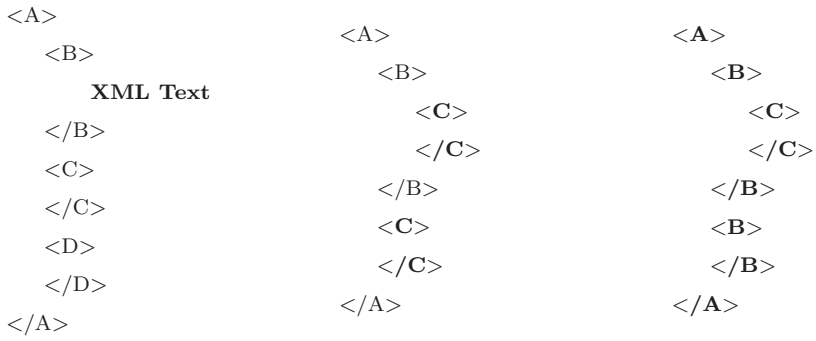


(a) XPath expression /A (long form /child::A)

(b) XPath expression /A/C (/child::A/child:C)

(c) XPath expression /A/B[C] (/child::A[child::C])

Figure 5.13: Results of XPath expressions emphasized in bold (1)



(a) XPath expression /A/B/text() (/child::A/child::B/child::text()

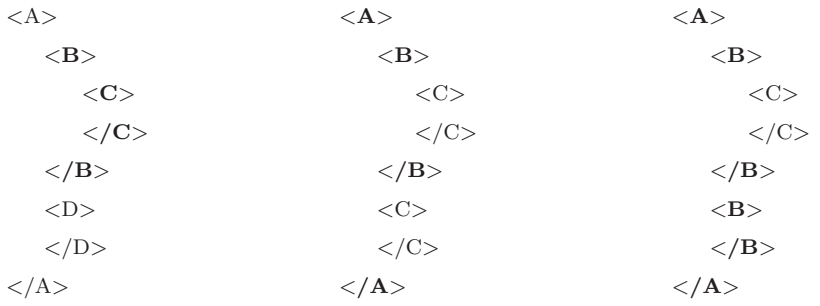
(b) XPath expression //C (/descendant::C)

(c) XPath expression //* (/descendant::*)

Figure 5.14: Results of XPath expressions emphasized in bold (2)

5.4.2.2 Containment of XPath Expressions

After having defined a list of XPath expressions which are to be omitted from the XML comparison process, an efficient algorithm for checking whether elements of XML documents are in the list of exclusions is needed. The algorithm receives as an input an XPath expression and a signature, i.e. path of an element in the XML document, and check whether the signature is contained in the XPath expression. A more difficult problem, namely the containment problem for two XPath expressions, was treated by Miklau



(a) XPath expression /A/B//* (/child::A/child::B/descendant-or-self::*

(b) XPath expression //C/parent::*

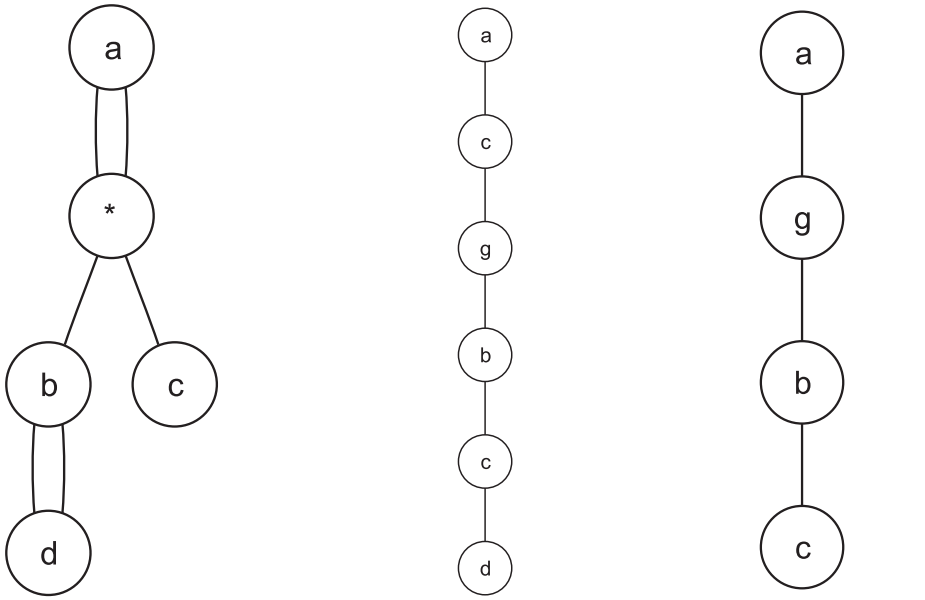
(c) XPath expression //B/ancestor-or-self::*)

Figure 5.15: Results of XPath expressions emphasized in bold (3)

et al. [MS04]. Furthermore, Miklau et al. have introduced an algorithm for efficiently checking the embedding problem between XPath expressions (in fact for a subclass of them called Tree Patterns) and trees. Based on this algorithm, and using the characteristics of our problem, we developed a dynamic programming approach for the embedding problem for XPath expressions and a signature. We consider the Tree Patterns subset of XPath which is commonly defined $X^{\{[],*,//\}}$. Figure 5.16 shows the tree representing XPath $a//*[b//d][c]$. The tree has two kind of edges: child edges represented by one line and descendant edges represented by two parallel lines. We use XPath to customize and thus filter parts of XML documents. As such, we use XPath in a slightly other context compared to the original proposal. Thus, the XPath expression given above would return nodes $*$ which are descendants of a, and have two children named b and c. The child b must have a descendant named d. We use XPaths to specify the tree pattern we would like to filter from the XML document. In this way, we are not interested anymore in the node $*$, but in the leaf nodes of the tree pattern. We could have restricted the filtering to one subset of tree patterns with $X^{\{*,//\}}$, but this would have been more restrictive on the user. In this way, we allow for complicated filters with a single expression.

5.4.3 CustX-Diff

In this subsection, we present an algorithm called CustX-Diff for detecting exact changes between parts of SBML files. Our algorithm builds on X-Diff [WDC03], a published XML change detection algorithm. X-Diff can



(a) XPath tree pattern (b) Embeddable path (c) Embeddable path

Figure 5.16: XPath tree patterns and two paths

detect the optimum differences between two unordered XML documents in polynomial time. It works for any type of XML data. For reasons explained in the previous sections, our change detection process is based on deciding for each XML element whether it should be included in the comparison or not.

To achieve this purpose, XPath expressions (XPEs) are used to specify filters for the XML documents to be compared. The XPEs we consider are the *tree patterns*, commonly defined as $X^{\{[],*,//\}}$. Note that we use XPEs only to address the XML document for obtaining filtered subtrees, and not to obtain a set of nodes. Furthermore, if a parent node is embeddable, all its children nodes are considered embeddable, too.

The pseudocode of CustX-Diff is given in Algorithm 5.2. It differs from the original X-Diff in that it includes the above described filtering in the comparison process. Thus, before performing the matching of nodes (line 10 of the algorithm), the considered nodes are checked to determine whether they fulfill the filtering conditions.

The same effect can be achieved by filtering the XML documents either during parsing of the XML documents or using an Aspect Oriented Programming (AOP) approach. For the latter, more details can be found in [QF06].

Algorithm 5.1: Matching of paths with XPath expressions

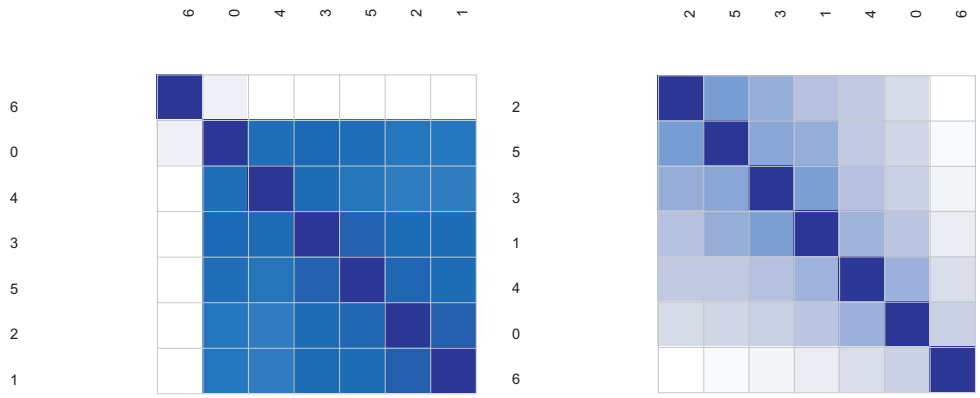
Input: Two strings p and q representing XPath expressions and an XML path, respectively
Input: An integer value indicating whether a simple comparison or an embedding algorithm is to be applied
Output: True if the path is matched against XPath expression, false otherwise

```
1  Parse p into pattern tree P and q into tree T;
2  do a postorder traversal of P and store the nodes in postP:
3  do a postorder traversal of T and store the nodes in postT:
4  for x ∈ postT do
5      for y ∈ postP do
6          childX=x.child;
7          descendantMatched=false;
8          if y.descendants=null then
9              descendantMatched=true;
10         end
11         for z ∈ y.descendants do
12             if D[z][childX] then
13                 descendantMatched=true;
14             end
15         end
16         childMatched=false;
17         if y.children=null then
18             childMatched=true;
19         end
20         for z ∈ y.children do
21             if C[z][childX] then
22                 childMatched=true;
23             end
24         end
25         C[x][y]= ((label(x)==label(y) || label(y)="*") &&
           descendantMatched && childMatched);
26         D[x][y]= (C[x][y] || D[childX][y]);
27     end
28 end
```

5.5 Evaluation of CustX-Diff

The first set of XML documents CustX-Diff is evaluated on a set of SBML documents consisting of seven metabolic models of the Valine/Leucine path-

way in C. glutamicum [MHOT06]. The models are derived from each other iteratively beginning with a small model (i.e. a small number of reactions) and increasing it in each step by adding new reactions to the model. The XML files contain all the information needed for the simulation as described in Section 5.3, such as the structure of the network, parameters, splines, etc. Figure 5.17 shows the distance matrices visualized using the methods introduced in Chapter 6. The darker the color of the respective cell, the greater the similarity between the respective model pair.



(a) XDiff distances (b) CustXDiff distances

Figure 5.17: Matrix of edit distances between a set of models derived from each other incrementally

Figure 5.17(a) presents the results of X-Diff and Figure 5.17(b) presents the results of CustX-Diff. CustX-Diff considers only the topology of the models, which is encoded inside the tag *listOfReactions*. The distances generated from X-Diff in Figure 5.17(a) do not give a clear information about the structural differences of the models because the rest of model definition affects the difference operation. It just shows that model number 6 is different from the rest of models.

In contrast, Figure 5.17(b) reflects much better the structural differences in the list of metabolic networks. After reordering the matrix (with techniques presented in Section 6.7), the sequence these models are derived from each other becomes clear (6, 0, 4, 1, 3, 5, 2).

Figure 5.18 shows the time in milliseconds needed to complete the pairwise comparisons (a total of 21) between the seven models. The time required by CustX-Diff is nearly constant. It is sometimes higher than the time X-Diff requires to perform the difference operation, which is due to the overhead introduced by the verification process to check if a certain part of the XML document is included in the filter or not.

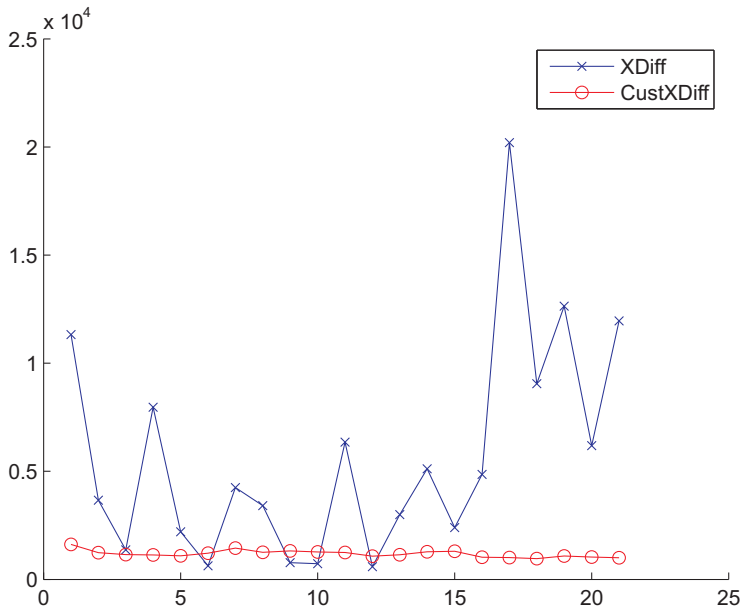


Figure 5.18: Time in milliseconds X-Diff and CustX-Diff require to complete the pairwise comparisons between seven models, whose edit distances are shown in Figure 5.17(b)

For further evaluation of CustX-Diff, a test set consisting of 90 XML documents in SBML format was selected from the Kyoto Encyclopedia of Genes and Genomes (KEGG) database [KG00]. The models selected in this case represent the Glycolysis/Gluconeogenesis pathway for 90 different organisms, in contrast to the above example which represented derivatives of the same pathway for the same organism. The topology of the graph describing the metabolic network is again encoded inside the tag *listOfReactions* and to compare two topologies with each other, we do not need to compare entire SBML documents, but just the hierarchy which is encoded between the opening and closing *listOfReactions* tag. The edit distances between the models were evaluated using both X-Diff and CustX-Diff algorithms.

Figure 5.19 presents the matrix of pairwise edit distances for the 90 models, labelled as they are indexed in KEGG. The respective matrix of X-Diff distances is not shown because as in the previous example, it could not be used to extract any visual pattern. The matrix of CustX-Diff distances reveals clear patterns of similar models (again after the reordering of columns and rows), as for example the family of E. coli strains which is highlighted in Figure 5.19. Similar patterns can be observed for other groups of organisms.

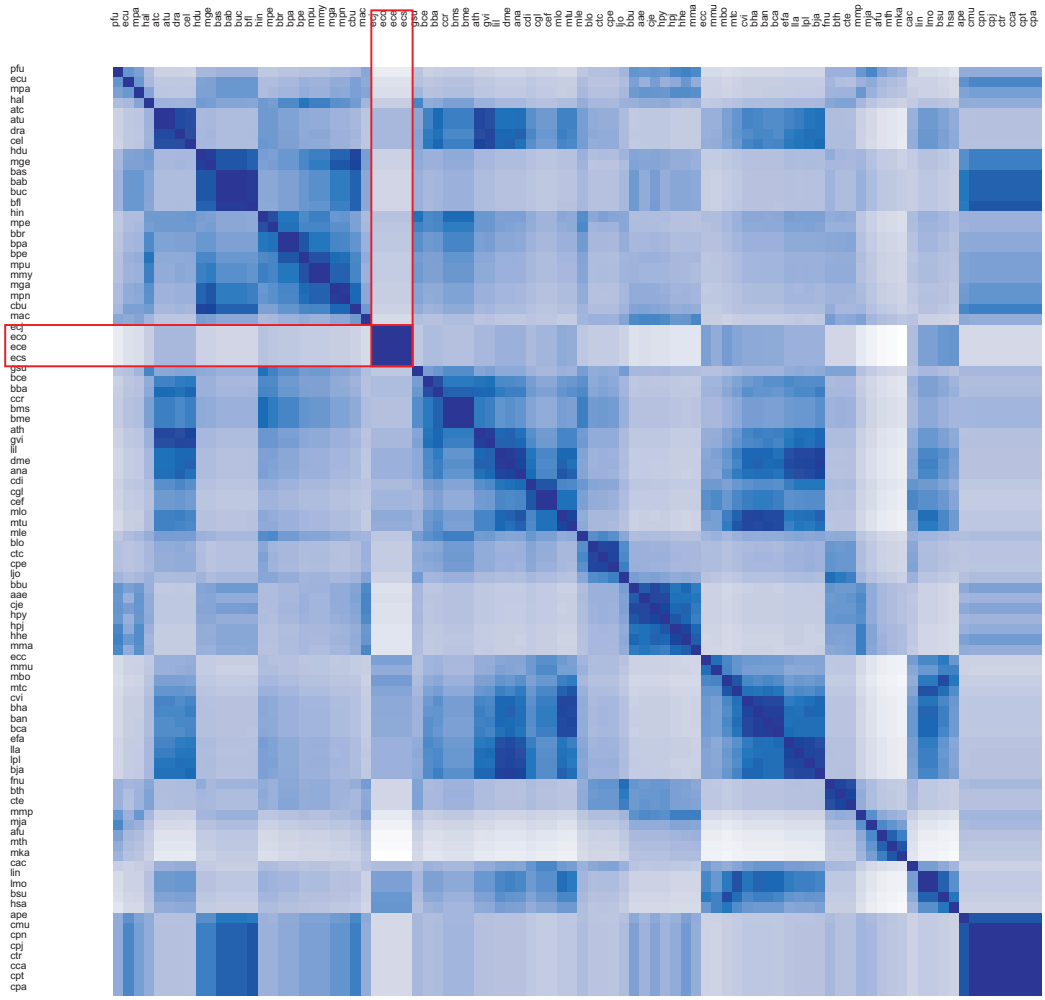


Figure 5.19: Matrix of edit distances between the same metabolic pathway for different organisms taken from KEGG [KG00]

5.6 Summary

In this chapter we presented CustX-Diff, a customizable change detection approach to compare metabolic networks saved as SBML or M3L files. CustX-Diff performs a tailored change detection operation which filters XML document parts according to specified XPath expressions. The subset of XPath expressions used allows the specification branching, wildcards and descendant relationships between nodes. The evaluation of CustX-Diff confirmed, as expected, lower edit distances between XML documents compared to X-Diff. As such, it allows potential human and machine users to focus the comparison process on specific parts of the documents.

Algorithm 5.2: Customizable XML Diff

Input: XML Documents D_1 and D_2, Lists of constraints L_1 and L_2
Output: Constrained edit script E which considers the lists above

```
1  Parse D1 into T1 and hash T1 and parse D2 into T2 and hash T2;
2  if HashValue(root(T1))==HashValue(root(T2)) then
3      return empty edit script; /* The XML files are equal        */
4  else
5      N1={set of leaves of T1}, N2={set of leaves of T2};
6      Filter out subtrees that have equal XHash values or belong
       respectively to list of constraints L1 or L2;
7      repeat
8          foreach x ∈ N1 do
9              foreach y ∈ N2 do
10                 if x ∉ L1 and y ∉ L2 and Signature(x)==Signature(y)
                   then
11                     Compute and save in dist the distance(x,y);
12         N1={set of parent nodes in N1}, N2={parent nodes in N2};
13     until N1 and N2 are not empty;
14     if HashValue(root(T1))==HashValue(root(T2)) then
15         Mmin=empty; /* The roots are different;              */
16     else
17         add the pair {root(T1, root(T2)} in Mmin;
18         foreach (x, y) ∈ Matching do
19             add matching to Mmin
20     x=root(T1), y=root(T2); Insert (x,y) into empty queue Q;
21     while Q not empty do
22         extract (x,y) from queue Q;
23         if (x, y) ∈ Mmin then
24             return "Delete T1, Insert T2"
25         foreach (xi, yi) ∈ Mmin where xi child of x and yi child of y do
26             if xi and yj are not leaves then
27                 add (Txi, Tyj) into the queue;
28             else if dist(xi, yj)!=0 then
29                 add Update(xi, value(yj)) in E;
30         foreach xi ∉ Mmin do
31             Add Delete(xi) in E;
32         foreach yj ∉ Mmin do
33             Add Insert(yj) in E;
```

6
Visualization of Sensitivity Matrices

6.1 Introduction

To visualize high-dimensional data sets, two issues need to be dealt with:

- Large dimensionality of data (which might be moderate in size)
- Large size of data

In both cases, the purpose of visualization is to optimally use the available screen space for conveying as much information as possible to the user. This task becomes challenging in the case of high-dimensional data sets encountered during the simulation of metabolic networks, such as the sensitivity matrices introduced in Section 2.3.2. In contrast to *static* high-dimensional data, sensitivity matrices are dynamic in nature, meaning that they evolve with the time. These sensitivity matrices indicate how the changes in the parameters of a metabolic network model affect the output of the model. Their analysis serves two purposes, namely to find redundancies, which are then used to derive a simpler model with the same properties, and to find important parameters, which could be adequately changed to achieve specific effects in the output of the model.

The focus of this chapter is on techniques for the visualization of static and *dynamic* multi-dimensional data (i.e. data that changes over time) and their application for the visualization of sensitivity matrices. Two problems can be distinguished in this case: (a) finding patterns/structures locally, i.e for static data or for a single point of time and (b) finding patterns that persist for several points of time or globally for all points of time.

In the framework of this dissertation, a toolkit called MatVis (Matrix Visualizer) has been developed as a visualization environment to offer solutions to these problems. It consists of several visualization methods, which represent different techniques for approaching high-dimensional data; used together,they provide multiple coordinated views to the user. The techniques provided in this toolkit include:

- Dimension reductioning techniques such as *multi-dimensional scaling* and Sammon mapping. These techniques are extended and combined with an interactive version of the K-Means clustering algorithm to allow the direct exploration of the clustering results in the multi-dimensional scaling view.

- Techniques such as the *colored reorderable matrix*, which allows the visualization of multi-dimensional data by mapping numerical values to colors, gray-scale values or symbols, which are then ordered automatically or interactively with the help of the user. Different algorithms for their ordering are introduced in this chapter. Furthermore, we explore different techniques for adapting the reorderable matrix method for time-varying data as well as algorithms for searching for similar reorderings, i.e. permutations of the columns which are consistent within certain time ranges.

- Novel visualization methods for large covariance/correlation matrices are introduced. The *reorderable covariance/correlation matrix* view allows the interactive visualization of the covariances and correlations dynamically as the time-varying sensitivity matrices are explored.

- The novel colored cluster membership matrix, represents the timely evolution of cluster memberships for the objects clustered by any partitional clustering algorithm (such as K-Means or Fuzzy C-Means). It is based on calculating a cumulated adjacency matrix that gathers information regarding the membership of objects in clusters for each point of time. By examining the color visualization of this matrix, changes in cluster memberships and possible outliers, i.e. objects that change clusters frequently, can be extracted. Furthermore, groups of objects which belong to the same cluster for a certain number of points of time can be distinguished.

For all the above mentioned techniques, different algorithmic and interaction aspects are discussed in detail in the course of this chapter.

The work presented in this chapter has been partially published in several papers [QWF04b, QWF04c, QWF05b, QWF05a].

The chapter is organized as follows. Section 6.2 gives a survey of related work in the field. Section 6.3 describes the input data in the context of

sensitivity analysis in metabolic modeling. Section 6.4 presents the MatVis toolkit, which includes the visualization techniques presented in this chapter. Section 6.5 focuses on the asymmetrical reorderable matrix, in its static and time varying variants and the related problems. Section 6.6 describes techniques related to low-dimensional projection of sensitivity matrices for purposes of visualization. Section 6.7 is concerned with reorderable matrix derivatives for proximity data and the related reordering problems. Section 6.9 concludes the chapter.

6.2 Related Work

Depending on which visual variables (see Chapter 3 for a definition of visual variables) they use, on how the screen space is used and the transformation that data undergoes, the techniques for visualizing high-dimensional data can be broadly categorized as:

- Dimensional subsetting, such as scatterplot matrices [CM88].
- Dimensional embedding techniques, e.g. dimensional stacking [LWW90].
- Axis reconfiguration techniques, such as parallel coordinates [ID90].
- Icon based techniques, such as Chernoff faces [Che73].
- Dimensional reduction techniques, such as multi-dimensional scaling, principal component analysis , and self-organizing maps [YH38, Sch35, Mar79].
- Tabular visualization techniques, such as the reorderable matrix and its derivatives [Ber81].

Some of the techniques focus on certain types of data; e.g. dimensional stacking is used either for discrete data or for discretized continuous data. Thus, in this section, we will focus on existing approaches that are relevant in the context of this chapter. The techniques described below are roughly divided into two categories: (1) simple visualization techniques, i.e. pure approaches which include only one technique for visualizing data and (2) combinations of visualizations techniques, which use several techniques for visualizing data, profiting from the advantages each one has to offer.

The reorderable matrix method proposed by Bertin [Ber81, Ber83] is a simple but robust approach to visualize tabular data. The main idea of the reorderable matrix method is to convert a tabular data set into an interactive 2D view. The 2D view has the same dimensions as the original data, and according to Bertin the data size should not exceed dimensions

of $X \times Y = 10000$, where X and Y represent the dimensions of the rows and columns of data. Data values are replaced by filled circles whose size depends on the actual value. With manual interaction or automatic permutations, different patterns in the data are made visible. Minnotte et al. [MW98] use reorderable matrices under another name, data image, to explore high-dimensional data. Marchette et al. [MS03a] use data images for outlier detection in data. Corrgrams proposed by Friendly [Fri02] is an approach similar to the reorderable matrix method to visually explore correlation matrices which are important in multivariate statistics. Bezdek and Hathaway [BH02] use an approach called ODI (Ordered Dissimilarity Image) for visually clustering data. Ghoniem et al. [GFC04] use an approach similar to the reorderable matrix in a different context, namely to compare the usual graph visualization approach with a matrix-based representation. However, the reordering of the matrix based representation is not considered, since it aims to assess the difference between the two representations. In [Sii03], Siirtola combines parallel coordinates with reorderable matrices [Ber81, Ber83] to visualize multi-dimensional data.

Dimension reduction is another alternative to visualize multivariate data in 2D or 3D. There are two types of dimension reduction techniques: linear and nonlinear. Principal component analysis (PCA) is a linear projection method where the projection is formed as a linear combination of the input. Multi-dimensional scaling (MDS) and the Sammon Mapping are two related nonlinear projection methods, with the former method preserving large distances and the latter preserving small distances. A survey of these techniques can be found in [Fod02].

Parallel coordinates introduced by Inselberg and Dimsdale [ID90] allow visualizing multi-dimensional data in parallel axes and is one of the popular methods used in multi-dimensional data visualization. Each dimension in this method corresponds to an axis; the axes themselves are organized as uniformly spaced horizontal (sometimes vertical) lines. A point in the high-dimensional space is represented as a line connecting points on each axis. To visualize massive data sets, different techniques such as brushing or sampling are used to reduce the clutter effect. Andrews curves [And72] is a visualization method similar to parallel coordinates based on a transformation similar to a Fourier transformation. Glyphs [Lit83] are graphical entities that convey the information present in the data set using attributes such as shape, size, color, and position. Chernoff faces [Che73] represent multi-dimensional data by means of faces with changing attributes. Thus, the problem of finding similar vectors is converted into the problem of finding similar faces. Star glyphs, where axes are arranged as radiating lines in equal angular distance, in contrast to parallel coordinates, represent another glyph technique for visualizing multi-dimensional data.

So far, the techniques dealt with static multi-dimensional data. The techniques for visualizing time-varying data on the other side, usually deal with univariate or vectorial data. A survey of visualization techniques for time-dependent data is given by Müller and Schumann [MS03b], which focuses on the visualization of time series data. The techniques for visualizing time-varing data into several categories, namely:

- Updating of static states, where static visualizations are changed in a timely matter based on the time-varying data. Such techniques can also be enhanced with query mechanisms, which allow filtering by selecting time periods. Inside this group, the techniques differ from each other in the way the update of the states is made (with or without animation, steerable from the user or not, etc.).

- Time series plots, where time is represented with its own dimension and finds usage in stock market diagrams, EKG plots, etc.

- Overview based methods, which aggregate the data for a certain time period, to produce a unified visualization of the data. In this context, a large group consists of force-directed methods, which could be applied after the data has been transformed in the form of a similarity matrix. The methods can be interactive in their spirit, allowing the user to control the process of aggregation. For example, Groenen and Franses [GF00] use multi-dimensional scaling to visualize time-varying stock market correlations.

- Other methods, which extend the existing static metaphors for time-varying data, e.g. TimeHistograms by Kosara [KBH04] as an extension of standard histograms to visualize time-varying data.

6.3 Description of Input Data

Before explaining in detail the visualization techniques that are part of MatVis, the information contained in time-varying sensitivity matrices and their preprocessing are described.

6.3.1 Sensitivity Matrices

To illustrate the input data, i.e. the sensitivity matrices, a small example is introduced. The model presented visually in Figure 6.1 [Noa05], has 7 metabolites, where S_f is the input substrate, S, P_{1X}, P_{2X} are extracellular metabolites and A,P_1, P_1 intracellular metabolites. Furthermore, the model has 6 reactions named r_1, ..., r_5 and r_f.

The model is described by the system of differential equations in 6.1, as illustrated in Chapter 2. The parameters of the model are listed in Table 6.1. Even for such a small model, the number of parameters increases rapidly depending on the number of reactions and the complexity of the respective kinetic laws.

$$
\begin{aligned}
r_f &= D(S_f - S) \\
r_1 &= \frac{r_{max} \cdot S}{K_{mS}(1 + \frac{P_2}{K_{Ia}}) + S(1 + \frac{P_2}{K_{Ib}})} \\
r_2 &= \frac{r_{max}(A - \frac{P_1}{K_{eq}})}{K_A(a + \frac{P_1}{K_{mP}}) + A} \\
r_3 &= k_{diff}(P_1 - P_{1X}) \\
r_4 &= \frac{r_{max}(A - \frac{P_2}{K_{eq}})}{K_A(a + \frac{P_2}{K_{mP}}) + A} \\
r_5 &= k_{diff}(P_2 - P_{2X}) \\
\dot{S} &= r_f - v \cdot r_1 \\
\dot{A} &= r_1 - r_2 - r_4 \\
\dot{P_1} &= r_2 - r_3 \\
\dot{P_{1X}} &= v \cdot r_3 \\
\dot{P_2} &= r_4 - r_5 \\
\dot{P_{2X}} &= v \cdot r_5
\end{aligned}
\tag{6.1}
$$

Table 6.2 shows parts of the raw similarity matrix and its normed versions with respect to the metabolite concentrations and flux rates. These matrices are selected from the point of time $t = -5s$, when the system is in a stationary state and the metabolite concentrations are nearly constant. The parameters that have zero values for all metabolites and reactions are excluded from the table.

The values in the matrices indicate how the changes in the parameter values affect the concentrations of metabolites and reaction rates respectively. A positive value indicates an increase of the concentration (or reaction rate) for an infinitesimal increase in the parameter value and a negative value indicates a decrease. For absolute sensitivity values, the sensitivity indicates how much the concentration changes when the respective parameter changes by one unit. For normed sensitivities on the other side, the values indicate the percentage of change when the parameter value is increased one percent. For example, an increase of $1mM$ of $r1$ K_{mS}, would bring a decrease of $3.26mM$ in P_2, whereas an increase of 1% in the value of $r1$ K_{mS} would decrease 20%

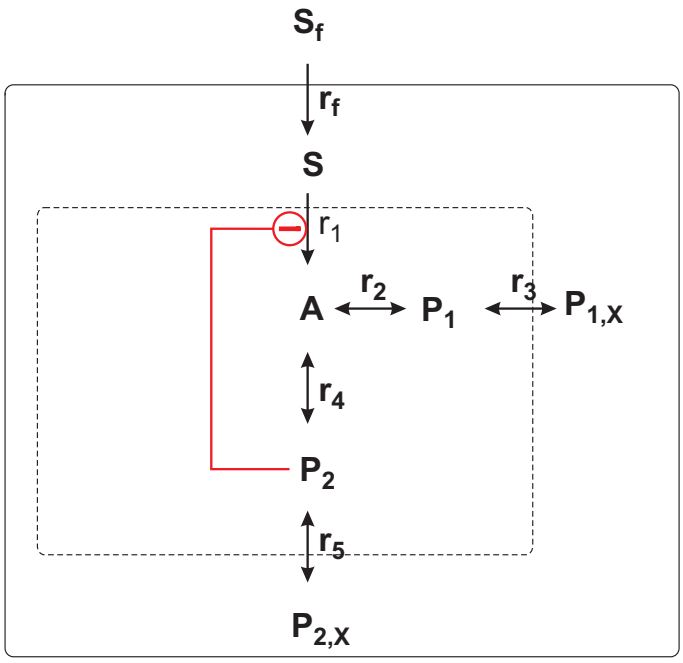


Figure 6.1: A sample metabolic model used to illustrate the visualization of sensitivities

Table 6.1: Parameters of the model in Figure 6.1

Reaction	Param.	Value	Unit
r_1	K_{Ia}	0.4	mM
	K_{Ib}	0.3	mM
	K_{mS}	0.05	mM
	r_{max}	10.0	mMs^{-1}
r_2	K_{eq}	5.0	
	K_{mP}	0.6	mM
	K_{mS}	0.01	mM
	r_{max}	1.0	mMs^{-1}
r_3	k_{diff}	0.6	s^{-1}
r_4	K_{eq}	2.0	
	K_{mP}	0.3	mM
	K_{mS}	0.1	mM
	r_{max}	0.6	mMs^{-1}
r_5	k_{diff}	0.05	s^{-1}

Param.	Value	Unit
S_f	100	mM
D	0.3e-4	s^{-1}
Initial Concent.		
S	0.05	mM
A	0.44	mM
P_1	0.95	mM
P_{1X}	0.00	mM
P_2	0.77	mM
P_{2X}	0.00	mM

the concentration of P_2. Parameters such as $r2$ K_{eq}, that control reactions developing in both directions, have a low absolute sensitivity and a high

Table 6.2: Sensitivity matrix for the time moment $t = -5s$

	r_1				r_2		r_3	r_4	r_5
	K_{Ia}	K_{Ib}	K_{mS}	r_{max}	K_{eq}	r_{max}	k_{diff}	K_{eq}	k_{diff}
Raw Sensitivity ($S^{(raw)}$)									
S	-0.01	-0.01	0.16	0	0	-0.01	-0.01	0	-0.02
A	0.15	0.06	-1.8	0.01	-0.05	-0.37	-0.44	-0.07	-0.55
P_1	0.15	0.06	-1.8	0.01	0.03	0.2	-1.26	-0.07	-0.48
P_{1X}	0.01	0	-0.12	0	0	0.01	0.02	0	-0.01
P_2	0.27	0.11	-3.26	0.02	-0.08	-0.55	-0.65	0.22	-2.5
P_{2X}	0	0	-0.02	0	0	0	0	0	0.05
Normalized with respect to metabolite concentrations ($S^{(m)}$)									
S	-0.49	-0.14	0.74	-0.93	-0.36	-0.49	-0.34	0.4	-0.11
A	0.13	0.04	-0.2	0.26	-0.61	-0.83	-0.59	-0.3	-0.06
P_1	0.06	0.02	-0.09	0.12	0.15	0.21	-0.79	-0.14	-0.03
P_{1X}	0.09	0.03	-0.14	0.18	0.19	0.26	0.29	-0.16	-0.01
P_2	0.14	0.04	-0.21	0.26	-0.51	-0.7	-0.49	0.55	-0.16
P_{2X}	0.17	0.05	-0.27	0.34	-0.36	-0.48	-0.31	0.43	0.89
Normalized with respect to flux rates ($S^{(f)}$)									
r_1	0.05	0.02	-0.07	0.09	0.07	0.09	0.07	-0.06	0.02
r_2	0.06	0.02	-0.08	0.11	0.14	0.19	0.15	-0.13	-0.03
r_3	0.06	0.02	-0.09	0.12	0.15	0.21	0.16	-0.14	-0.03
r_4	-0.01	0	0.03	-0.03	-0.74	-1.03	-0.73	0.75	0.7
r_5	0.14	0.04	-0.21	0.26	-0.52	-0.7	-0.5	0.55	0.84

relative one. For this reason, relative sensitivities are used to compare the different parameters with each other.

The values of the normed matrix $S^{(m)}$ indicate a distributed control of metabolite concentrations from the model parameters. The parameters of reactions affecting the metabolites have usually a higher sensitivity than the rest.

In analogy with the sensitivity analysis for the stationary state, the same strategy can be used to analyze the system in an instationary state. For this purpose, pulse experiments are carried out, as explained in Chapter 2, to analyze the interaction mechanisms inside the cell in several different states of the system. The sensitivity matrices in an instationary state are time-varying and their analysis is important for gaining knowledge about the regulation of the metabolic network. As such the sensitivity of the parameters changes with time. After the system is balanced again some of the parameters return to the old sensitivity values and some of them keep the changed sensitivities.

6.3.2 Scaling the Input Data

Assuming that the input data is a set of matrices for consecutive points of time, their storage in our case is accomplished by keeping them in a CSV (Character Separated Value) file. Some visualization techniques of those described below require that the data is scaled to a certain range, e.g. the segment [-1, 1]. For example, the colored reorderable matrix needs to map the values into a color spectrum and as such the values need to be mapped into a well known interval. Furthermore, such scaling must take into consideration that comparable visualizations over time need to be obtained. Two methods are considered in this context. The first one considers normalizing the values using the maximum value over all time points, parameters, and metabolites (or reaction rates). With this method, different orders of magnitudes in the sensitivity values for different metabolites mean that the lower order of magnitudes will not be considered properly. As such, the input data is exposed to a row-based normalization process in which the maximum norm (L_∞ norm for a vector $\vec{x}$ is defined as $\|\vec{x}\| = max_i|x_i|$) is used to normalize the data within all points of time. The maximum norm is appropriate in this case, because it can be calculated very fast. After normalization, the values of the matrices for all points of time lie in the segment [-1, 1]. Formally, the second method can be formalized as below:

$$\|x_i\|_\infty = max_k max_j(S^M_{i,j}(t)), i = 1..m, j = 1..n, t = 1..T \quad (6.2)$$

$$S^M_{(scaled)} = diag(\|x_i\|_\infty)^{-1} \cdot S^M \quad (6.3)$$

6.4 The MatVis Toolkit

The MatVis visualization toolkit consists of several coordinated views that interact with each other. These views are selected so that they are as "orthogonal" as possible to each other for bringing different perspectives of data to the user. Figure 6.2 shows a screenshot of the MatVis GUI with some of the provided views. Basically, MatVis is composed of the following visualization techniques:

- The Colored Reorderable Matrix, which enables a low level visualization of the underlying data by transforming the matrix values into an interactive colored matrix.

- The Multi-dimensional Scaling view, which includes classical MDS and the Sammon mapping to provide an aggregated view of the similarities in the input data.

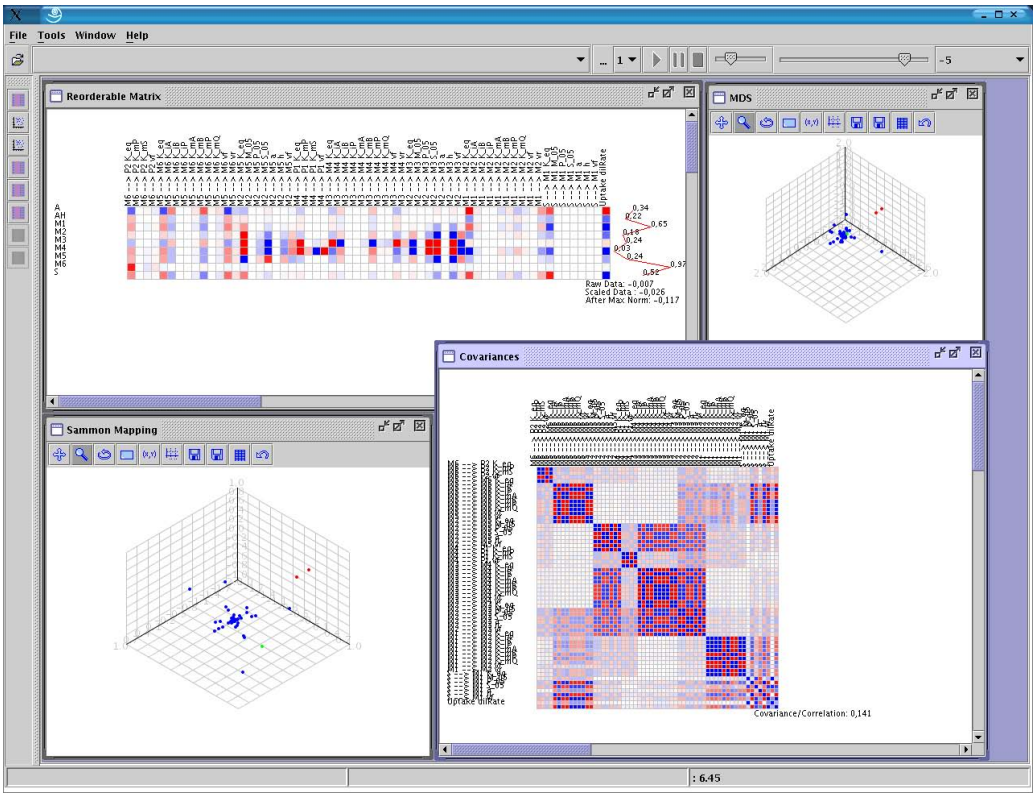


Figure 6.2: The MatVis graphical user interface

- The Reorderable Correlation Matrix, enabling an interactive view of both correlations and covariances of the input data.

- The Reorderable Cluster Membership Evolution Matrix, which based on results of clustering process for every time point, constructs a proximity matrix which contains information about the similarity of the objects of visualization, in this case parameters, with each other.

Although the methods contained in the toolkit are tailored to time-varying sensitivity matrices, no further domain knowledge information is used. In this way, the framework can be used to visualize any kind of similar data set of time-varying multi-dimensional data.

In the following, each of the techniques will be described in detail both from the implementation point of view and from the advantages/disadvantages they offer.

	r1 K_Ia	r1 K_Ib	r1 K_mS	r1 r_max	r2 K_eq	r2 K_mP	r2 K_mS	r2 r_max	r3 kdiff	r4 K_eq	r4 K_mP	r4 K_mS	r4 r_max	r5 kdiff
A	0.007	0.002	-0.01	0.013	-0.032	-0.001	0.002	-0.043	-0.031	-0.016	-0.001	0.001	-0.002	-0.003
P1	0.031	0.009	-0.046	0.059	0.072	0.003	-0.005	0.101	-0.384	-0.066	-0.003	0.004	-0.01	-0.012
P1X	0.156	0.045	-0.237	0.299	0.318	0.014	-0.024	0.432	0.479	-0.268	-0.012	0.016	-0.037	-0.017
P2	0.064	0.02	-0.097	0.124	-0.241	-0.011	0.018	-0.329	-0.231	0.257	0.012	-0.016	0.036	-0.074
P2X	0.193	0.055	-0.294	0.371	-0.397	-0.018	0.03	-0.535	-0.343	0.472	0.02	-0.028	0.063	0.979
S	-0.132	-0.039	0.2	-0.253	-0.098	-0.004	0.007	-0.133	-0.091	0.108	0.005	-0.007	0.015	-0.03

(a) Point of time $t = -5s$

	r1 K_Ia	r1 K_Ib	r1 K_mS	r1 r_max	r2 K_eq	r2 K_mP	r2 K_mS	r2 r_max	r3 kdiff	r4 K_eq	r4 K_mP	r4 K_mS	r4 r_max	r5 kdiff
A	-0.435	0.42	0.652	-0.076	-0.044	-0.002	0.003	-0.06	-0.042	0	0	0	0	-0.007
P1	-0.617	0.58	0.927	-0.13	0.021	0.001	-0.001	0.03	-0.429	-0.013	-0.001	0.001	-0.002	-0.028
P1X	0.119	0.058	-0.181	0.262	0.294	0.013	-0.022	0.401	0.417	-0.252	-0.011	0.016	-0.035	-0.026
P2	-0.541	0.541	0.814	-0.07	-0.303	-0.014	0.023	-0.415	-0.292	0.319	0.015	-0.02	0.045	-0.094
P2X	0.166	0.062	-0.251	0.337	-0.449	-0.02	0.034	-0.609	-0.404	0.514	0.023	-0.031	0.07	0.963
S	0.004	-0.005	-0.006	-0.001	-0.001	0	0	-0.001	-0.001	0.001	0	0	0	0

(b) Point of time $t = 0.1s$

	r1 K_Ia	r1 K_Ib	r1 K_mS	r1 r_max	r2 K_eq	r2 K_mP	r2 K_mS	r2 r_max	r3 kdiff	r4 K_eq	r4 K_mP	r4 K_mS	r4 r_max	r5 kdiff
A	-0.435	0.42	0.652	-0.076	-0.044	-0.002	0.003	-0.06	-0.042	0	0	0	0	-0.007
P1	-0.617	0.58	0.927	-0.13	0.021	0.001	-0.001	0.03	-0.429	-0.013	-0.001	0.001	-0.002	-0.028
P1X	0.119	0.058	-0.181	0.262	0.294	0.013	-0.022	0.401	0.417	-0.252	-0.011	0.016	-0.035	-0.026
P2	-0.541	0.541	0.814	-0.07	-0.303	-0.014	0.023	-0.415	-0.292	0.319	0.015	-0.02	0.045	-0.094
P2X	0.166	0.062	-0.251	0.337	-0.449	-0.02	0.034	-0.609	-0.404	0.514	0.023	-0.031	0.07	0.963
S	0.004	-0.005	-0.006	-0.001	-0.001	0	0	-0.001	-0.001	0.001	0	0	0	0

(c) Point of time $t = 0.5s$

	r1 K_Ia	r1 K_Ib	r1 K_mS	r1 r_max	r2 K_eq	r2 K_mP	r2 K_mS	r2 r_max	r3 kdiff	r4 K_eq	r4 K_mP	r4 K_mS	r4 r_max	r5 kdiff
A	0.001	0.052	-0.001	0.059	-0.008	-0.001	0.001	-0.043	-0.007	-0.037	-0.006	0.006	-0.019	0.01
P1	0.001	0.08	-0.001	0.091	0.062	0.004	-0.006	0.333	-0.358	-0.057	-0.009	0.01	-0.029	0.015
P1X	-0.008	0.262	0.013	0.296	0.223	0.014	-0.02	0.997	0.266	-0.193	-0.026	0.03	-0.106	0.028
P2	0.006	0.327	-0.007	0.371	-0.049	-0.003	0.005	-0.268	-0.046	0.047	0.008	-0.009	0.027	-0.075
P2X	0.015	0.697	-0.018	0.815	-0.136	-0.008	0.012	-0.609	-0.127	0.157	0.017	-0.019	0.05	0.936
S	0	-0.012	0	-0.013	-0.004	0	0	-0.016	-0.004	0.004	0.001	-0.001	0.003	-0.004

(d) Point of time $t = 39s$

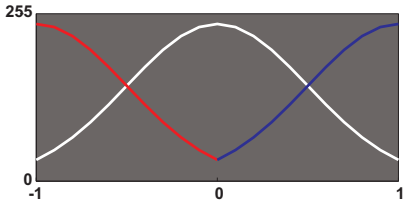
Figure 6.3: Four matrices in four different points of time. The first point is before the substrate pulse, the three others are after the pulse.

6.5 The Asymmetrical Reorderable Matrix

6.5.1 Basics

The information that is hidden in the input data, such as similarities or correlation, is difficult if not impossible to extract only by looking at numbers. Considering that the human user is more sensitive to visual stimuli, the substitution of numbers by symbols or colors is a good way to eliminate this problem. Thus, the basic idea of our approach is to transform a matrix of numerical data into a matrix of colors. In the original version of the reorderable matrix, numerical values are replaced with ink blobs. Since we consider values in [-1, 1], the use of color spectra is more appropriate. The color spectrum we use is defined by three colors: two border colors and the transition color, as shown in figure 6.4(a). Spectra with more colors are not used

(a) The color spectrum Red-White-Blue



(b) Mapping of values into colors

Figure 6.4: The color spectrum Red-White-Blue in (a) is used for mapping values into colors as illustrated in (b)

though in specific problem domains this could be useful. The segment [-1,1] is divided in as many small segments as nuances of colors are used (we use 511 colors; 256 for red, 256 for blue, but white is common to both), and the transformation to colors is done according to Formula (6.4) where ColorMap is a function of the spectrum in Figure 6.4(a).

$$color = ColorMap(value \times 255) \tag{6.4}$$

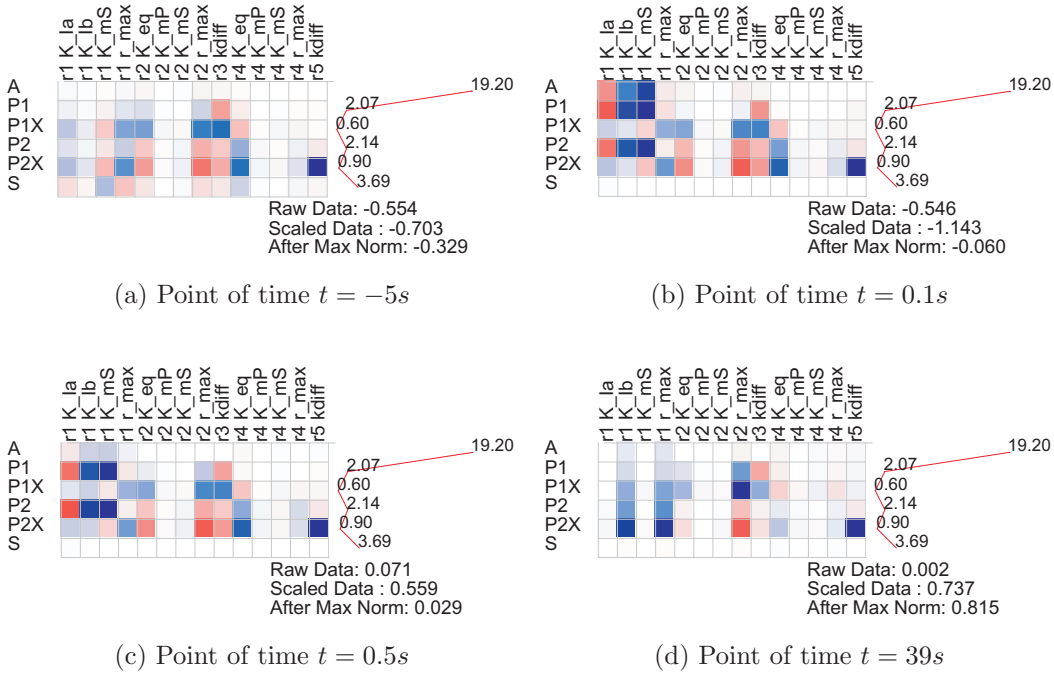
Our idea of using color scales as a means of communicating fine variations in numeric data is partially inspired by the red-green heat-maps that bioinformaticians have successfully used to visualize microarray data.

Figure 6.3 shows four matrices which were extracted from the set of time-varying sensitivity matrices generated during the simulation of the example metabolic network model presented in Section 6.3. Figure 6.5 shows the corresponding visualization as described above without any postprocessing steps regarding the ordering of columns. In the right of the matrix visualizations, the maximum norm as described in 6.3.2 is plotted, which is the same for all time points.

6.5.2 Local Reordering Techniques

In order for this color matrix visualization to be successful, the columns and rows of these matrices must be reorganized via manual permutations or by algorithms for the automatic generation of the most suitable permutations, since the color visualization alone does not allow the easy detection of structures in the data. Automatic ordering algorithms can be focused on different ordering objectives:

- Reordering the matrix in one dimension, for discovering the structures existing in that dimension.
- Reordering the matrix to obtain blocks of similar values in the matrix.



(a) Point of time $t = -5s$

(b) Point of time $t = 0.1s$

(c) Point of time $t = 0.5s$

(d) Point of time $t = 39s$

Figure 6.5: The color visualization of the matrices presented in Figure 6.3. The color hues indicate higher sensitivity just after the pulse in 6.5(b) and 6.5(c).

In the context of this work, the first group of techniques is important. However, in other contexts, the second group of techniques might be also important.

6.5.2.1 One-Dimensional Reordering Problem

One of the approaches for making patterns visible in the reorderable matrix is to sort either rowwise or columnwise, depending on how the objects and their patterns are placed. The problem encountered in this context is given formally in the following. Here, the formulation for column ordering is given, the row based formulation follows similarly. Suppose that we have a matrix of values $X_{m \times n}$, where n is the number of columns. Each column of X is treated as a vector and the distance function d gives the Euclidean distance between these vectors. Then, the ONE DIMENSIONAL ORDERING problem is defined as the permutation π, which minimizes the function $\sum_{i=0}^{n-2} d(\pi_i, \pi_{i+1})$, where $d(\pi_i, \pi_{i+1})$ is the distance between columns π_i and π_{i+1}. This definition is similar to the definition of the TRAVELING SALESMAN PROBLEM (TSP), but in contrast to the TSP the sum does not include the distance between the last element of permutation and the first one. The respective

decision problem would be defined as finding the optimal permutation where $\sum_{i=0}^{n-2} d(\pi_i, \pi_{i+1}) < B$, where $B \in \mathbb{R}$ is a fixed upper bound.

Theorem 1: ONE DIMENSIONAL ORDERING is NP-complete.
Proof. To prove Theorem 1, it is required to define a polynomial transformation from a known NP-complete problem to the ONE DIMENSIONAL ORDERING (ODO) problem. HAMILTIONIAN PATH (HP) will be used for this purpose [GJ79]. We will indeed show that the intractability of HP implies the intractability of the ODO decision problem. Thus ODO is as hard as HP.

For this purpose, a polynomial transformation t which transforms instances of the HP problem into instances of ODO problem is sought. This transformation, except being polynomial, must also fulfill the condition that for every instance P of HP that we know that a hamiltonian path exists if and only if there is a solution for the corresponding problem $f(P)$. Suppose we have the graph $G = (V, E)$ where $|V| = n$ is an instance of HP. The transformation t transforms every instance G to an instance of ODO, composed by the set of columns C identical to V, where the distance between columns i and j is defined to be 1 if $v_i, v_j \in E$ and 2 otherwise. The bound B is set equal to $n - 1$. This transformation is clearly polynomial since the maximal number of edges E we consider is $n(n-1)/2$. We need now to show that G contains a Hamiltonian Path if and only if there is an ordering of $t(G)$ with length no more than B. If $< v_0, v_1, ..., v_{n-1} >$ is a Hamiltonian Path in G, than its length is $\sum_{i=0}^{n-2} 1 = n - 1$ hence the forward direction is proved. Suppose now that $< v_0, v_1, ..., v_{n-1} >$ represents an ordering of the columns with length no more than $B = n - 1$. Since for each column pair, the distance is either 1 or 2 and the computed value of the ordering is the sum of $n - 1$ numbers with values either 1 or 2, all the distances are 1 (if any distance is 2 than there should be a distance 0 to balance this value, which is not possible). This implies that the every pair v_i, v_{i+1} is an edge of E and consequently $< v_0, v_1, ..., v_{n-1} >$ is an Hamiltonian Path of G. ∎

Although the general one-dimensional ordering problem is NP-complete, there are special cases when polynomial time algorithms exist. One such example is given by Bar-Joseph et al. [BJGJ01] in a different context. If ordering is performed on data which is before subjected to a hierarchical clustering procedure, then optimal ordering proceeds in polynomial time, although of the order $O(n^4)$. This comes from the fact that the number of possibilities to be considered is reduced and the one dimensional order must be embeddable in the binary tree resulting from the hierarchical clustering.

Algorithm 6.1: Weighted ordering algorithm

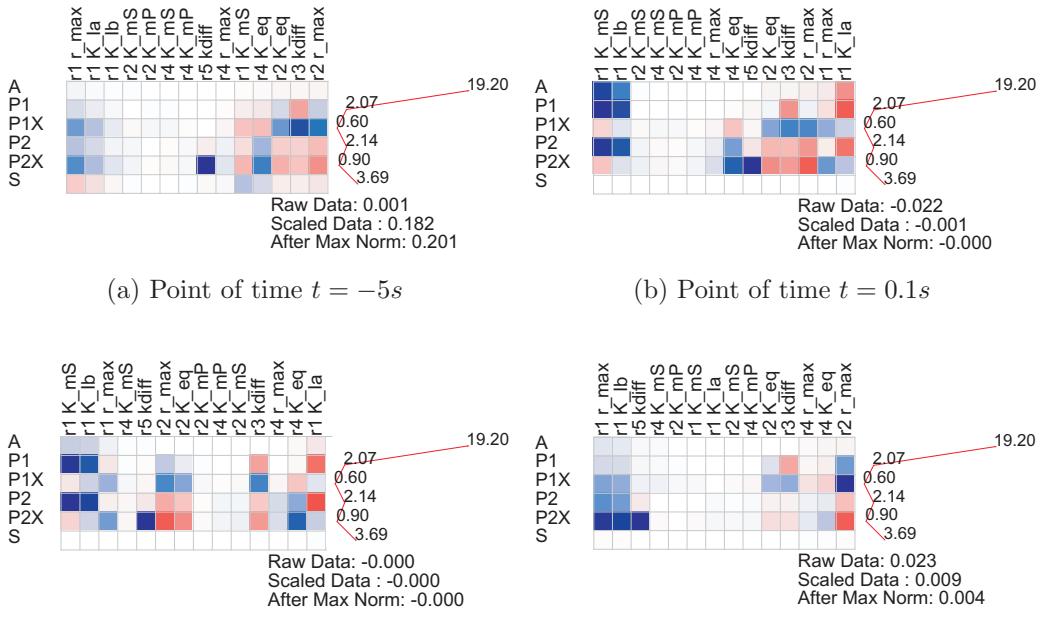
Input: Input the matrix $S_{m \times n}$ corresponding to the set of parameters $P = \{p_1, ..., p_n\}$, each with m features
Input: The weight vector $w = (w_1, ..., w_m)$
Output: The ordered matrix $\widetilde{S}$

```
for i = 1 to n do
    for j = 1 to m do
        weight(i) += S(i, j) × w(j);
    end
end
Sort weight vector e.g. with quicksort and keep the indices in I;
Obtain the ordered matrix S̃ by reordering S according to I;
```

6.5.2.2 Heuristics for One-Dimensional Reordering

Considering that no exact solution can be found for the one dimensional ordering problem, a range of heuristics were adapted to accomplish this purpose. The first one is a weight-based sorting algorithm.

Algorithm 6.1 shows the weighted ordering algorithm. Its functioning de-



(a) Point of time $t = -5s$

(b) Point of time $t = 0.1s$

(c) Point of time $t = 0.5s$

(d) Point of time $t = 39s$

Figure 6.6: The weighted sorting of colored visualization of the matrices presented in Figure 6.3 with Algorithm 6.1

Algorithm 6.2: Weighted spectral seriation

Input: The Euclidean distance matrix $D_{n\times n}$ corresponding to the distances between pairs of parameters $P = \{p_1, ..., p_n\}$, each with m features, the weighting factor σ

Output: The ordering represented by the permutation π

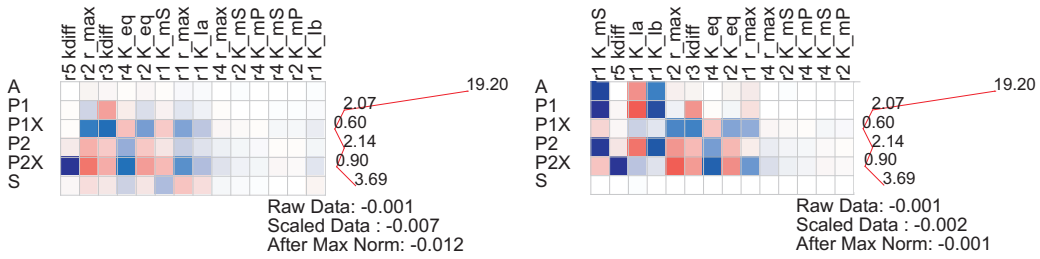
1 $A_{ij} = exp(-\frac{D_{ij}^2}{2\sigma^2})$ for $i \neq j$
2 $A_{ii} = 0$
3 $G_{ii} = \sum_{j=1}^{n} A_{ij}$
4 $G_{ij} = 0$ for $i \neq j$
5 $L = G^{-\frac{1}{2}} A G^{-\frac{1}{2}}$
6 Find the largest eigenvector x_1 of L
7 Sort x_1 e.g. with quicksort and keep the indices in π;

pends on the weight vector w. By choosing the weight vector $w = (1, d, d^2, ... d^{m-1})$ or a permutation thereof, where d is the maximum discretization value used in the visualization, matrices with m dimension up to 50 can be ordered. This happens due to the overflow that can occur in step 3 of the algorithm. This includes a large range of sensitivity matrices generated from middle scale models encountered in our project. For larger dimensions, a simple modification of step 3 can be accomplished using the formula $weight(i) += log(S(i, j) \times w(j))$ for avoiding the overflow. Figure 6.6 shows the three visualizations of the matrices in Figure 6.5 after the columns have been permuted according to Algorithm 6.1.

A similar problem to the ordering of the reorderable matrix is represented by *seriation*, a concept which comes from archaeology and is used for placing artifacts discovered in archaeological sites in chronological order. Various techniques are used to achieve seriation [Ken71] and *spectral methods* play an important role in this context. However, in the classical approach to seriation, they differ from our case because they consider only binary data. To adapt these techniques for our problem, the method introduced in [NJW02] for achieving spectral clustering, which will be encountered again in Chapter 7, is further extended for accomplishing seriation. Algorithm 6.2 shows the weighted spectral ordering algorithm. The distance matrix is modified appropriately using a Gaussian weighting function. Other techniques used in spectral clustering would be to select k nearest neighbors or all the neighbors which lie in a ϵ neighborhood of a certain point. Such approaches are, however, more time-consuming compared to Gaussian weighting. Eigenvalue decomposition is then applied to the weighted matrix L, which is similar to the Laplacian matrix of simple unweighted graphs. To order the matrix, the largest eigenvector (i.e. the eigenvector corresponding to the largest eigenvalue) is taken and the order of its values defines the order of the matrix.

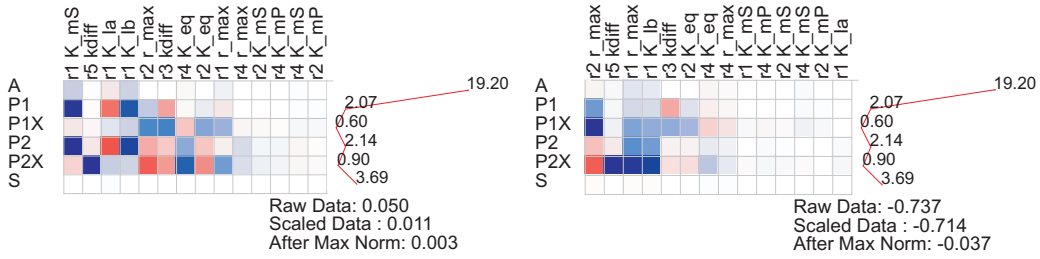
Figure 6.7 shows the result of weighted spectral clustering on the same set of matrices of Figure 6.3. In contrast to Figure 6.6, in Figure 6.7 the matrices seem more visually disordered. The various spectral clustering methods, which in contrast to us use k eigenvectors simultaneously, differ from each other in the set of these eigenvectors used (k largest ones or k smallest ones). The proper selection of the eigenvector is crucial in the ordering process. As such, we extended the idea further by introducing an exhaustive spectral ordering algorithm, as shown in Algorithm 6.3. Algorithm 6.3 tries exhaustively all the eigenvectors of L and keeps the ordering that minimizes the sum of distances of consecutive matrix columns. Figure 6.8 shows that the results of the exhaustive spectral sorting, which are visually superior compared to the simple spectral sorting.

A third class of heuristics is constructed by considering the ordering problem as an optimization problem. As such, we can use the similarity that one dimensional ordering has with the Traveling Salesman problem (TSP) [JM97], an analogy which will be used again in Section 6.5.3.The TSP is defined as follows: A salesman has to visit N cities. Each city is visited only once and the final city is the same as the starting city. In which order should the salesman visit the cities such that the distance traveled is minimized? In contrast to the TSP, the path traversed here is not a cycle. To formal-



(a) Point of time $t = -5s$

(b) Point of time $t = 0.1s$



(c) Point of time $t = 0.5s$

(d) Point of time $t = 39s$

Figure 6.7: The simple spectral sorting of colored visualization of the matrices presented in Figure 6.3 Using Algorithm 6.2

Algorithm 6.3: Exhaustive Weighted Spectral Seriation

Input: Euclidean distance matrix $D_{n\times n}$ corresponding to the distances between pairs of parameters $P = \{p_1, ..., p_n\}$, each with m features, The weighting factor σ

Output: The ordering represented by the permutation π

1 $A_{ij} = exp(-\frac{D_{ij}^2}{2\sigma^2})$ for $i \neq j$
2 $A_{ii} = 0$
3 $G_{ii} = \sum_{j=1}^{n} A_{ij}$
4 $G_{ij} = 0$ for $i \neq j$
5 $L = G^{-\frac{1}{2}} A G^{-\frac{1}{2}}$
6 $d_{min} = \infty$
7 **for** $i = 1$ *to* n **do**
8 Find the i_{th} largest eigenvector x_i of L
9 Sort x_i e.g. with quicksort and keep the indices in $\pi^{(i)}$;
10 $d_{temp} = \sum D_{\pi_i \pi_{i+1}}$
11 **if** $d_{temp} \leq d_{min}$ **then**
12 $d_{min} = d_{temp}$
13 $\pi = \pi^{(i)}$
14 **end**
15 **end**

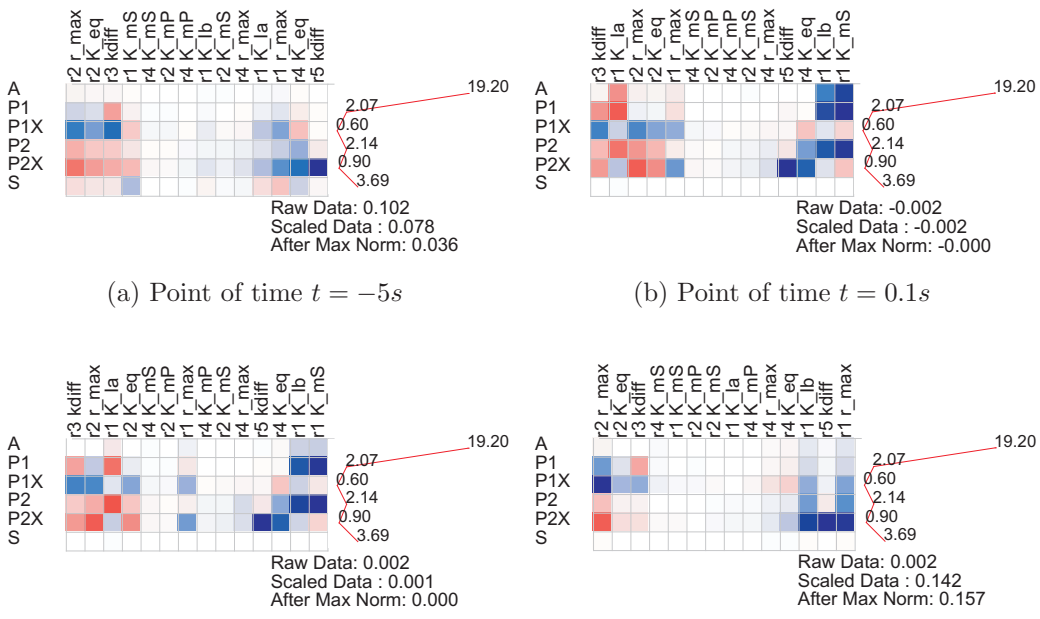
ize the idea, let $S_{m\times n}$ represent a sensitivity matrix with dimensions $m \times n$ and $D_{n\times n}$ be the Euclidean distance matrix obtained when considering the columns of S as vectors in the space R^m. The purpose is to find the permutation $\pi = \{\pi_1, ..., \pi_n\}$, which minimizes the value $\sum_{i=1}^{n} d(\pi_i, \pi_{i+1})$. To optimize this sum, a range of techniques can be used from local search to more sophisticated techniques such as genetic programming and simulated annealing. The same sum of distances can be used as an indicator of the goodness of TSP-based approaches and in connection with the other heuristics presented above. Figure 6.9 shows the matrices as they are ordered by using simulated annealing for finding the (possibly local) optimum sum of the distances.

6.5.2.3 Heuristics for Block Ordering

The problem of automatically computing the "optimal" permutations for both the columns and rows of the reorderable matrix method has been addressed also by Mäkinen and Siirtola [MS00]. The authors make an analogy between reordering the matrix and the bipartite graph drawing problem and give heuristics on how to automatically reorder matrices. Heuristics for drawing bipartite graph drawing focus on layouts with as few edge crossings as

possible. If the bipartite graph is represented by an adjacency matrix, the problem of drawing the bipartite graph with the minimal number of edge crossings is transformed into the problem of grouping the ones in the adjacency matrix (representing edges) into blocks alongside the diagonal of the matrix. However, the difference between the ordering of adjacency matrices and the reorderable matrix consists in the fact that adjacency matrices contain only binary values representing if a edge is present or not, whereas the reorderable matrix takes values from an ordered set $\{v_1, v_2, ..., v_{max}\}$. In the Sugiyama algorithm [STT81], the vertices on one side of the bipartite graph are ordered according to the order of average values of their adjacent nodes. Mäkinen and Siirtola apply the Sugiyama algorithm (basically the part of Sugiyama algorithm dealing with crossing minimization) for ordering matrices both with binary values and with values coming from a discrete set. In the latter case, a threshold is defined interactively and the problem is converted to the former case.

However, the threshold based approach cannot be applied for the visualization of sensitivity matrices containing arbitrary values. The graph obtained for arbitrary values presents a complete bipartite graph and since the graph is complete, it makes no sense speaking about the reduction of the number of crossings. To achieve automatically block ordering, we apply the techniques



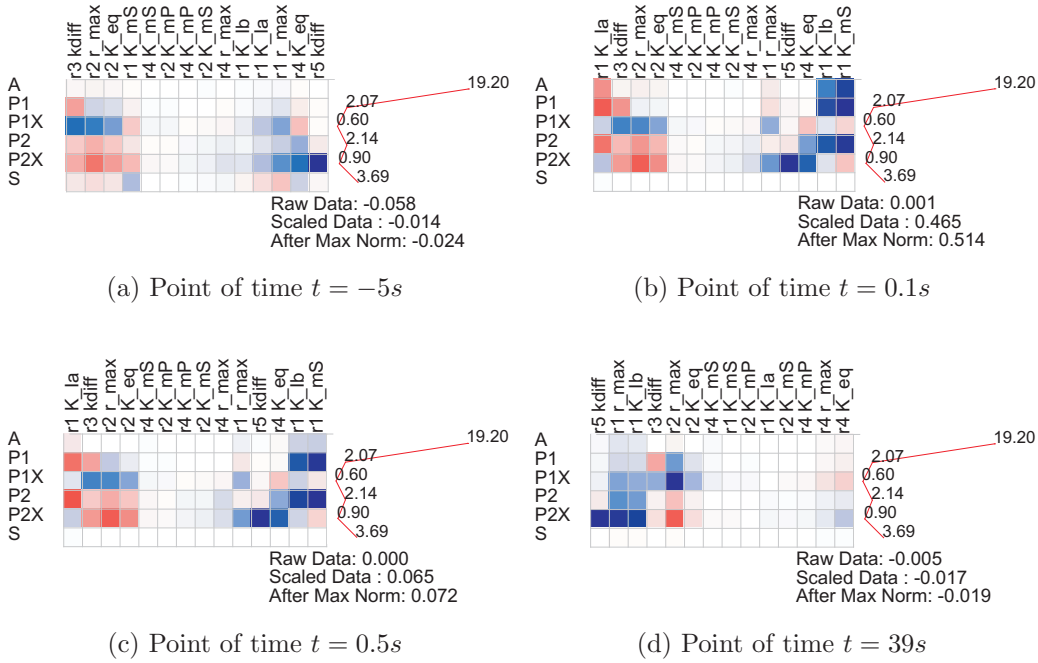
(a) Point of time $t = -5s$

(b) Point of time $t = 0.1s$

(c) Point of time $t = 0.5s$

(d) Point of time $t = 39s$

Figure 6.8: The exhaustive spectral sorting of colored visualization of the matrices presented in Figure 6.3 using Algorithm 6.3



(a) Point of time $t = -5s$

(b) Point of time $t = 0.1s$

(c) Point of time $t = 0.5s$

(d) Point of time $t = 39s$

Figure 6.9: The ordering of colored visualization of the matrices presented in Figure 6.3 using Simulated Annealing

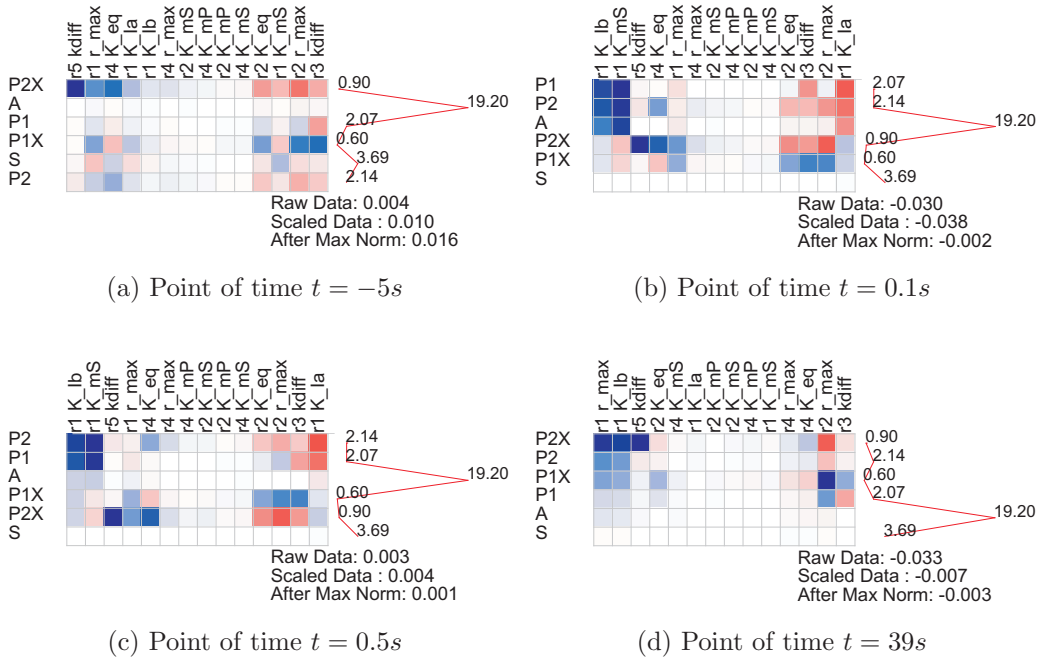
presented above for one dimensional ordering both on columns and the rows of the reorderable matrix. This approach keeps the spirit of the Sugiyama algorithm, especially when Algorithm 6.1 is applied on both columns and rows of the matrix, since the Sugiyama algorithm is nothing more than a weighted sorting. Figure 6.10 shows the three matrices used for illustration in this chapter after applying the one dimensional sort on both dimensions.

6.5.2.4 Reordering via Interaction

In addition to the automatic generation of the optimal permutation, our realization of the reorderable matrix method allows reordering in an interactive way, supporting the user during the visual exploration process. The user can select a set of columns to observe by highlighting the selection or dimming the rest out. Our implementation allows the user to swap columns of matrices in the case when he or she is not pleased with the found permutation.

6.5.2.5 Discussion

The above techniques are applied interactively in the MatVis environment. As such, the user must be conscious about the advantages of the specific meth-



(a) Point of time $t = -5s$

(b) Point of time $t = 0.1s$

(c) Point of time $t = 0.5s$

(d) Point of time $t = 39s$

Figure 6.10: The ordering of colored visualization of the matrices presented in Figure 6.3 on both dimensions

ods. First of all, the techniques need to provide their results in a reasonable time, otherwise the user will experience delays in the visual exploration process. From the complexity point of view, Algorithm 6.1 has a complexity of order $O(n\ log\ n)$ as it is derived from Quicksort. Applied in two dimensions as described in Section 6.5.2.3 it will have a complexity of $O(n\ log\ n + m\ log\ m)$, where m and n are the dimensions of the matrix to be ordered. Weighted spectral ordering presented in Algorithm 6.2 depends on the calculation of eigenvalue decomposition of a n-dimensional square matrix. The rest of the algorithm has a complexity of $O(n\ log\ n)$. In Algorithm 6.3 the same factor is of order $O(n^2\ log\ n)$. From the quality of orderings they generate, Algorithm 6.1 although quick and easy to understand, can sometimes generate orderings with outliers in the middle of similar columns. This comes from the fact that a weighting vector is used, which implies a priority for the rows of the matrix (if the columns are sorted). However, these difficulties can be overcome by interactively specifying an order of priorities during the visualization process. From the spectral ordering techniques, it is clear that exhaustive ordering has superior results because it includes the simple spectral ordering. They produce stable results and especially the exhaustive approach also of high quality.

Figure 6.11 shows the sum of Euclidean distances of consecutive columns

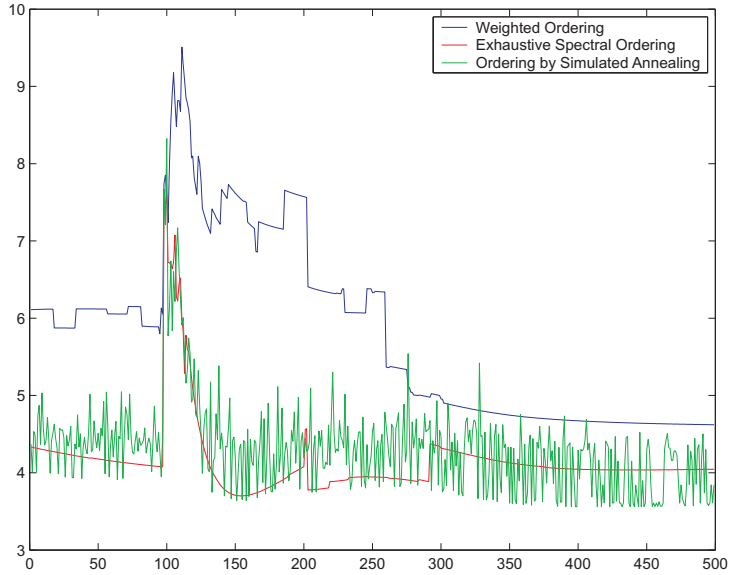


Figure 6.11: Comparison of the different ordering techniques for 500 time points

for three of the above mentioned ordering methods. Simple spectral ordering is ommitted because its results are always worse (or the same) than exhaustive spectral ordering. The results are reported for a set of sensitivity matrices consisting of 500 time points for the model described in Section 6.3. Exhaustive spectral ordering has the best results overall; however, simulated annealing generates comparable results. Weighted ordering is the worst; however considering that the purpose of all three methods is to help the user in the visual exploration process, all three generate satisfiable results.

6.5.3 The Time-Dependent Reorderable Matrix

The reorderable matrix presented until now considered time-varying data one matrix at a moment. Considering that we deal with evolving time-varying matrices, the permutations calculated in subsection 6.5.2 can vary over time. Below, the index of one parameter over an extract of several time points is shown. We see that although the index stays constant for several consecutive time points, it jumps on several states when the entire time frame is considered.

$$\overbrace{11,...,}^{85\text{ times}}\ \overbrace{5,...,}^{12\text{ times}}\ ...\ \overbrace{2,...}^{200\text{ times}}$$

In this context, enabling the user to explore the data with the reorderable matrix without being distracted by the swapped columns, means to find the optimal permutation over the time which "considers" the local orderings for each time point. Thus, a discrete optimization problem is raised: we have to

find the optimal permutation which has the minimum distance to all other permutations over time.

6.5.3.1 Global Reordering Problem

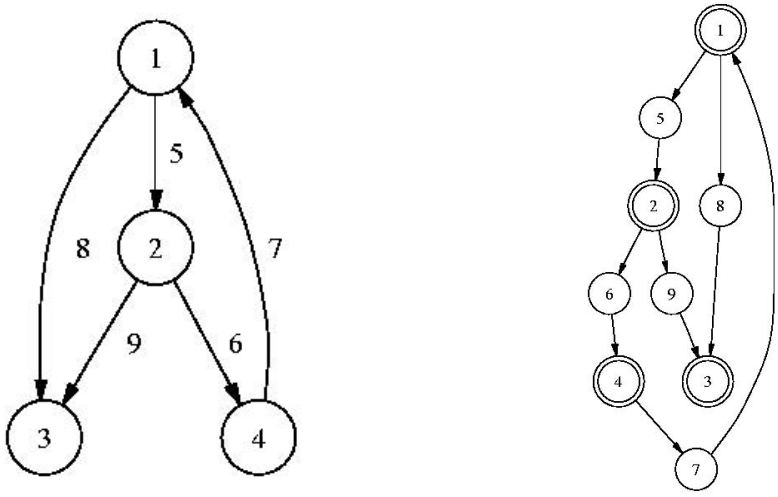
Basically, the global reordering problem, i.e. the problem of finding the permutation, which on the average "agrees" with the permutations calculated for every time step, is similar to the problem of finding a *Kemeny optimal permutation*. The Kemeny optimal permutation is defined as the ordering σ which minimizes the sum of Kendall distances with orderings τ_1, τ_2,..., τ_k. Kendall distance is defined as the number of swaps of adjacent pairs required to transform one permutation into the other.

The complexity of Kemeny optimal permutation is studied in several works [ITT89, DKNS01, BBD05]. Bartholdi et al. [ITT89] treat the problem in a different context, namely while considering the difficulty of manipulating certain kinds of election schemes. The ordering of matrices globally bears more similarity with the works of Dwork et al. [DKNS01] and Biedl et al. [BBD05] and thus the following theorem will be based on the theorems provided in these works.

Dwork et al. remark that computing the Kemeny optimal permutation for two permutations is trivial by outputting one of the input permutations. However, in the context of visualization with the reorderable matrix it might be interesting not just to output one of the input permutations but to compute the permutation, which is equidistant using the Kendall distance from both permutations. For three permutations, Dwork et al. remark that the complexity is still open [DKNS01]. Formally, the GLOBAL ORDERING problem is stated as follows: For a list of permutations $\pi_1, ..., \pi_k$, find the permutation σ which minimizes $\sum_{i=1}^{k} d_K(\sigma, \pi_i)$.

Theorem 2: GLOBAL ORDERING is NP-complete for k full permutations, where $k \geq 4$ and k is even.
Proof. The reduction follows from the feedback arc set problem. The feedback arc set problem is stated as follows. Given a directed graph $G = (V, E)$, and an integer $L \geq 0$, the question is whether there exists a subset F of E such that $|F| < L$ and $(V, E - F)$ is acyclic.

Let's consider first the case when k is even. Let $n = |V|$ and $m = |E|$. Given G, a new graph G' is produced by introducing new nodes, which are formed by splitting the edges of the graph G into two edges. Formally, the graph $G' = (V', E')$ is defined as follows: V' denotes the union of V and the set $\{v_e : e \in E\}$ and $E' = \{(i, v_{i,j}), (v_{i,j}, j) : (i, j) \in E\}$. Figure 6.12 illustrates the transformation from G to G'. The original nodes 1,2,3 and 4 are left unchanged. Furthermore, the edges of the original graph, which are numbered intentionally, are transformed into nodes in the new graph G'. The



(a) Graph G (b) Graph G'

Figure 6.12: The graph G and its transformation G'

graph G' has a minimum feedback arc set of size L if and only if G does. This fact is proven in the following Lemma.

Lemma: 6.3.1 G' has a minimum feedback arc set of size L iff G does.

Proof. To prove the only if direction, suppose that G has a feedback arc set of size L. We can assume that the edges e_1, e_2,...,e_L are removed from the graph G and the resulting graph is then acyclic. By removing one of the two edges created for every edge $e_i, i \in 1..L$ in G', the resulting graph in G' is acyclic. Indeed, if any cycle exists afterwards in G' then from the construction procedure described above this cycle traverses the nodes $v_1^{G'}$, $v_2^{G'}$, ..., $v_{C-1}^{G'}$, $v_C^{G'}$, $v_1^{G'}$. If we assume that $v_1^{G'}$ is also a node of G, which does not restrict our proof because we can begin the numbering from such a node, then the cycle has the form $v_{i_1}^G$, $e_{j_1}^G$, ..., $v_{i_I}^G$, $e_{j_J}^G$, $v_{i_1}^G$. This means that a cycle traversing the nodes $v_{i_1}^G$, ..., $v_{i_I}^G$, $v_{i_1}^G$ exists in G because both edges created in G' are in the modified graph G' and as such the respective edge is in the modified graph G. This contradicts the assumption that the modified graph G is acyclic.

To prove the if direction, if G' has a minimum feedback arc set of size L, it is clear from the construction that no two adjacent edges (having a common node), which are created for every edge in G, can belong to such a feedback set because removing one of them would result in a smaller feedback arc set. Again, for every edge removed in G' we remove its corresponding edge in G. If a cycle exists in G, then it has the form $v_{i_1}^G$, ..., $v_{i_I}^G$, $v_{i_1}^G$. By introducing the

artificial nodes added in G', we achieve the cycle $v_1^{G'}$, $v_2^{G'}$, ..., $v_{C-1}^{G'}$, $v_C^{G'}$, $v_1^{G'}$ in G'. All the edges of this cycle belong to the modified graph G' (after the removal of the edges belonging to the minimum feedback arc set) because if not than the respective edges in the cycle in G would also have been removed in G. The existence of such a cycle contradicts the assumption that G' has a minimum feedback arc set. ∎

Let us define the basis permutation Z by ordering the vertices of G' so that the original vertices of G receive the numbers 1, ...,n and the vertices of G' induced by edges of G receive the numbers n+1, ...,n+m. Let out(i) denote the sequence of out-neighbors of vertex i in G' and in(i) the sequence of in-neighbors. If $1 \leq i \leq n$ than the lists out(i) and in(i) contain only vertices of the original graph G. Two pairs of permutations are constructed from the vertices of G:

$$\begin{aligned}
\pi_1 &= 1, out(1), 2, out(2), ..., n, out(n) \\
\pi_2 &= n, out(n)^r, n-1, out(n-1)^r, ..., 1, out(1)^r \\
\pi_3 &= in(1), 1, in(2), 2, ..., in(n), 2 \\
\pi_4 &= in(n)^r, n, in(2)^r, 2, ..., in(1)^r, 1
\end{aligned}$$

where L^r denotes the reversal of the list. The permutation π_1 represents an ordering of the vertices of G' where the out-neighbors of each vertex of G are preceded by the vertex itself, but the internal order of out-neighbors and their predecessor vertices is done according to Z. The permutation π_2 reverses the order of out-neighbors while keeping the predecessor vertices (of graph G) before their out-neighbors. Similarly, in the permutation π_3 the vertices are preceded by their in-neighbors (in the original version [DKNS01] this is misswritten and is corrected in [BBD05]) and in t_4 these orders are analogously to π_2 reversed.

The claim is that G has a feedback edge set of size L iff there is a permutation π^*, such that $\sum_{r=1}^{4} d_K(\pi^*, \pi_r) \leq L'$, where $L' = 2L + 2(\binom{n}{2} + \binom{m}{2} + m)$.

Suppose that G has a feedback edge set F of size L. The set $F' = \{(i, v_{i,j}) | (i,j) \in F\}$ is a feedback arc set for G' and as such the graph $(V', E' - F')$ is acyclic. Thus, we can apply a *topological sorting* and obtain an ordering π of vertices V' such that for every edge $(i,j) \in E' - F'\}$ i is placed before j in π. The sum of Kendall distances of π with $\pi_i, 1 \geq i \leq 4$ of the following four terms, where the three first ones are independent of how π was obtained:

1. For each pair $i, j \in V$, exactly one of π_1 and π_2 places i above j and the other places j above i, contributing thus 1 to $d_K(\pi, \pi_1) + d_K(\pi, \pi_2)$ and analogously contributing of 1 to $d_K(\pi, \pi_3) + d_K(\pi, \pi_4)$. Together, they account for the term 2n(n-1)/2.

2. Similarly, for each pair $i, j \in E$, and there are m(m-1)/2 such pairs, once i is ranked above j, then j is ranked above i for both π_1, π_2 and π_3, π_4 pairs, accounting for the term 2m(m-1)/2.

3. Furthermore, for pairs i, j where $i \in V$ and $j \in E$ with respect to π_3 and π_4 (because the vertices of G come after their in-neighbors in π_3 and π_4), and for pairs i, j where $i \in E$ and $j \in V$, with respect to π_1 and π_2 (because the out-neighbors of vertices of V are listed after the vertices V) contribute each 1 to the sum of Kendall distances, totaling 2m.

4. The last term accounts for the contribution to the total Kendall distance of pairs i, j where $i \in V$ and $j \in E$ with respect to π_1 and π_2 and for pairs i, j where $i \in E$ and $j \in V$, with respect to π_3 and π_4. Such pairs contribute 2 to the sum of distances for every edge of G' which does not satisfy the order implied by the topological ordering of vertices of V'. The number of such edges is at most $|F'| = L$.

To prove the other direction, suppose that there exists a permutation π that has a total Kemeny distance of at most L' = 2L + 2(n(n-1)/2 + m(m-1)/2 + m) and we need to prove that the graph G' has a feedback set of size L. The three terms defined above are constant for every permutation and as such the fourth term contribution is equal to 2L. But again for every pair i, j where $i \in V$ and $j \in E$ with respect to π_1 and π_2 and for every pair i, j where $i \in E$ and $j \in V$, with respect to π_3 and π_4 the contribution to the sum of distances is either 0 or 2. Denoting the edges which contribute 2 to the sum with F' and keeping in mind that these edges do not satisfy the ordering in π, their removal would result in an acyclical graph (V',E'-F'). Thus, G' has a feedback arc set of size L and from Lemma 6.3.1 G has a feedback arc set of size L.

Thus, computing the optimal Kemeny permutation for four full lists is NP-hard. For an even number of permutations $k \geq 4$, the proof proceeds similarly by adding $\frac{k-4}{2}$ pairs of permutations which are the reverse of each other. In this case, the overall distance is increased by a $\frac{k-4}{2}\binom{n+m}{2}$ term. ∎

The proof for odd k provided in [DKNS01] has some flaws and as such we provided above only the proof for even ks. The correction of these flaws is not the object of this thesis and remains an open issue for further work. The two cases differ from each other because of the fact that when combining an odd number of ranking, for every pair of objects there is a well defined majority preference for ranking one above the other (or vice versa). This might not be the case when combining an even number of permutations.

6.5.3.2 Similarities on Orderings

Permutations define a bijective function $\pi : \{1, 2, ..., n\} \rightarrow \{1, 2, ..., n\}$. As such, every permutation has two representations, namely the *order* vector and the *rank* vector. The order vector presents an ordered list of elements, whereas the rank vector presents the list of ranks of the elements. Thus if we have the permutation presented with order vector $3, 1, 2, 5, 4$ its respective rank vector is $2, 3, 1, 5, 4$. These two representations identify uniquely their counterpart and furthermore they complement each other i.e. if we have permutation π and its rank σ then $\sigma = \pi^{-1}$ and $\pi \cdot \sigma = \{1, 2, ..., n\}$.

Similarity on orderings can be defined on their order vector or on their rank vector. The similarity measures are divided into two groups: spatial and disorder. Spatial similarity measures the closeness of permutations in the n-dimensional space, whereas disorder similarity measures the effort required to transform one permutation to the other. Deza and Huang [DH98] give a survey of metrics on permutations, without however distinguishing between order and rank vectors. We will mention below if a specific distance is calculated with respect to its order vector or its rank vector.

The Spearman distance is computed from the rank vectors following the formula:

$$d_S(\pi_a, \pi_b) = \sqrt{\sum_{i=1}^{n} (\pi_a(i) - \pi_b(i))^2} \tag{6.5}$$

With proper normalization, the Spearman's Rho coefficient is derived as follows:

$$\rho = 1 - \frac{6d_S^2}{n^3 - n} \tag{6.6}$$

The value of ρ lies in $[-1, 1]$ and it serves also as a correlation measure between the two orderings. Similar to the Spearman distance, the Spearman's footrule distance is defined (in analogy to the Manhattan distance for Euclidean space) as:

$$d_{footrule}(\pi_a, \pi_b) = \sum_{i=1}^{n} |\pi_a(i) - \pi_b(i)| \tag{6.7}$$

In the same spirit, the maximum norm can also be defined(see [DH98]).

The Kendall distance is based on the other side on the order vector. It represents the number of swaps of consecutive elements required to transform one permutation into the other.

$$d_K(\pi_a, \pi_b) = \sum_{i<j} I\{(\pi_a(i) - \pi_a(j)) \cdot (\pi_b(i) - \pi_b(j)) < 0\} \tag{6.8}$$

Here $I\{...\}$ denotes the indicator function, which has value 1 when the condition inside curly brackets holds, otherwise 0. Thus, the same order implies a Kendall distance of zero, a Kendall distance of one would mean one swap and the Kendall distance of $\frac{n(n-1)}{2}$ is the greatest possible distance between two permutations.

Its respective correlation similarity measure, Kendall's tau is defined as follows:

$$\tau = 1 - \frac{4d_K}{n^2 - n} \tag{6.9}$$

Diaconis and Graham [DG77] proved that for any two permutations of length n, π and σ, $d_K(\pi, \sigma) \leq d_{footrule}(\pi, \sigma) \leq 2d_K(\pi, \sigma)$.

The Cayley distance (d_C) is similar to the Kendall distance but swaps can occur between any two elements. Cayley distance is calculated as $n - C(\pi_a^{-1}\pi_b)$ [DG77], where $C(\pi_a^{-1}\pi_b)$ is the *number of cycles* of the permutation $\pi_a^{-1}\pi_b$. Permutation cycles represent subsets of permutations, whose elements map the subset into itself e.g. the permutation $4, 1, 3, 2$ contains the cycles $(1, 4, 2)$ and (3).

Table 6.3 [DG77] shows some of the properties of these distances. For the mean and variance only the leading terms are given. The Cayley distance has the lowest variance and this maybe explains the fact that this distance has not received much attention in literature.

Table 6.3: Properties of permutation distances

	Max	Mean	Variance
d_S	$\lceil \frac{1}{2}n^2 \rceil$	$\frac{1}{3}n^2$	$\frac{2}{45}n^3$
$d_{footrule}$	$\frac{1}{3}(n^3 - n)$	$\frac{1}{6}n^3$	$\frac{1}{36}n^5$
d_C	n-1	n-log(n)	log(n)
d_K	$\frac{1}{2}(n^2 - n)$	$\frac{1}{4}n^2$	$\frac{1}{36}n^3$

The Hamming distance is defined as the number of items whose rankings differs from one permutation to the other.

$$d_{Hamming}(\pi_a, \pi_b) = \sum_{i=1}^{n} I\{\pi_a(i) \neq \pi_b(i)\} \tag{6.10}$$

The last distance is known under different names as Ulam, edit or Levenshtein distance.

$$d_{edit}(\pi_a, \pi_b) = n - \text{length of the longest increasing subsequence of } \{\pi_b\pi_a^{-1}(1), ..., \pi_b\pi_a^{-1}(n)\} \tag{6.11}$$

It can be described differently as n minus the number of items which have the same rank in both permutations or as Levenshtein distance [Lev66] between two permutations, which is the number of minimal insertions, deletions or substitutions needed to convert one permutation to the other. The Levenshtein distance is calculated efficiently by a dynamic programming approach.

The last two distances have a smaller spectrum of values with minimum and maximum values 0 and n for the Hamming distance and 0 and n-1 for the Levenshtein distance. However, in different contexts they might be appropriate similarity measures. For example, if π_a = 1,2,3,4,5,6,7 and π_b = 7,1,2,3,4,5,6 then the two permutations have a Kendall distance of 6, and a Levenshtein distance of two (delete 7 at the end of first permutation, add 7 to the beginning). Such situations might occur frequently in the seriation of the reorderable matrix. Obviously such a change in permutations does not represent radical changes in the order of the matrix.

6.5.3.3 Heuristics for Global Reordering

Considering that the problem of finding exact solutions to the global ordering i.e. aggregation of permutations using the Kendall distance is NP-complete, heuristics for approaching this problem are needed. According to Bertin [Ber81, Ber83], the reorderable matrix method can be used for exploring matrices where one dimension can go up to 500 (and with distortion techniques or other focus+context techniques even larger matrices can be visualized), creating thus a large search space reaching in the worst case orders of 500!. The problem of finding the optimal permutation can be formulated as follows :

- Let π_t for t=1,..,n be the permutations of m columns of time-varying sensitivity matrix X(t) at all n points of time as calculated using one of the methods described in section 6.5.2.
- Let $d(\pi_1, \pi_2)$ be the similarity function between two permutations, as defined in Section 6.5.3.2.
- Find the permutation $\hat{\pi}$, for which $\sum_{k=1}^{n} d(\hat{\pi}, \pi_k)$ is minimized.

It is clear that finding the optimal permutation depends on the similarity measure used for defining the proximity between permutations. Furthermore, even the measurement of the goodness of the optimal permutation itself depends on this measure, since we need to estimate the sum of distances between the optimal permutation and the rest. Finding the optimal solution for the respective distance can be either NP-hard as in the case when the Kendall distance is used (as shown in Section 6.5.3.1), or efficient techniques might exist for its estimation as in the case of the Spearman footrule

Algorithm 6.4: Spearman footrule aggregation

Input: The permutations $\pi_1, \pi_2, ..., \pi_k$ of length n
Output: The spearman footrule permutation $\tilde{\pi}$
`/* Compute the sum of permutations */`
for $j = 1$ *to* k **do**
  for $i = 1$ *to* n **do**
    rankSum(i)=rankSum(i)+$\pi_j(i)$
  end
end
Sort *rankSum* (e.g. with quicksort) and keep the indices in $\tilde{\pi}$;

distance. Using the Kendall distance provides an aggregate solution, which fulfills several goodness properties, which are studied also in the context of aggregating election results. Thus, the optimal aggregation in the case of the Kendall distance, called also Kemeny optimal aggregation, fulfills the *Condorcet criterion* [DKNS01].

For this reason, we will develop heuristics for finding the aggregate permutation of a list of permutations, assuming that Kendall distance is used to measure the similarity between permutations. Basically, the heuristics can be divided into two large groups:

- Heuristics trying to "guess" good solutions based on the information provided by the existing list of permutations
- Heuristics based on optimization algorithms, which try to iteratively improve a certain solution of the problem. It is understandable that the first class of heuristics can be used to generate good initial solutions for this category of heuristics.

The first heuristic presented tries to aggregate the rank vectors of the permutations. It is basically an implementation of the Spearman footrule aggregation function. The ranks of each object, columns of the matrix in our case, are summed with each other for all permutations and the ordering implied by sorting the sum of ranks implies the aggregation. Its functioning is similar to the sum of *Borda* counts, which is often used in election procedures, where each of the n candidates receives a ranking from 0 (worst) to $n-1$ (best) and the winner(s) are those with highest sum of ranks. The Spearman footrule aggregation minimizes the sum of Spearman distances and can be used as a 2-approximation for the Kemeny optimal permutation [DKNS01, DG77]. It should be emphasized that the Spearman footrule aggregation does not necessarily belong to the list of input permutations being aggregated.

Another heuristic we found to often deliver good results works similarly to the Spearman footrule aggregation. We imposed the condition that the

Algorithm 6.5: Barycenter heuristic

Input: The permutations $\pi_1, \pi_2, ..., \pi_k$ of length n
Output: The average permutation $\tilde{\pi}$
`/* Compute the average of permutations */`
for $j = 1$ *to* k **do**
  for $i = 1$ *to* n **do**
    rankSum(i)=rankSum(i)+$\pi_j(i)$
  end
  rankSum(i)=$\frac{rankSum(i)}{k}$
end
`/* Calculate the Spearman distance of the average from every permutation */`
for $j = 1$ *to* k **do**
  dist(j)=$d_S(\pi_j, rankSum)$
end
`/* Output the permutation which has the minimal distance */`
Find the index j_{min} of the minimum of *dist* and set $\tilde{\pi} = \pi_{j_{min}}$;

permutation must belong to the list of permutations being aggregated. The heuristic is presented in Algorithm 6.5. Instead of calculating the sum of ranks, we calculate the mean of ranks for each object. The selected permutation is the permutation of the list which has the smallest Spearman distance from the vector of rank means. In this case, the distance between a permutation and a n-dimensional vector which is not a permutation, is calculated. However, as the Spearman distance is basically the Euclidean distance of permutations in the n-dimensional space, the solution works. The Spearman footrule distance can also be used in this case.

The last approach for solving the aggregation problem for permutations is again (similar to ODO in Section 6.5.2) based on heuristics for solving the traveling salesman problem. Whereas in Section 6.5.2 the transformation from ODO to TSP was straightforward, here the analogy is not so obvious. If a tour of the salesman is a permutation of the indices of cities if they are numbered from 1 to N, then the goal is to find the optimal permutation which minimizes the overall distance. The aggregation of permutations is thus defined in analogy to the optimal tour in TSP. The big difference is in the cost function; the aggregation of permutations should have minimal (Kendall) distance to the list of permutations it aggregates. Here we use Kendall distance, which is harder to calculate, but other distances defined in Section 6.5.3.2 could be used in the same way. The complexity of calculating the goodness of a certain permutation with respect to the objective function

Algorithm 6.6: TSP heuristic

Input: The permutations $\pi_1, \pi_2, ..., \pi_k$ of length n
Output: The optimized permutation aggregation $\tilde{\pi}$
`/* The cost function is defined as` $\sum_{j=1}^{k} d_K(\tilde{\pi}, \pi_j)$ `*/`

```
1 Initialize π_temp randomly or using one of the methods described in
  Algorithm 6.4 or in Algorithm 6.5
  /* Permute the solution locally until no improvement is
     found                                                  */
2 for j = 1 to n do
3 |   for i = 1 to n do
4 |   |   if reversal of elements i and j in permutation π_temp
  |   |   decreases the cost function then
5 |   |   |   reverse i and j;
6 |   |   end
7 |   end
8 end
9 Set π̃ = π_temp ;
```

is $O(kn\ log\ n)$, where k is the number of permutations to be aggregated and n the permutation length.

Algorithm 6.6 shows the approach, when using Next Best for optimization. Next Best functions as a local search technique, and local search methods in general, and the k-opt algorithms in particular [JM97] are among the best heuristics for solving the TSP. However, k-opt techniques cannot be used easily in our case since they are based on geometrical properties of the tour the salesman travels. The Next Best heuristic takes an initial solution, which can be generated randomly or using the techniques described above (e.g. Spearman Footrule Aggregation in Algorithm 6.4 or Barycenter Heuristic in Algorithm 6.5), and tries to optimize it locally by swapping the places of two positions in the result; if the gain in the cost function is higher than zero, then the solution is improved by accomplishing the swap of positions. Random Walk and Simulated Annealing [1] [JM97, KGV83] are two other optimization techniques, which were also implemented. Random Walk functions straightforwardly, by substituting the steps 2-8 of Algorithm 6.6 with the

[1] Simulated annealing tries to optimize difficult optimization problems by mimicking the process the atoms undergo in a metal when it is heated and then slowly cooled. This technique is unlikely to find the optimum solution, but it can often find very good solutions. The process of simulated annealing is controlled by the temperature T, which is slowly decreased. Furthermore, "bad" solutions, which decrease the value of cost function (when maximization is the purpose) by δ are accepted if the condition $e^{-\delta/T} > R$ holds, where R is a random number in $[0, 1]$. The general idea of the approach is that by allowing the selection of "bad" solutions, the method allows to explore more of the possible space.

lines:

generate random numbers i and j between 1 and n;
if *reversal of elements i and j in permutation* π_{temp} *decreases the cost function* **then**
  reverse i and j;

Simulated annealing adds to the improvement condition the simulated annealing condition, which allows for possibly worse states at the benefit of exploring more thoroughly the search space.

generate random numbers i and j between 1 and n;
if *reversal of elements i and j in permutation* π_{temp} *decreases the cost function or SA condition is fulfilled* **then**
  reverse i and j;

Whereas Random Walk and Simulated Annealing need a lot of time to converge, at the benefit of delivering better results, Next Best is faster. Considering that the techniques are used in the context of visualizing data interactively, some of the quality of the optimal solution can be traded off for achieving timely responses.

6.5.3.4 Discussion

Again, the techniques need to provide their results in a reasonable time, otherwise the user will experience delays in the visual exploration process. Thus, Algorithm 6.4 has a complexity of $O(kn + n\ log\ n)$ and the Barycenter heuristic in Algorithm 6.5 has a complexity of $O(2kn + k)$. The TSP heuristic presented in Algorithm 6.6 has a complexity of $O(kn\ log\ n)$ per iteration. Here k and n represent the number of permutations being aggregated and length of permutation respectively. The goodness of the aggregation algorithms can be judged both from the minimum sum of Kendall distances from the list of permutations and by considering the distance of the aggregate permutation from each permutation in the list. Figure 6.13 shows five time series presenting the distance of the respective aggregate permutation from each permutation in every time point. The aggregate permutation is calculated for each of them using Spearman footrule aggregation, Barycenter, TSP initialized randomly, TSP initialized using Spearman footrule and TSP initialized using Barycenter. The TSP heuristic itself is optimized using the Next Best approach. The sum of the distances for each of the approaches is 17126, 2842, 2767, 2767, and 2767, respectively. The results of the TSP

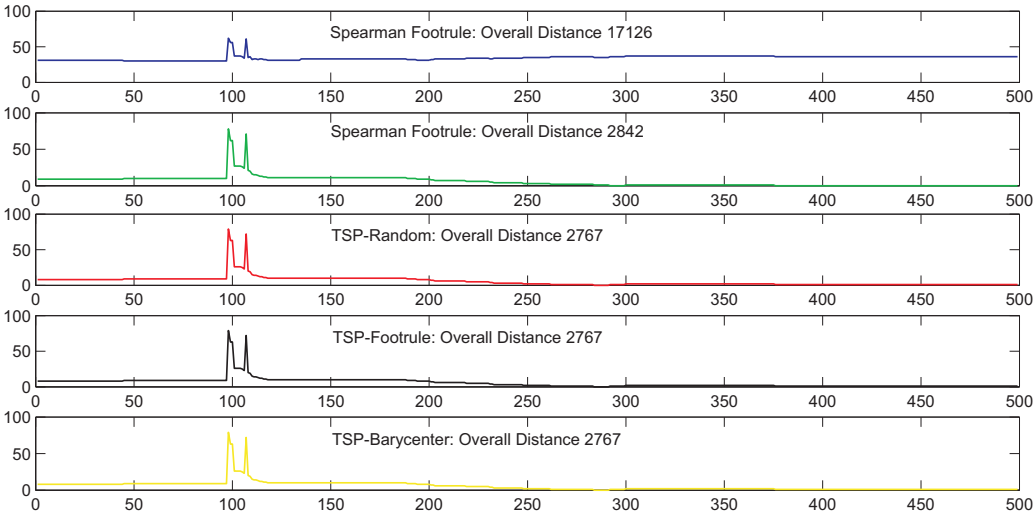


Figure 6.13: Comparison of the different heuristics for aggregation

heuristic variations are comparable with each other, independent of initialization, when considering the sum of distances. The Spearman Footrule has the worst results. All plots have the same pattern because of the effect of substrate pulse on the values of the sensitivity matrix.

6.6 Dimension Reductioning Techniques

The reorderable matrix and its derivatives visualize raw data (or its normalized forms) without distortions. In this context, they allow for viewing fine changes in objects. Sometimes, instead of viewing the raw data, a map of similarities is needed to create an idea what relationships exist between the objects in the data set. In this context, dimension reductioning techniques play an important role. In the following, we will discuss how such techniques can be used to visualize time-varying high-dimensional data.

6.6.1 Multi-Dimensional Scaling

Multi-dimensional scaling (MDS) is concerned with the construction of a configuration of n points in Euclidean space using information about the distances between these points. MDS is often used to project data nonlinearly from a high-dimensional space to a low-dimensional one, usually a 2D or 3D space. The purpose of MDS is that the distances between points in the lower-dimensional space approximate the distances between points in the higher-dimensional space best. MDS allows the use of similarity/dissimilarity measures instead of strict distances and thus enables to flexibly view the

Algorithm 6.7: General template of MDS algorithms

1 Preprocess the matrix D;
2 Project the data into lower dimensions;
3 Refine the solution in order to improve the cost function;

relationships between data items.

MDS in general defines a family of techniques (without specifying implementation details), which are focused on constructing configurations of n entities in Euclidean space using information about the distances between these entities (by entities we mean any multi-dimensional vectors). The origins of MDS are in psychology, where it is used for the study of proximity data for different entities. MDS methods are widely used in econometric and social sciences as well as other fields where data analysis is needed such as biology, chemistry, etc. MDS is in itself an exploratory method; it helps to provide insight into the relationships of the objects being studied and it is not about measuring something accurately.

For visually exploring high-dimensional data sets, reduction of dimensions is important because it enables the visual inspection of the data set in lower dimensions. There are quite often cases when the reduction of dimensions cannot be captured with linear methods such as *Principal Component Analysis* (PCA), where the projection is expressed as a linear combination of the components of the original data. MDS concentrates on non-linear projection methods. There are several approaches for reproducing nonlinear, high-dimensional data structures. Two large groups are distinguished: *Metric MDS* and *Non-metric MDS*. The difference between the two consists in the fact that metric MDS assumes that there is a true configuration of points in the low-dimensional space with the specified interpoint distances δ_{rs}. If the projected distance is d_{rs}, then metric MDS assumes that $d_{rs} = \delta_{rs} + e_{rs}$, where e_{rs} indicates the projection error. Non-metric MDS assumes a less rigid relationship between d_{rs} and δ_{rs}, namely that $d_{rs} = f(\delta_{rs} + e_{rs})$, where f is a monotone increasing function. In this way, non-metric MDS considers only the rank order between object distances. Algorithm 6.7 shows the general template of MDS algorithms. Such algorithms are usually composed of three steps: the preprocessing of the matrix, which might be a weighting procedure, the actual projection of the data, and the refinement of the solution. Some of the approaches might omit the first or third step, and sometimes both.

Classical MDS and its derivatives are a well known implementation of metric MDS. Non-metric MDS has several well known implementations which are mainly based in steepest descent optimization algorithms, and often the techniques differ in the cost function used. For both cases we will discuss implementation issues and our novel modifications in the following.

Algorithm 6.8: Classical MDS [Mar79]

Input: The distance matrix D

Output: The Solution of the Classical MDS

1 Preprocess the matrix D;
2 Construct the matrix A where $a_{ij} = (-\frac{1}{2}d_{ij}^2)$;
3 Construct the matrix B where $b_{ij} = a_{ij} - \overline{a_{i.}} - \overline{a_{.j}} + \overline{a_{..}}$ where $\overline{a_{i.}}$ is the mean of row i, $\overline{a_{.j}}$ is the mean of column j and $\overline{a_{..}}$ is the overall mean;
4 Compute the k largest eigenvalues λ_1, λ_2,... , λ_k where k is the dimension (in our case, $k = 2$ or $k = 3$);
5 Get the corresponding k eigenvectors v_1,... , v_k and normalize them by $v_i' \cdot v_i = \lambda_i$ and take the normalized eigenvectors as a solution of MDS ;

6.6.1.1 Classical Multi-Dimensional Scaling

Classical MDS is a technique based on linear algebra for achieving low-dimensional projections of data. It is similar in spirit to Principal Component Analysis; however in contrast to PCA the starting point is a distance matrix. PCA works with the covariance or correlation matrix.

The classical algorithm for metric multi-dimensional scaling [YH38, Sch35, Mar79] relies on the fact that the coordinates of the projection Z can be derived from the eigenvalue decomposition of the matrix $Z'Z$. As such, the distance matrix D should be converted in such a form and for this purpose, double centering is performed on the original distance matrix D in lines 2 and 3 of Algorithm 6.8.

The full pseudocode of the classical MDS algorithm is given in Algorithm 6.8. This algorithm has the advantage of being robust and fast, which ensures good response times with time-varying data. The performance of the algorithm is strongly dependent on the procedure for finding the eigenvalues of the distance matrices (calculated in step 3).

6.6.1.2 Sammon Mapping

Sammon Mapping [Sam69] is a non-metric MDS approach, which in contrast to MDS preserves smaller distances better than large ones. It tries to minimize an error criterion given by the following expression:

$$E = \frac{1}{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij}} \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \frac{(d_{ij} - \hat{d}_{ij})^2}{d_{ij}} \tag{6.12}$$

Here d_{ij} indicates the original distance and $\hat{d}_{ij}$ indicates the distance in the low-dimensional space.

For minimizing E in 6.12, any minimization technique can be used. Usually, the steepest descent technique [2] is used for this purpose, which proceeds iteratively in improving the solution using the formula

$$y(i+1)_{jk} = y(i)_{jk} + \alpha \frac{\frac{\delta E_s(i)}{\delta y_j k(i)}}{|\frac{\delta^2 E_s(i)}{\delta^2 y_j k(i)}|} \tag{6.13}$$

where $y(i)_{jk}$ presents the value of the k^{th} dimension of the j^{th} point after the i^{th} iteration. α is a constant with the value 0.3 or 0.4, described in the original version as the *magic factor*.

6.6.2 Projection of Sensitivity Matrices

The application of multi-dimensional scaling techniques to sensitivity matrices needs an intermediate step, namely the derivation of distance matrices from the raw sensitivity matrices. Based on the data type under consideration, different distance measures can be used. However, for sensitivity matrices, after the normalization process described in Section 6.3.2, Euclidean distance is used.

Algorithm 6.9 shows the proceedings of the classical MDS, as implemented in MetVis. The Sammon Mapping works as described in the previous section; its initialization is carried out using the solution generated by classical MDS.

Figure 6.14 shows the two respective views of the classical MDS and the Sammon Mapping. Small distances between parameters indicate larger similarities between them.

6.6.2.1 Time-Varying Similarity Measures

For taking into account time-varying features of data, other similarity measures can be used instead of the Euclidean distance. For this purpose, step 3 of Algorithm 6.9 can be modified to define cumulative similarity measures.

In this case, the distance $d_{ij} = \sqrt{c_{ii} - 2c_{ij} + c_{jj}}$, where c_{ij} is the cumulative correlation coefficient between two vectors. The cumulative information matrix of a set of time-varying matrices $X(t)$ is calculated as $I(t) = \sum_{k=T_{min}}^{T_{max}} X'(k).X(k)$. $I(t)$ is converted by normalization to a correlation matrix which is then used as a similarity measure that cumulatively considers time-varying matrices. The time window from T_{min} to T_{max} allows

[2] Gradient descent could be used also and in that case we do not need to compute the second order derivative. In that case $y(i+1)_{jk} = y(i)_{jk} + \alpha \frac{\delta E_s(i)}{\delta y_j k(i)}$

Algorithm 6.9: Template of the MDS algorithm

Input: The data matrix X_t representing the normalized time-varying matrices as described in section 3.1

Output: The Solution of the MDS

1 Let the vectors $X_1(t)...X_n(t)$ represent the columns of the matrix $X(t)$ at time t;
2 Compute the matrix $D(t)$ for $t = 1$ to T_{max}, where the elements d_{ij} of this matrix are calculated as follows:
3 $d_{ij} =\| X_i(t) - X_j(t) \|$;
4 Construct the matrix $A(t)$ where $a_{ij}(t) = (-\frac{1}{2}d_{ij}^2(t))$;
5 Construct the matrix B where $b_{ij} = a_{ij} - \overline{a_{i.}} - \overline{a_{.j}} + \overline{a_{..}}$ where $\overline{a_{i.}}$ is the mean of row i, $\overline{a_{.j}}$ is the mean of column j and $\overline{a_{..}}$ is the overall mean;
6 Compute the k largest eigenvalues $\lambda_1(t)$, $\lambda_2(t)$,... , $\lambda_k(t)$ where k is the dimension (in our case, $k = 2$ or $k = 3$);
7 Get the corresponding k eigenvectors $v_1(t)$,... , $v_k(t)$ and normalize them by $v_i'(t) \cdot v_i(t) = \lambda_i(t)$ and take the normalized eigenvectors as a solution of MDS for time t;

more flexibility for the users so that they can take into consideration a variable number of consecutive matrices, e.g. from the beginning to time point t, or only time point t, etc. To illustrate the step by an example, suppose we have two matrices in two consecutive points of time 1 and 2, $X(1)$ and $X(2)$:

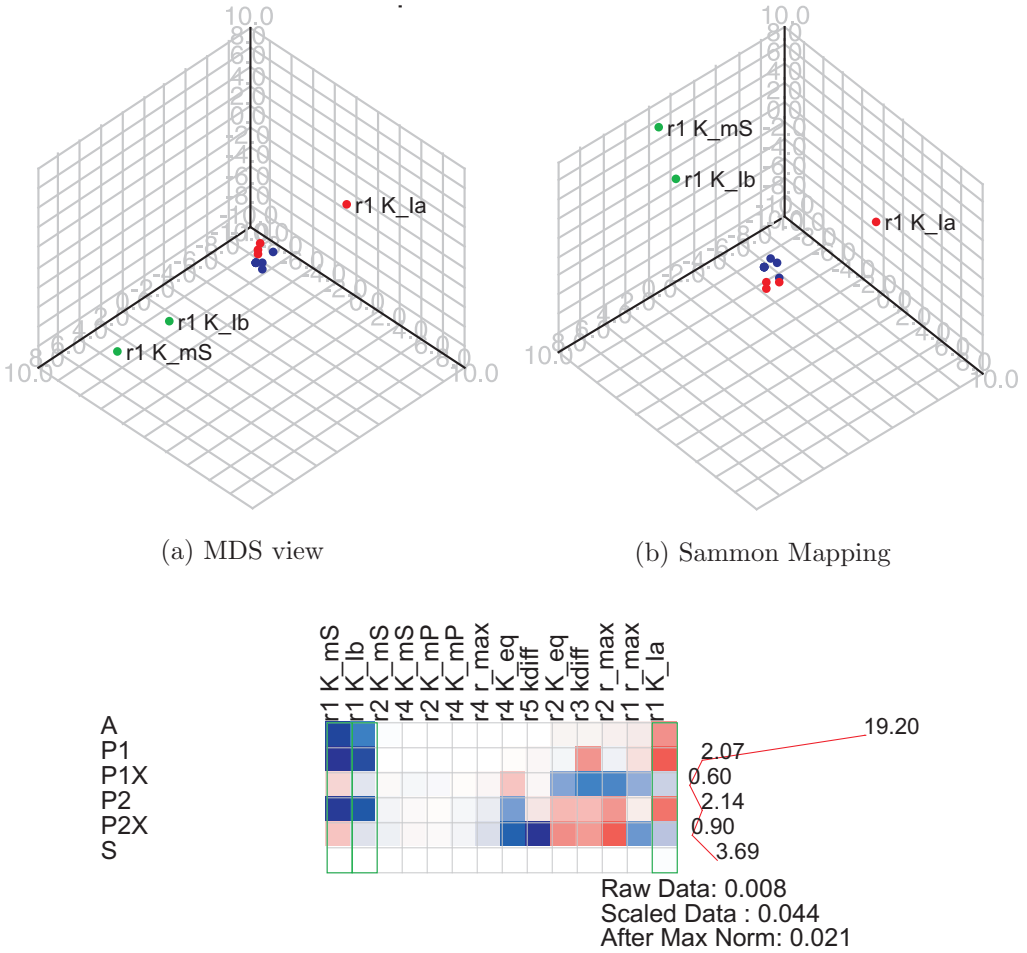
$$X(1) = \begin{pmatrix} 2 & 1 \\ 0 & 0 \\ 1 & 2 \end{pmatrix} \text{ and } X(2) = \begin{pmatrix} 2 & -1 \\ 0 & 0 \\ 1 & -2 \end{pmatrix} \tag{6.14a}$$

$$\text{Then } I(1) = \begin{pmatrix} 6 & 4 \\ 4 & 6 \end{pmatrix} \text{ and } I(2) = \begin{pmatrix} 12 & 3 \\ 3 & 12 \end{pmatrix} \tag{6.14b}$$

$I(1)$ contains information about time 1 whereas $I(2)$ contains information about time 1 and time 2. It is important that we cumulate the sums of $X(t) * X(t)$, otherwise if we directly add the matrices $X(1)$ and $X(2)$, the second column would be zero.

6.6.2.2 Comparison of Configurations: Procrustes Transformation

MDS, when applied to time-varying data, can possibly generate different configurations for different time points. Taking also into consideration that such configurations are insensitive (i.e. represent the same information) regardless of rotation and translation, a means for comparing such configurations



(a) MDS view

(b) Sammon Mapping

(c) The respective reorderable matrix

Figure 6.14: The MDS and Sammon Mapping views of the data in Figure 6.3(b)

is needed.

In this context, the Procrustes transformation [Sib78] can be used as a comparison tool between two configurations. In the original version, the Procrustes transformation is used as a goodness of fit measure, when the original configuration of data is available. We will use it to compare pairs of configurations for different time points.

The Procrustes transformation calculates a mapping and measures the similarity between two point sets. Its starting point are the two configurations represented by the vectors $x_1, ..., x_n$ (or the matrix $X_{n\times p}$) and $y_1, ..., y_n$ (or the matrix $Y_{n\times r}$). We can assume that $p \geq r$, and adding zeros for obtaining two configurations with the same number of points in the same dimensions.

How good the two configurations fit to each other can be measured by finding the minimum of sum of squares:

$$\sum_{i=1}^{n}(x_i - y_j)'(x_i - y_j) \tag{6.15}$$

Since MDS configurations are insensitive to rotation, reflection and translation, finding the minimum translates to the problem of finding the matrix $A_{p\times p}$, the scaling factor c and the vector $b_{1\times p}$, which achieve:

$$R^2 = min_{A,b}\sum_{i=1}^{n}(x_i - cA'y_j - b)'(x_i - cA'y_j - b) \tag{6.16}$$

Assuming that the points x_i and y_i are centered at the origin, A and b are found by least squares [Mar79, Sib78]. This assumption works for configurations created with classical MDS, but not with those created by non-metric MDS algorithms such as the Sammon Mapping. Furthermore, the procedure is not symmetrical with respect to X and Y and as such X is chosen as a reference configuration. The solution comes from the Singular Value Decomposition of the matrix $Z = Y'X$,

$$Z = V\Gamma U \tag{6.17}$$

The matrix A and vector b are then estimated as

$$A = VU' \tag{6.18}$$

$$b = \bar{x} - A'\bar{y} \tag{6.19}$$

where $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$ and $\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$. In case that scaling is also involved, the scaling factor c is calculated as

$$c = \frac{tr(\Gamma)}{tr(YY')} \tag{6.20}$$

6.6.3 Clustering and Projection

MDS techniques, although useful in exploring relationships between objects, do not deliver exact information about possible clusters in data. To achieve this purpose, clustering techniques which function independently of the dimension reduction procedure are needed. Chapter 7 focuses on the problem

Algorithm 6.10: The pseudocode of the k-means algorithm [Mac67, HW79]

Input: N objects to cluster x_i i=1..n; number of clusters K
Output: K clusters of objects $cluster_1, cluster_2, ..., cluster_K$

```
Select K initial cluster centroids c_1, c_2, ..., c_K;
repeat
    for i=1 to N do
        for k=i to K do
            d_ik = ||x_i − c_k||;
        end
        Assign object x_i to the cluster k with the minimum d_ik;
    end
    for k=i to K do
        Set S_k = {x | x ∈ cluster_k};
        c_k = (Σ_{∀x∈S_k} x) / |S_k|, where |S_k| is the cardinal of the set ;
    end
until convergence criteria is met;
```

of clustering time-varying sensitivity matrices. In this section, clustering is considered in the context of enhancing dimension reductioning techniques for better estimation of groups in the 2D or 3D visualizations. Thus, results of clustering are visualized directly in the projection and the user can judge about possible groups in data. For clustering the parameters, K-means a partitional clustering algorithm, is used.

K-means is a popular iterative descent algorithm for non-hierarchical clustering [Mac67, HW79]. It is used for quantitative data and tries to partition the data into K clusters, where K should be known beforehand. For each cluster, a mean feature vector is calculated and the algorithm tries to find the clusters with the minimum Euclidean distance with respect to this mean feature vector. Two problems arise when applying the K-means algorithm: 1) a change in the number of clusters could possibly bring quite different clusters and 2) the initialization of the model is crucial. The first problem is addressed using an interactive version of K-means: the user selects the number of clusters and the partition is then directly reflected in the respective projection view (MDS or Sammon Mapping). The initialization process is carried out either randomly or using principal component analysis as described in [Spä85]. Random initialization brings sometimes clearer results due to the fact that PCA initialization is often local minimum for the Sammon cost function.

In Figure 6.14, objects belonging to different clusters are plotted using different colors. From Figure 6.14(b) it is evident that clusters are visually

better separated by the Sammon Mapping than by the classical MDS. The number of clusters for this example is set $K = 3$. Parameters with lower sensitivity (which are in the middle of the reorderable matrix, visualized for comparison purposes in 6.14(c)), belong also to the same cluster. The highlighted parameters, $r1K_{mS}$ and $r1K_{Ib}$, which have high sensitivity but different effect on the metabolites, are highlighted in the figure. They belong to different clusters, and in the Sammon mapping view they have the largest pairwise distance.

6.6.4 Discussion

The presented methods for dimension reductioning allow the visual exploration of relationships between parameters in two or three dimensions. The goodness of these methods can be measured in two ways, namely by measuring how much the distances of the configuration in lower-dimensional space differ from the original distances of the parameters and by measuring how good these approaches separate possible groups/clusters existing in the parameter set.

The former is easy to measure and can be used as an overall measure of goodness for any projection technique. Measuring how good structures are preserved is more difficult, because we do not possess ground truth data related to any existing clusters in the parameters.

Figure 6.15 shows the normalized stress ($\frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(d_{ij}-\hat{d}_{ij})^2}{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}d_{ij}^2}$) and Sammon stress ($\frac{1}{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}d_{ij}}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{(d_{ij}-\hat{d}_{ij})^2}{d_{ij}}$) for classical MDS and the Sammon mapping. The vertical axis represents the stress whereas the horizontal axis represents the respective time points. Normalized stress(on the left) is more informative than the Sammon stress (right).

Classical MDS is more stable when normalized stress is considered. However, as the Sammon mapping focuses more on small distances, it preserves local structures present in data better.

6.7 The Symmetrical Reorderable Matrix

Section 6.5 handled the problem of visualizing data in form of matrices. It assumed that these matrices were asymmetrical i.e. they represented raw data, e.g. sensitivity matrices with different dimensions for parameters and metabolites. Sometimes, it is interesting to investigate another problem: visualizing symmetrical matrices which represent some kind of proximity in the raw data.

We will focus below on two such proximity measures: correlation/covariance

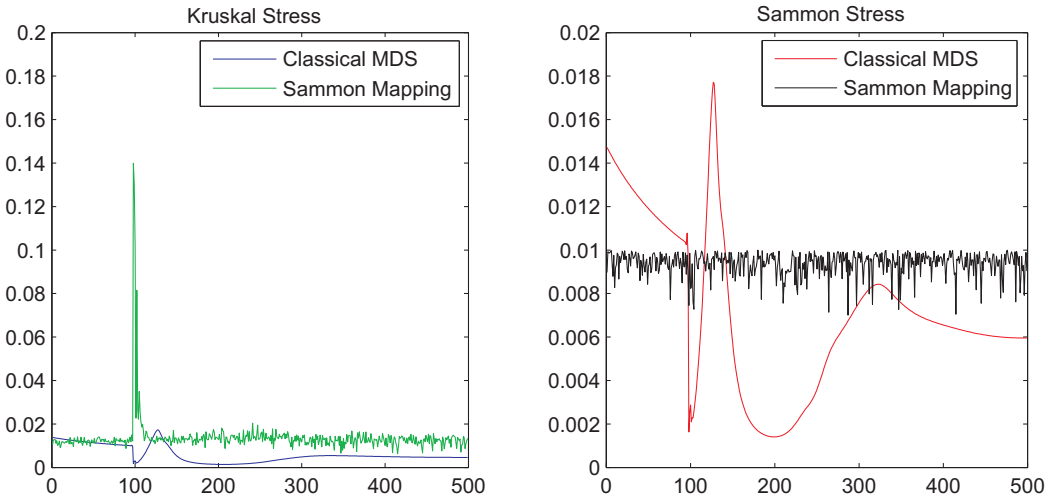


Figure 6.15: Comparison of the dimension reductioning techniques for 500 time points

data and cluster membership proximity. However, the techniques are generic in their nature and can be used for any other similar proximity measures.

6.7.1 Visualization of Proximity Matrices

In contrast to raw matrices, proximity matrices contain information regarding certain relationships between the objects in the data set. For example, the correlation matrix contains information about linear relationships between objects, the distance matrix contains information about distances between objects, etc. The focus is on symmetrical proximity matrices encountered when exploring relationships within a set of objects, in contrast to the *two mode* proximity matrices, which present proximities between two distinct object sets or other asymmetrical proximity measures.

To visualize such kind of matrices, the symmetrical reorderable matrix (SRM) method is defined, which is an extension of the previously introduced colored reorderable matrix for this special case. SRM differs from the reorderable matrix in two issues:

- In symmetrical matrices, if we change the order of a column of the matrix, the order of the respective row is also changed.
- Although some of the algorithms for reordering the raw matrix are also good heuristics for reordering symmetrix matrices, the two problems differ from each other.

The first issue is dealt with by interactively changing the order of the rows(columns) when the order of columns(rows) is changed.

Algorithm 6.11: Template of the VAT ordering algorithm [BH02]

Input: The dissimilarity matrix $R_{n\times n}$ corresponding to the set of images $O = \{o_1, ..., o_n\}$

Output: The ordered dissimilarity matrix $\widetilde{R}$

1 Set $K = \{1, 2, ..., n\}$; $I = J = \phi$; $P[0] = (0, ..., 0)$;
2 Select $(i, j) \in \arg\max_{p\in K, q\in K} R_{pq}$;
3 Set $P(1) = i; I = \{i\};$ and $J = K - \{i\}$;
4 **for** $r = 2$ *to* n **do**
5 | Select $(i, j) \in \arg\min_{p\in I, q\in J} R_{pq}$;
6 | Set $P(r) = j$; Replace $I = I \bigcup \{j\};$ and $J = J - \{j\}$;
7 | $r = r + 1$;
8 **end**
9 Obtain the ordered dissimilarity matrix $\widetilde{R}$ using the ordering array P as: $\widetilde{R}_{ij} = R_{P(i)P(j)}$, for $1 \leq i, j \leq n$.

Dealing with reordering of proximity matrices on the other side, is generally more difficult than with the simple reorderable matrix. Formally, reordering the proximity matrix would mean to find a permutation π, which minimizes

$$\Gamma(\pi) = \sum_{i,j} d(\pi(i), \pi(j)) \tag{6.21}$$

This is a special case of the *quadratic assignment problem* [GJ79, PRW94], where the weight function is the matrix W with values being 1 overall except the main diagonal. In the reorganization of the reorderable matrix we were looking only for minimum sum of consecutive objects in a permutation. This problem is also NP-hard [GJ79] and the respective decision problem NP-complete.

A heuristic for reorganizing proximity matrices is presented by Bezdek and Hathaway [BH02], who use a variant of Prim's algorithm for finding minimal spanning trees (MST) for ordering *dissimilarity images* presented in Algorithm 6.11. The algorithm does not compute the MST itself; it just keeps the order of vertices as they are added in the MST. It starts with an object which is at one end of an edge with maximum length, so as not to divide possible groups of objects into several parts. Some examples with Visual Assessment of cluster Tendency (VAT) generated orderings will be given in the following sections.

The TSP-based heuristics used in reordering raw matrices and in aggregating permutations were extended further to consider the new cost function defined in Equation 6.21 as described in Algorithm 6.12. The same comments about modification of the local search procedure in Algorithm 6.6 in Section

Algorithm 6.12: TSP heuristic for reordering proximity matrices

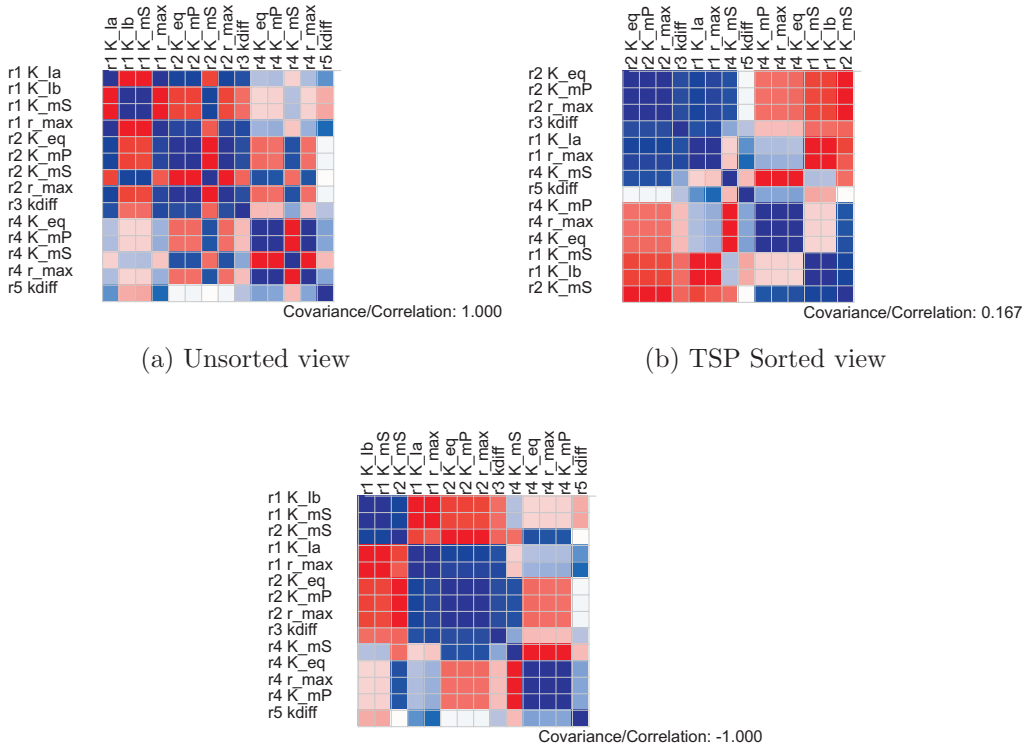
Input: The proximity matrix $X_{n\times n}$
Output: The optimized permutation $\tilde{\pi}$
`/* The cost function is defined as` $\Gamma(\pi) = \sum_{i,j} d(\pi(i), \pi(j))$ `*/`
Initialize π_{temp} randomly or using the permutation generated by VAT in Algorithm 6.11
`/* Permute the solution locally until no improvement is found */`
for $j = 1$ *to* n **do**
  for $i = 1$ *to* n **do**
    if *reversal of elements* i *and* j *in permutation* π_{temp} *decreases the cost function* **then**
      reverse i and j;
    end
  end
end
Set $\tilde{\pi} = \pi_{temp}$;

6.5.3.3 are valid also for Algorithm 6.12.

6.7.2 Reorderable Covariance/Correlation Matrix

Covariance and correlation are closely related parameters that indicate the extent to which two random variables co-vary. In our case they give us a measure on how the parameters of the model are related to each other. Whereas the approach described in [Fri02] is a good step forward in the exploration of correlation matrices, we have implemented a similar approach containing two additional aspects: (a) Prim's ordering algorithm for proximity matrices is implemented, and (b) interaction with the user is allowed. Figure 6.16 shows the reorderable correlation matrix for the data in Figure 2.1(a). Figure 6.16(a) shows the initial unordered correlation matrix and Figure 6.16(b) shows the same matrix after applying Prim's algorithm. In Figure 6.16(c), this ordering is then changed by iterative click-and-drag steps to obtain a more satisfiable ordering.

The data in Figure 6.16(b) is divided into three large groups of correlated parameters. Although some of the information present in Figure 6.16(c) could also be extracted from Figure 6.10, the correlation matrix emphasizes it in a much better way, for example the parameters $r1K_{Ia}$ and $r1K_{Ib}$ are correlated negatively with each other.



(a) Unsorted view

(b) TSP Sorted view

(c) Sorted view after manual permutation

Figure 6.16: The reorderable correlation matrix of the data in Figure 6.3(b). After sorting the parameters, three large groups of correlated parameters become visible.

6.7.3 Reorderable Cluster Membership Evolution Matrix

Partitional clustering algorithms such as K-means, which was described shortly in Section 6.6.3, divide the objects of analysis into non-overlapping groups of objects similar to each other. Since we have time-varying data, it is interesting to find out how the found clusters evolve over time.

Usually, partitional algorithms create an assignment array A containing information about cluster membership for each object clustered. For a run of K-means for n objects, A[i] would have a value between 0 and K-1 for i=1..n. Such assignment vectors might have the form (1,1,1,1,0,1,1,1,1,1,1,0,1,0) in the case when we run K-means with K=2, meaning that only three elements belong to the cluster labeled as cluster 0; the rest belongs to the cluster labeled as cluster 1.

The vector representation given above does not help much in this case

since (1,1,1,1,0,1,1,1,1,1,1,0,1,0) and (0,0,0,0,1,0,0,0,0,0,0,1,0,1) give us the same information; only the cluster labels have changed. For this reason, we transformed the assignment vectors into what we call a membership adjacency matrix. This matrix contains a 1 if the two objects are in the same cluster, 0 otherwise. An element is considered as being not in the same cluster with itself since this is not important for our application. It can be verified that for both assignment vectors the same matrix would result. Another advantage of this matrix representation is that these matrices can be cumulated for different segments of time, resulting in matrices that describe how often the respective objects were in the same cluster for the selected time segment.

Algorithm 20 shows the pseudocode of the above described procedure. The matrix in Figure 6.17(b) shows the result of cumulation for the entire time range for the example used for illustration. The advantage of this cumulation matrix is that it contains information about cluster membership for all points of time. However, the numerical values in this matrix are difficult to interpret as the dimensions of the matrix grow. For this reason, this matrix is transformed into a frequency matrix with values in [0,1] by dividing the values by the number of points of time which were used in the cumulation. This transformed matrix is then visualized as a reorderable matrix, which we call 'Cluster Membership Evolution Matrix', as shown in Figure 6.17.

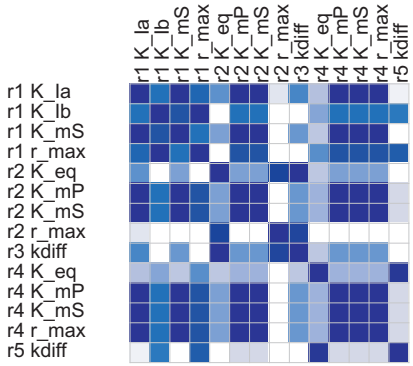
Figure 6.17(a) represents the cumulative adjacency matrix where the time interval from $t = 0$ to $t = 10s$ is taken into consideration. The number of clusters for each point of time is three in this case. Figure 6.17(b) represents the corresponding colored visualization. Figure 6.17(c) is the result of reordering of columns with exhaustive spectral ordering. Figure 6.17(e) which differs from Figure 6.17(c) in the fact that it was generated using four clusters instead of three. Possible outliers in this context, i.e. parameters that change clusters frequently over all points of time are represented by columns/rows visualized with lighter colors. The higher granularity implied by the colors in Figure 6.17(e) suggest that three clusters are more appropriate. Figure 6.17(d) presents the VAT generated ordering, which is provided for comparison purposes. Though K-Means is used as the input of the Cluster Evolution Algorithm, any other partitional clustering algorithm could be used instead of K-Means.

6.7.4 Discussion

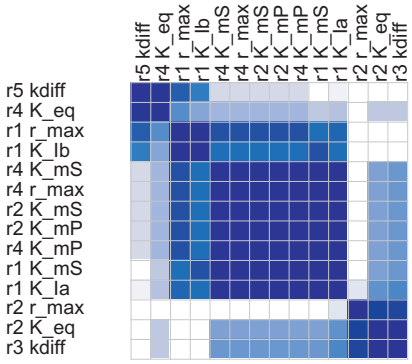
The visualization methods presented in this section serve both as complementary views to the previous approaches as well as stand alone visualization techniques for proximity data. They allow for discovering structures and clusters (or their absence) in a natural way. The reorderable correlation matrix view is important not only in the context of exploring correlations be-

	r1 K_la	r1 K_lb	r1 K_mS	r1 r_max	r2 K_eq	r2 K_mP	r2 K_mS	r2 r_max	r3 kdiff	r4 K_eq	r4 K_mP	r4 K_mS	r4 r_max	r5 kdiff
r1 K_la	1	0.476	0.913	0.505	0.359	0.942	0.942	0.058	0.398	0.175	0.942	0.942	0.942	0.029
r1 K_lb	0.476	1	0.563	0.913	0	0.476	0.476	0	0	0.291	0.476	0.476	0.476	0.437
r1 K_mS	0.913	0.563	1	0.476	0.301	0.913	0.913	0	0.34	0.146	0.913	0.913	0.913	0
r1 r_max	0.505	0.913	0.476	1	0	0.563	0.563	0	0	0.379	0.563	0.563	0.563	0.524
r2 K_eq	0.359	0	0.301	0	1	0.301	0.301	0.699	0.961	0.146	0.301	0.301	0.301	0
r2 K_mP	0.942	0.476	0.913	0.563	0.301	1	1	0	0.34	0.233	1	1	1	0.087
r2 K_mS	0.942	0.476	0.913	0.563	0.301	1	1	0	0.34	0.233	1	1	1	0.087
r2 r_max	0.058	0	0	0	0.699	0	0	1	0.66	0	0	0	0	0
r3 kdiff	0.398	0	0.34	0	0.961	0.34	0.34	0.66	1	0.146	0.34	0.34	0.34	0
r4 K_eq	0.175	0.291	0.146	0.379	0.146	0.233	0.233	0	0.146	1	0.233	0.233	0.233	0.854
r4 K_mP	0.942	0.476	0.913	0.563	0.301	1	1	0	0.34	0.233	1	1	1	0.087
r4 K_mS	0.942	0.476	0.913	0.563	0.301	1	1	0	0.34	0.233	1	1	1	0.087
r4 r_max	0.942	0.476	0.913	0.563	0.301	1	1	0	0.34	0.233	1	1	1	0.087
r5 kdiff	0.029	0.437	0	0.524	0	0.087	0.087	0	0	0.854	0.087	0.087	0.087	1

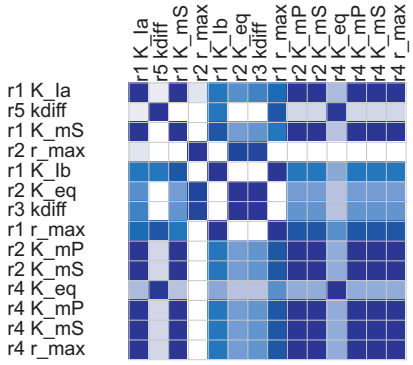
(a) Matrix of frequencies of cluster adjacency memberships



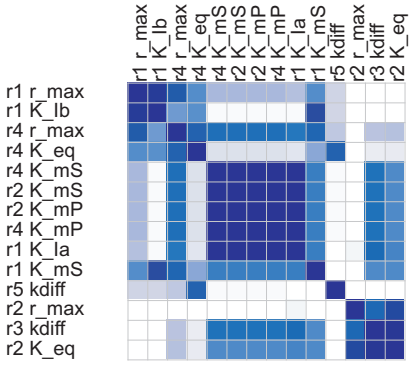
(b) Unsorted cluster membership evolution matrix



(c) Cluster membership evolution matrix sorted with Exhaustive Spectral Sorting (Algorithm 6.3)



(d) Sorted view using VAT algorithm



(e) Sorted cluster membership evolution matrix (aggregating clustering results for K-Means with K=4)

Figure 6.17: The reorderable cluster membership evolution matrix accumulating results from $t = 0s$ to $t = 10s$

Algorithm 6.13: Algorithm for displaying cluster evolution

Input: Data $X_{mxn}(t)$
Output: Visualization of Cluster Evolution

```
1  Initialize all elements of A with 0;
2  for t=1 to time_max do
3  |  for i=1 to n do
4  |  |  for j=i+1 to n do
5  |  |  |  if samecluster(i,j) then
6  |  |  |  |  C[i,j]=1;
7  |  |  |  end
8  |  |  end
9  |  end
10 |  A=1/t((t-1)A+C);
11 |  Update the Reorderable Matrix Display of A;
12 end
```

tween parameters of the metabolic network but any other correlation matrix. The reorderable cluster membership matrix on the other side is tailored for time-varying multi-dimensional data. Inspired from this approach, clustering techniques adapted for time-varying multi-dimensional data (e.g. sensitivity matrices) based on cluster ensembles are presented in Chapter 7.

6.8 Visualization of Sensitivity Results for the E. coli Model

The E. coli model presented in Chapter 5 was analyzed thoroughly using MatVis in [Noa05]. Some of the results of this analysis related to the visualization of sensitivity matrices will be discussed below.

The network analyzed considers 18 metabolites and 30 reactions, whose relationships are described with mechanistic models. In total, 116 parameters are present, resulting in sensitivity matrices describing the influence of metabolites from parameters of dimensions 18×116 for every time point. The sensitivity matrices describing the dependency of reaction rates from parameters are of dimensions 30×116 for every time point.

6.8.1 Visualization of sensitivity matrices

Figure 6.18 shows the respective sensitivity matrices both with respect to metabolites (above) and fluxes (below). From the figure, many parameters

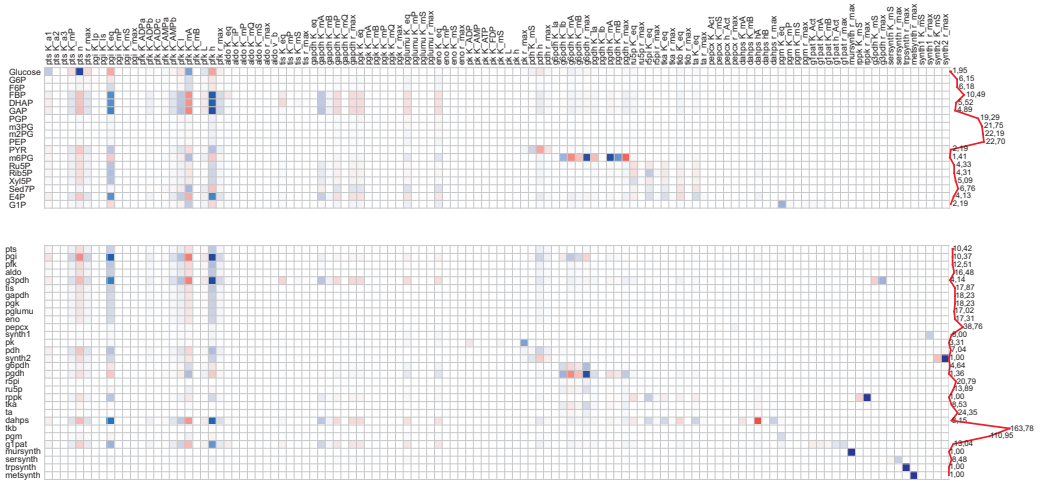


Figure 6.18: Sensitivity matrices for fluxes and metabolites of the E. coli model for the stationary state

exhibit only a small influence on the respective metabolites or fluxes. Furthermore, some metabolites and reaction rates are sensitive to changes of the same parameter and several other parameters affect either metabolites or reaction rates. To make the analysis of such large data sets easier for the user, some of this parameters which show low sensitivity values can be excluded from further examination. Techniques for filtering these visualizations will be described in detail in Section 8.3.

Figure 6.18 shows the sensitivity matrix with respect to metabolites after the removal of 74 parameters. The most sensitive parameters are highlighted in green. These parameters are part of the following reactions: Phosphotransferase (Pts), Phosphoglucoisomerase (Pgi) and Phosphofructokinase (Pfk). Their changes influence the metabolites which are products of specific reactions, in this case Dihydroxyacetonphosphate (DHAP) and Glycerinaldehydphosphate (GAP). A concentration change in GAP would bring a change also in the concentration of Erythrose-4-phosphate (E4P) (the Pentose-Phosphate-Pathway in Figure 4.11 clarifies the relationships between the two metabolites). From the reordering, the parameters are divided into three large groups at the left, at the right and at the center of the matrix. From the colors of the boxes, it is clear that the sensitivities in the two groups differ by their sign, i.e. for the positive sensitivity in one group there is a negative sensitivity in the other group. The maximum norm drawn at the right of the matrix indicates that metabolites PGP, 3PG, 3PG and PEP are affected more by the changes in parameters. However, these changes are not visible at this time point because they come after the substrate pulse is given.

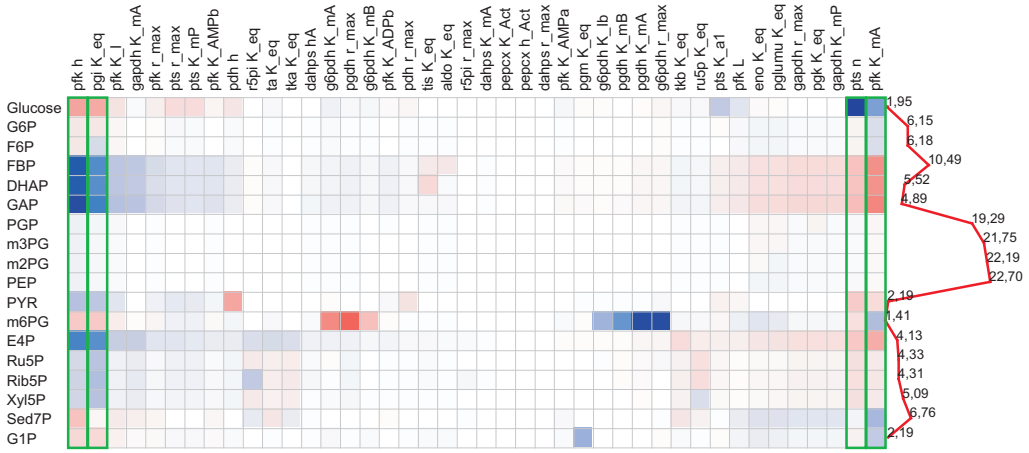


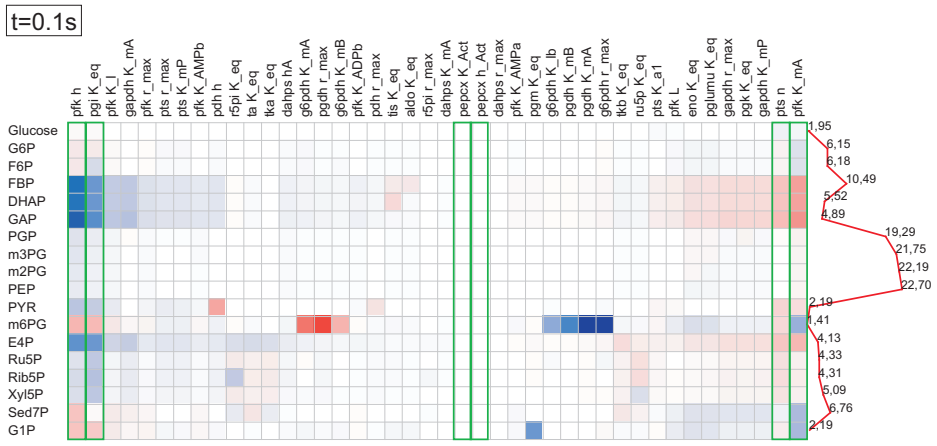
Figure 6.19: Reduced sensitivity matrix with respect to metabolites for the E. coli model

6.8.2 Dynamic sensitivities

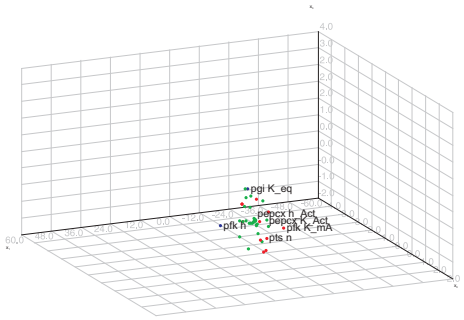
Figure 6.20 and Figure 6.21 show the reorderable matrices and the respective Sammon Mapping projections for four selected points of time.

At the time point $t = 0.1s$, which is directly after the pulse is given, the visualization is quite similar to the data in stationary state shown in Figure 6.19. Three clusters are visualized in the Sammon Mapping view which correspond roughly to three zones in the reordered matrix, namely to the left, middle and right of the matrix. The left and right group possess a higher sensitivity compared to the middle group. Focusing on the reorderable matrix (Figure 6.20(a)), on the second column representing parameter pgi_keq, which controls the transformation of $G6P$ into $F6P$, an increase of this parameter would affect all the consecutive metabolite concentrations (FBP, DHAP, GAP). On the other side, the high consumption of $G6P$ would mean that less $G6P$ is available to be transformed into $6PG$, hence there is a negative sensitivity of pgi_keq with respect to $6PG$. Figure 6.23, which presents the cluster membership evolution for the time interval $t = 0s$ to $t = 10s$, indicates that the parameters pgi_keq and $pfk\ h$ belong to the same cluster for this period of time; similarly $pts\ n$ and $pfk\ K\_mA$ belong to the same cluster (different from the previous). These parameters represent two groups of highly sensitive parameters, which differ from each other only from the sign and are highlighted in Figure 6.23. The other group represents relatively unsensitive parameters.

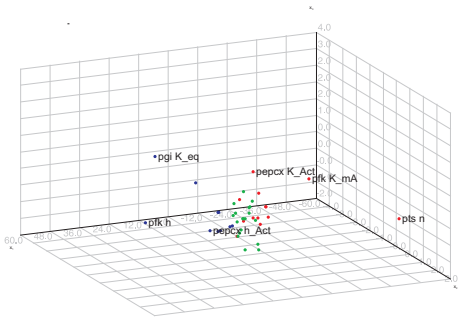
At the time point $t = 23.0s$, the sensitivities change thoroughly compared to the stationary state ones. The sensitivity values of nearly all parameters are increased, which is reflected in the increased distances in the Sammon Mapping view in Figure 6.20(c). Furthermore, the sign of some similarity



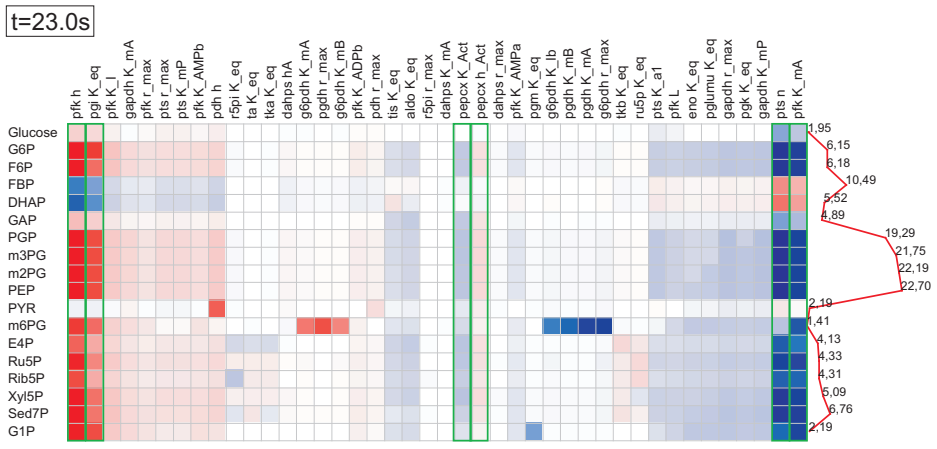
(a) Matrix for time t=0.1s



(b) Sammon mapping for t=0.1s

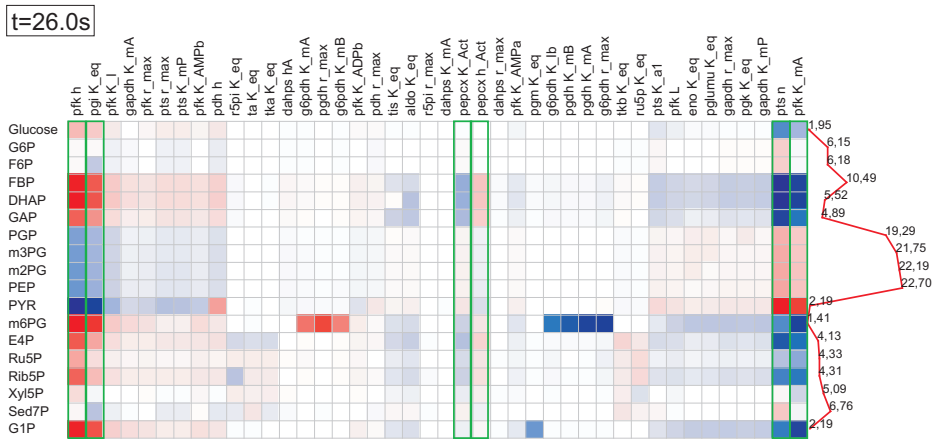


(c) Sammon mapping for t=23.0s

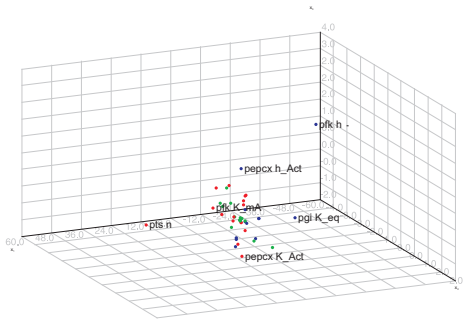


(d) Matrix for time t=23.0s

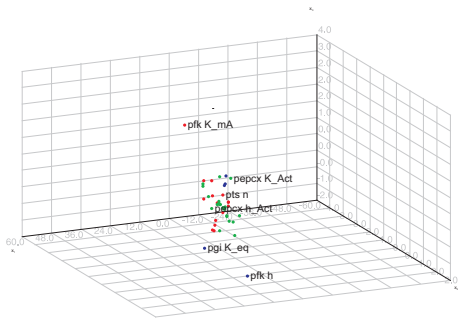
Figure 6.20: Reorderable matrix and Sammon Mapping for two time points



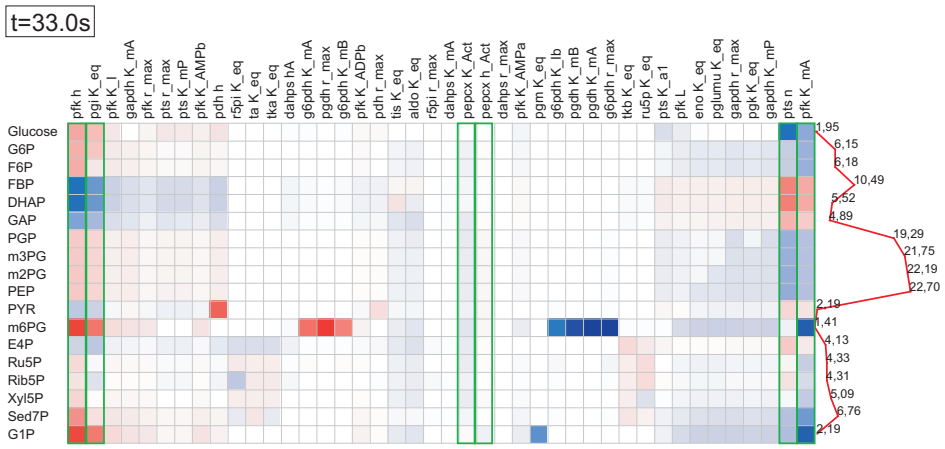
(a) Matrix for time t=26.0s



(b) Sammon mapping for t=26.0s

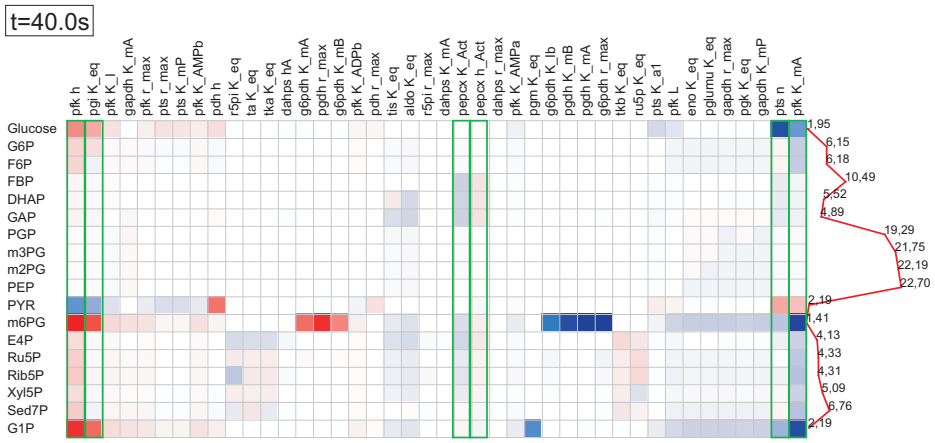


(c) Sammon mapping for t=33.0s

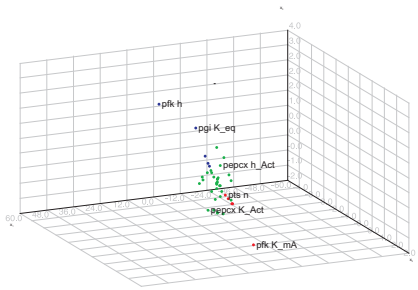


(d) Matrix for time t=33.0s

Figure 6.21: Reorderable matrix and Sammon Mapping for two time points



(a) Matrix for time t=40.0s



(b) Sammon mapping for t=40.0s

Figure 6.22: Reorderable matrix and Sammon Mapping for one time point

values is reversed in this time point compared to the stationary state. For example, an increase in *pgi_keq* brings now a decrease of the concentration of $F6P$, whereas in the stationary state it brought an increase in the same concentration. This fact seems contradictory in the first sight but it is explained with the activation effects of PEP and FBP (see Figure 4.11). The increase in FBP increases also the activation effect of enzyme $PepCx$, which results in greater consumption of PEP. In this way, the inhibition effect of PEP on the enzyme Phosphofructokinase (Pfk) is decreased, resulting in increased transformation of $F6P$ into FBP (implying the decrease of $F6P$ concentration). For $t = 26.0s$, a different configuration is encountered. The decrease of FBP concentration decreases also its activator effect, which decreases also the transformation of PEP. This by itself increases the inhibition effect on the Pfk enzyme, explaining the high sensitivity of *pgi_keq* with respect to FBP, $DHAP$ and GAP.

The sensitivities for $t = 33.0s$ are quite similar to those for $t = 23.0s$ and

result from some kind of oscillation in the system. However, the amplitude of values in this point of time is lower than for $t = 23.0$, which is reflected in smaller distances in the Sammon Mapping view.

Near the end of simulation, the sensitivity values presented in Figure 6.22 are decreased and a new stationary state is reached.

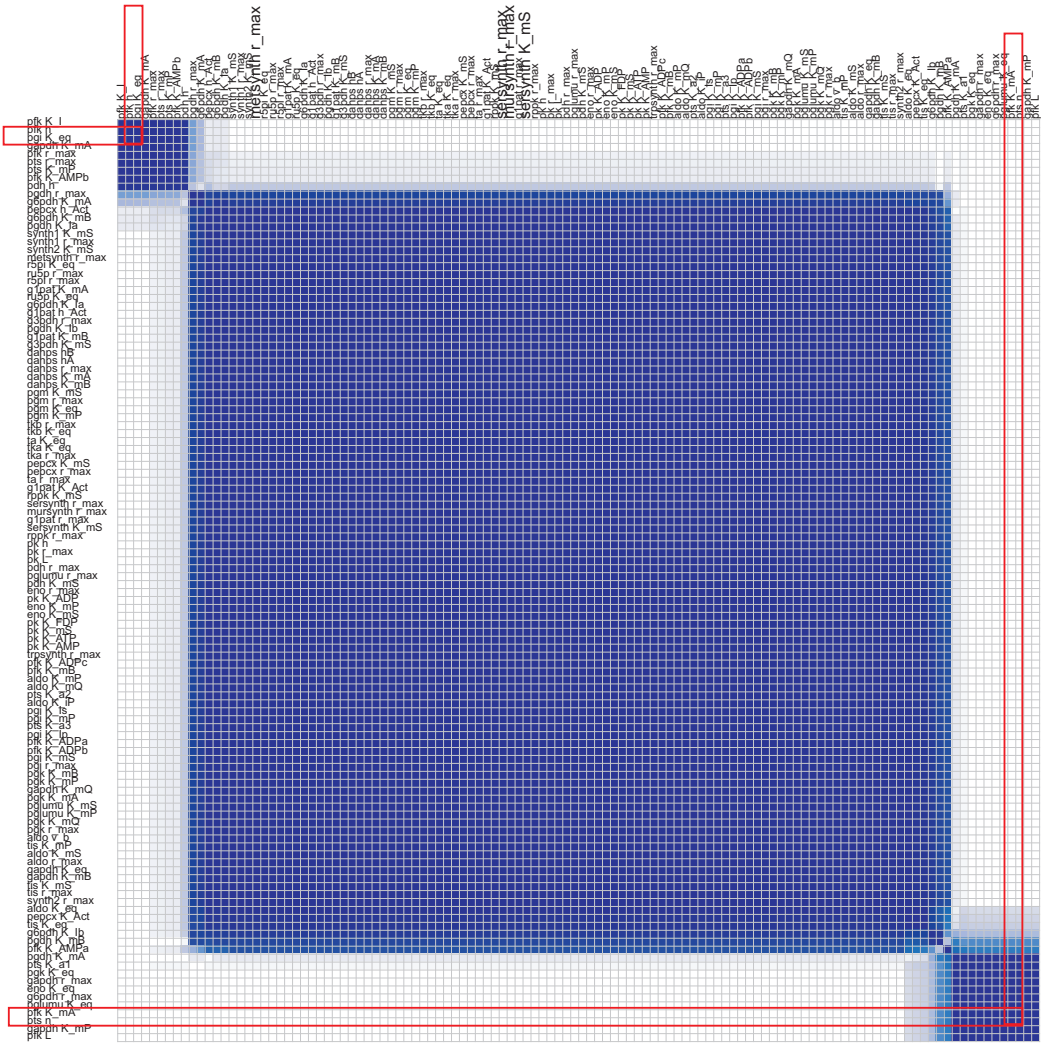


Figure 6.23: The reorderable cluster membership evolution matrix for the sensitivity matrices of the E. coli model aggregated for the interval $t = 0$ to $t = 23s$

6.9 Summary

In this chapter, different techniques dealing with visualization of time-varying multi-dimensional data in general and sensitivity matrices in particular were

presented. First, sensitivity analysis and its output, sensitivity matrices were explained in the context of metabolic modeling. Then the MatVis toolkit, consisting of several novel methods and extensions to existing approaches to visualize time-varying matrices was introduced.

The asymmetrical reorderable matrix, and issues related to its visualization and reordering were described. Furthermore, problems and solutions for global reordering of the time-dependent reorderable matrix for evolving multi-dimensional data were also discussed.

Dimension reductioning was also discussed in this context, and dimension reduction techniques, such as multi-dimensional scaling and the Sammon mapping, were combined with an interactive version of the K-means clustering algorithm in order to create a graspable view of the multi-dimensional data and the possible clusters and groups in the data.

Finally, the symmetrical reorderable matrix, a novel method for visualizing time-varying proximity data was also presented. Two examples, namely the reorderable correlation/covariance matrix for visualizing the time-varying covariances and correlations and the cluster membership evolution view enabling the user to distinguish clusters or parts of them which are consistent within all time points were proposed.

The benefits of the different methods were illustrated by visualizing sensitivity matrices generated during the simulation of metabolic network models.

7
Clustering Algorithms for Time-Varying Data

7.1 Introduction

The aim of clustering algorithms is to partition a data set into groups of similar objects, where objects inside the same cluster are more closely related to each other than objects assigned to different clusters. In contrast to (supervised) classification, clustering is a data driven approach that does not require class labels. Cluster algorithms are grouped into hierarchical and partitional algorithms. The latter are grouped into hard clustering methods and fuzzy clustering methods.

Since different clustering algorithms behave differently with different data sets and very often behave differently with the same data set, cluster ensembles have been proposed [SG02, Fre01, GMT05]. The main purpose of cluster ensembles is to merge the results of several clusterings together in order to obtain a partitioning which ideally is robust and improves the quality of the clusters. In general, cluster ensemble algorithms consist of the following steps:

1. Obtain the different partitions of objects;

2. Build a similarity matrix for each partition;

3. Merge the matrices based on similarity;

4. Find the final partitioning using the merged similarity matrix.

So far, cluster ensembles have either been used to improve the clustering results when applied to static data sets or they have been used to merge together multiple clusterings of the same objects described by different sets of features.

In this chapter, we investigate the problem of clustering time-varying multidimensional data using cluster ensembles. Whereas in Chapter 6 clustering was used as an enhancement of dimensional reductioning techniques for understanding better the low dimensional plots, in this chapter the focus is on clustering frameworks which generate aggregate clusterings for time-varying data. For clustering such data, a novel fuzzy cluster ensemble is presented, which is based on the combination of different fuzzy partitionings, in contrast to previous ensemble clustering approaches where only hard partitionings were considered. Several problems encountered in this context are addressed in this chapter, namely: (I) defining appropriate similarity measures between two objects based on a given fuzzy partition; (II) merging the calculated similarity matrices with each other; and (III) partitioning the resulting matrix into clusters.

Time-varying multidimensional data can be thought of as an (ordered) sequence of multidimensional data. Such data often occurs, e.g. in the analysis of (a) time dependent models in biology (i.e. sensitivity analysis of metabolic models), (b) video or audio streams, (c) gene expression data over different time frames, (d) patient data for certain periods of time, (e) logs of multivariate sensors. In this context, we can assume that data will be processed incrementally. Thus, it is convenient to keep only a summary of the past data in order to be able to process future data since such data sets can get very large in size and cannot fit in main memory anymore. Consequently, we propose to use cluster ensembles, which fit perfectly because the summary of the past data can be represented naturally by an accumulated similarity matrix, calculated during the ensemble process. The accumulated similarity matrix contains information about cluster membership for the past and current data and as such is suitable to generate clusters which take into consideration the evolving memberships.

To demonstrate the performance and robustness of the the approach, we report comparative results based on synthetic test sets as well as time varying sensitivity matrices. The fuzzy cluster ensemble for time-varying data delivers better results on the average than K-Means or Fuzzy C-Means and it has also a smaller variance, while it increases the running time only slightly. Furthermore, the application of FCM ensemble for analysis of time-varying sensivity matrices is discussed.

The remainder of this chapter is organized as follows. Section 7.2 reviews current approaches for cluster ensembles. In Section 7.3, the proposed fuzzy clustering ensemble approach is described. Section 7.4 presents comparative results. Section 7.4.5 discusses application of ensemble for sensitivity matrices

analysis. Section 7.5 concludes the chapter and outlines areas for future research.

7.2 Related Work

A survey of data clustering algorithms can be found in [JMF99]. The idea of using cluster ensembles in cluster analysis is inspired by supervised learning approaches, where classification ensembles had been used earlier.

Fred [Fre01] presents an ensemble scheme called the *voting-k-means* algorithm. The algorithm is based on a co-association matrix which collects information from different runs of the k-means algorithm. A fixed threshold based voting scheme is then used to partition this matrix into the final clusters. This idea was developed further under the name *evidence accumulation* [FJ02b, FJ02a, FJ05] where the co-association matrix is used as the input to a hierarchical clustering algorithm.

Strehl and Ghosh [SG02, SGM02] consider three approaches for solving the cluster ensemble problem. All three approaches basically consist of the following steps: several runs of different clustering algorithms are performed; based on these clusterings, a similarity graph is constructed; a graph partitioning algorithm is executed to obtain the final clusters. One of the approaches is based on a co-association matrix similar to Fred's approach [Fre01]. The other two model the similarity between points as a hypergraph. The authors mention that when applying the *cluster-based similarity partitioning algorithm (CSPA)*, a soft clustering membership matrix could be used instead of the hard one, without going into detail how this step can be realized [SG02].

Yang and Kamel [YK03] apply *swarm intelligence* techniques to the cluster ensemble problem by using ant colony algorithms representing different clusterings.

Gionis et al. [GMT05] convert the problem of finding the final clustering with a minimum number of total disagreements with existing clusterings into the problem of *correlation clustering* and use algorithms applied for the latter to solve the original problem.

Fern and Brodley [FB03] combine clusterings of random projections of high dimensional data to reduce instability and improve the performance of the final cluster results. They discuss how the cluster ensemble problem could be solved by using bipartite graph partitioning [FB04].

Zeng et al. [ZTGFR02] consider the problem under a slightly different point of view, namely as a meta-clustering problem where the final purpose is to merge the results of different clustering techniques such as k-means, SOM etc. Hard clustering results are converted into fuzzy ones by making the assumption that the density of points inside a cluster follows a Gaussian distribution and the Euclidean distance is used to calculate the similarity of

two objects in the cluster space. The similarity matrices are then combined with each other (averaged) and the final result is obtained using a hierarchical clustering algorithm. However, this approach makes use of the original features of the objects in the data set. Furthermore, the Euclidean distance used for measuring similarity sometimes performs worse than other distance measures. Hu and Yoo [HY04] follow the same idea, but use a graph partitioning algorithm to obtain the clusters instead of the hierarchical clustering algorithm.

A related concept to the time-varying multidimensional data we study is that of *data streams*. Clustering data streams is treated by several authors. Guha et al. [GMM+03] present theoretical and practical results related to the clustering of data streams. Furthermore, they present a single pass algorithm for clustering streaming data, which is based on a facility location algorithm. Aggarwal et al. [AHWY03, AHWY04, AHJY05] have also considered the clustering problem for data streams. A complex approach for clustering data streams is presented, which is concepted as a framework composed of two parts: an online component dealing with the storage of detailed summary statistics about the stream and an offline component which is utilized by an analyst to accomplish a customizable clustering process [AHWY03]. Common user inputs are the time horizon or the number of clusters. Furthermore, they treat the problem of projected clustering of high dimensional data streams [AHWY04]. Projected clusters represent clusters which are meaningful in a subset of dimensions due to the sparsity of the data in high dimensions.

7.3 Fuzzy Ensemble Clustering

To improve the robustness of clustering time-varying data, the basic idea of the proposed *Ensemble Fuzzy C-Means* (Ensemble-FCM) approach is to combine the fuzzy partitions as they are created by Fuzzy C-Means (FCM) incrementally into a symmetric proximity matrix. However, the idea could be extended further to include hard clusterings by using proper similarity measures between objects partitioned into clusters or by fuzzifying hard clustering partitions, although fuzzification could require access to the original features of the data set, as described by Zeng et al. [ZTGFR02], in contrast to our approach which considers only the fuzzy partition matrix. By providing Ensemble-FCM with a memory horizon (in our case t), the stability and robustness of the clustering is increased, especially when abrupt changes in data do not allow proper clustering at certain points of time.

Algorithm 7.1 gives an overview of the steps of our proposed fuzzy cluster ensemble approach for time-varying data. The same algorithm can be used also as a fuzzy cluster ensemble for static data, by feeding a certain number

Algorithm 7.1: Template of the Fuzzy Clustering Ensemble algorithm

Input: The data matrix X_t representing the information about n objects to cluster in time t
Input: The number of clusters $c^{(Ensemble)}$
Output: The Cluster Ensemble Partition

```
1  Let all elements of A_0 be initialized with 0;
2  Cluster the Data X_t using FCM;
3  for i=1 to n do
4  |  for j=i to n do
5  |  |  C_t[i,j]=SimilarityFunction(i,j);
6  |  end
7  end
8  A_t = Merge(A_{t-1}, C_t);
9  Use similarity matrix A to create the final partition of the data;
10 Output the final partition;
```

of times N_{max} the same data matrix to the algorithm. The output of the algorithm depends on four criteria: I) the fuzzy partition generated by each fuzzy C-Means run in step 2; II) the similarity function used in step 5, which converts the fuzzy partitions into a similarity matrix; III) the way how the merging of similarity matrices is obtained in step 8; and IV) how the final partitioning of the resulting similarity matrix is done in step 9. Each of these criteria is discussed in the following subsections.

7.3.1 Fuzzy C-Means Algorithm

In practice, quite often there are situations where an object does not belong fully to a cluster but is more or less divided into several clusters, e.g. an object which is equidistant from two cluster centers and belongs thus to both clusters to some degree. The Fuzzy C-Means (FCM) algorithm [Bez76] has been developed to cope with this kind of situations. Algorithm 7.2 shows the steps that define the FCM algorithm. The output of the algorithm depends on three factors: the initial partitioning of the data, the number of clusters and the number of iterations. The initial partitioning is obtained randomly; the number of clusters given to FCM ($c^{(FCM)}$) is taken as the number of clusters given to the ensemble ($c^{(Ensemble)}$) multiplied by a constant. Depending on the size and complexity of the data sets, FCM needs different numbers of iterations to converge.

In comparison to the K-means algorithm which also could have been used to cluster the data, FCM is often more stable regarding local minima. This

Algorithm 7.2: Fuzzy C Means algorithm [Bez76]

Input: Input n Objects to Cluster x_i i=1..n, number of clusters $c^{(FCM)}$

Output: The Fuzzy Partition of the Data $U_{n \times c^{(FCM)}}$

Initialize $U_{n \times c^{(FCM)}}$ randomly such as $\forall i \sum_{j=1}^{c^{(FCM)}} U(i,j) = 1$;
repeat
  Calculate the vectors of cluster centers $c_j = \frac{\sum_{i=1}^{n} u_{ij}^m \times x_i}{\sum_{i=1}^{n} u_{ij}^m}$ for j=1..$c^{(FCM)}$ where m is the fuzzification factor, usually m=2;
  Calculate $d_{ij} = \|x_i - c_j\|$;
  for *i=1 to n* **do**
    for *j=i to* $c^{(FCM)}$ **do**
      if $d_{ij} > 0$ **then**
        $u_{i,j} = \frac{1}{\sum_{k=1}^{c} (\frac{d_{i,j}}{d_{ik}})^{\frac{2}{m-1}}}$;
      else
        $u_{i,j} = 1$;
      end
    end
  end
until $max\{\delta U\} < tolerance$ *or NumberOfIterations reached*;

is due to the fact that cluster memberships in FCM are fuzzy in contrast to crisp memberships in K-means. Since the search for the optimum tries to minimize an Euclidean sum of squares, which depends on the membership values, a point belonging to one cluster in K-means could belong to another cluster in the next iteration, bringing thus an abrupt change of the sum of squares. In FCM this process is smoother, leading to more stability.

7.3.2 Converting the Fuzzy Partition into a Similarity Matrix

The fuzzy clustering algorithm generates a so called fuzzy partition matrix containing information about the membership grades of objects in clusters. For illustration purposes let us consider the fuzzy partition matrix U representing the cluster membership grades of six objects $o_i, i = 1..6$ (rows) to four clusters $c_j, j = 1..4$ (columns).

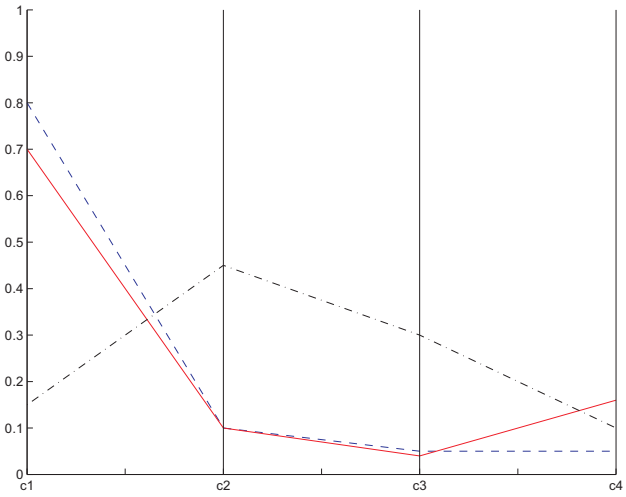


Figure 7.1: Membership grades visualized using parallel coordinates

$$C^{(d_1)} = \begin{pmatrix} .80 & .70 & .25 & .15 & .10 & .05 \\ .70 & .70 & .25 & .15 & .16 & .16 \\ .25 & .25 & .50 & .20 & .25 & .30 \\ .15 & .15 & .20 & .45 & .45 & .30 \\ .10 & .16 & .25 & .45 & .65 & .25 \\ .05 & .16 & .30 & .30 & .25 & .60 \end{pmatrix}$$

Figure 7.2: Fuzzy similarity matrix and its graphical representation

$$U = \begin{pmatrix} .80 & .10 & .05 & .05 \\ .70 & .10 & .04 & .16 \\ .25 & .05 & .20 & .50 \\ .15 & .45 & .30 & .10 \\ .01 & .65 & .09 & .25 \\ .05 & .05 & .60 & .30 \end{pmatrix}$$

Figure 7.1 shows the visual representation of the membership vectors for the objects o_1(dashed line), o_2 (solid line) and o_3 (dash-dotted line) using parallel coordinates. It is evident that o_1 and o_2 belong more to cluster c_1 and less to the rest, whereas o_4 belongs more to c_2 but belongs to a considerable degree also to c_3.

The fuzzy partition matrix can be considered as a projection of the data from the original feature space into the cluster space. Thus, the problem that is raised is to derive a new matrix describing the similarity between two objects in the cluster space (steps 4-8, Algorithm 7.1). Different similarity/dissimilarity functions can be used for this purpose [BM95]. We have

$$C^{(hard)} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

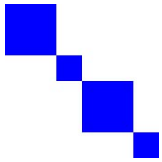


Figure 7.3: Hard similarity matrix and its graphical representation

implemented and tested several of these functions; three of them are listed below.

$$d_1(o_i, o_j) = 1 - max_{k=1}^{c} min\{U(i,k), U(j,k)\} \tag{7.1a}$$

$$d_2(o_i, o_j) = max_{k=1}^{c} |U(i,k) - U(j,k)| \tag{7.1b}$$

$$d_3(o_i, o_j) = \sqrt{\sum_{k=1}^{c} (U(i,k) - U(j,k))^2} \tag{7.1c}$$

d_3 and d_2 are the Euclidean and infinity norms, respectively, whereas the definition of d_1 comes from fuzzy theory and expresses how strong two sets intersect. A short evaluation of these dissimilarity measures is presented in section 7.4.

Thus, for the objects o_1 and o_2, based on the membership matrix U and using the above distance functions, we get $d_1(o_1, o_2) = .3$, $d_2(o_1, o_2) = .11$ and $d_3(o_1, o_2) = .14$. To convert them to similarities and to scale them in the range $[0, 1]$, each element is transformed according to the formula:

$$C^{(d_k)}(o_i, o_j) = \frac{max - d_k(o_i, o_j)}{d_k(o_i, o_j)} \text{ where k=1,2,3} \tag{7.2}$$

where max is the largest value the distance could reach. For d_1 and d_2 max is equal to 1 and for the distance d_3 max is equal to $\sqrt{2}$. The similarity matrix for the matrix U using the d_1 distance is shown in figure 7.2. The right part of the figure 7.2 presents the visualization of the matrix on the left using obtained obtained with techniques presented in Chapter 6. Darker colors represent larger values in the corresponding element of the similarity matrix.

The matrix $C^{(d_1)}$ contains information on how much two objects belong to the same cluster. It contains much more information than the analogous hard clustering similarity matrix presented in figure 7.3, which is obtained

from the fuzzy one by using the maximum criteria.

7.3.3 Merging of Similarities

After the similarities have been calculated, another problem is to find a way to merge these similarities for different clustering results (step 9, Algorithm 7.1). We have considered two alternatives: the MINIMUM (Formula 7.3b) and MEAN (Formula 7.3a) alternative.

$$A_t(i,j) = \frac{(t-1) \times A_{t-1}(i,j) + C_t(i,j)}{t} \qquad \text{for i,j=1..n} \qquad (7.3a)$$

$$A_t(i,j) = min(A_{t-1}(i,j), C_t(i,j)) \qquad \text{for i,j=1..n} \qquad (7.3b)$$

Matrix A_{t-1} stores information about the accumulation of cluster results for the first t-1 runs, whereas matrix C is the similarity matrix calculated for the t-th run, as described in 7.3.2. An evaluation of these two alternatives is described in section 7.4. Other alternatives would be to compute the MEDIAN of the accumulated values, but such an approach is much more expensive computationally than MINIMUM or MEAN.

7.3.4 Final Partitioning

The final step of the cluster ensemble algorithm is to partition the resulting accumulated similarity matrix into clusters. Several solutions were considered for this problem. The first solution is the use of a Minimum Spanning Tree inspired approach [Zah71] for partitioning the accumulated similarity matrix. However, the results are unbalanced clusters consisting very often of a single node only.

Consequently, two other possibilities were considered, namely to treat the problem of finding the final clusters of objects as a graph partitioning problem or to use *spectral clustering* for the same purpose.

7.3.4.1 Graph Partitioning

The graph to be partitioned is a complete graph with nodes representing the objects to be clustered and weighted edges represented by the respective values of the accumulated similarity matrix. Thus, the problem is to partition the complete undirected weighted graph $G = (V, E)$, where V represents the objects ($|V| = n$) and E is the same as the accumulated similarity matrix A_t. The METIS [KK98] approach was used to partition this graph into clusters due to its scalability and the good quality of the partitions it creates. METIS generates balanced clusters and consists of three basic steps:

1. *Coarsening phase* where the original graph is transformed into a sequence of graphs with decreasing size by collapsing vertices and edges

2. *Partitioning phase* where the smallest graph obtained in the above step is partitioned using heuristic algorithms

3. *Uncoarsening phase* which projects back the partition in the smallest graph into its predecessors. In this phase, vertices of the graph can be swapped only between neighboring partitions.

This approach is very fast but has the drawback that generates only balanced clusters. As such, it can be used in the cases when prior information confirming this fact exists. For other cases, the spectral clustering based partitioning, which is explained below can be used.

7.3.4.2 Partitioning by Spectral Clustering

Spectral clustering is concerned with the clustering of data based on the spectral decomposition of a preprocessed version of the distance matrix of the data. A subset of (normalized) eigenvectors generated from the spectral decomposition is assumed to represent the new features of the objects and which are then clustered using the well known K-means algorithm. The different spectral cluster algorithms differ themselves on the preprocessing of the distance matrix, on the subset of eigenvectors selected after decomposition as well as normalization (or lack thereof) of these eigenvectors.

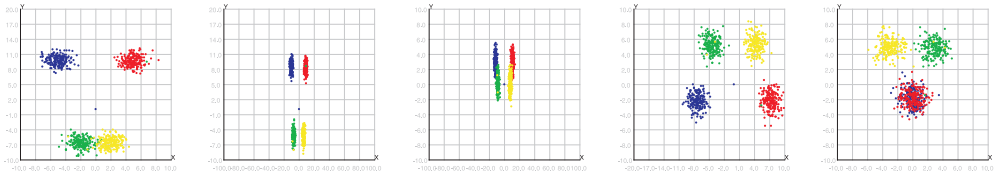
Considering that spectral clustering algorithms take as input distance matrices, they are natural candidates for performing the partition of the accumulated similarity matrix. Consequently, the spectral clustering algorithm by Ng et al. [NJW02], shown in Algorithm 7.3 is used to perform the spectral partitioning. The preprocessing performed includes Gaussian weighting of the distances and the calculation of the Laplacian of the weighted distance matrix (Steps 1-3). After the spectral decomposition, the largest k eigenvectors are normalized to unit length by dividing each coordinate with the vector length and are stacked in a $n \times k$ dimensional matrix, which is clustered using K-means. The partition created in this way is the result of spectral clustering. In step 6 of Algorithm 7.3, K-means can be replaced with fuzzy C-means if a fuzzy partition instead of a hard one is needed.

Algorithm 7.3: Spectral Clustering [NJW02]

Input: Proximity matrix Δ and the number of clusters k

Output: Partition of objects into clusters

1 Preprocess the proximity matrix, calculating $a_{ij} = exp(\delta_{ij})/2\sigma^2$;
2 Calculate a diagonal matrix D as the sum of rows of A;
3 Calculate the matrix $L = D^{-1/2}AD^{-1/2}$;
4 Estimate the eigenvectors and eigenvalues of L using the spectral decomposition theorem, $L = U\Lambda U'$;
5 Form the matrix Y by stacking the eigenvectors $u_1, ..., u_k$ after norming them to unit length;
6 Cluster Y using K-means and return the achieved assignment vector as the spectral clustering result;



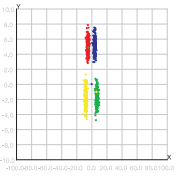
(a) Spin parameter t = 0 (t_{max} = 9)

(b) Spin parameter t = 1 (t_{max} = 9)

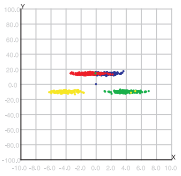
(c) Spin parameter t = 2 (t_{max} = 9)

(d) Spin parameter t = 3 (t_{max} = 9)

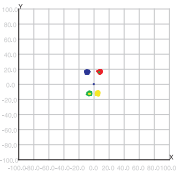
(e) Spin parameter t = 4 (t_{max} = 9)



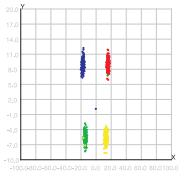
(f) Spin parameter t = 5 (t_{max} = 9)



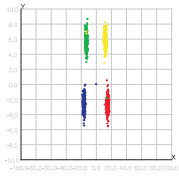
(g) Spin parameter t = 6 ($t_{max} = 9$)



(h) Spin parameter t = 7 ($t_{max} = 9$)



(i) Spin parameter t = 8 (t_{max} = 9)



(j) Spin parameter t = 9 (t_{max} = 9)

Figure 7.4: Four Gaussian clusters and their transformation using a spiral function

7.4 Experimental Results

7.4.1 Indices for Evaluating Clustering Results

Evaluating the results of a clustering algorithm is not a trivial task. For this purpose, both internal and external criteria that show the goodness of the results can be defined [JD88]. Whereas internal criteria try to assess the cluster quality using only the data themselves, external criteria use a priori information (such as category labels assigned beforehand) for assessing the cluster quality. For the synthetic data sets external indices as listed below will be used.

7.4.1.1 External Goodness Measures

F-measure (F) similar to [YK03]. Let $n_{i,j}$ be the number of objects belonging to cluster i and class j, n_i be the number of objects in cluster i and m_j be the number of objects in class j. Let $precision(i,j) = \frac{n_{i,j}}{m_j}$ and $recall(i,j) = \frac{n_{i,j}}{n_i}$. Then

$$F(i,j) = \frac{2 \times precision(i,j) \times recall(i,j)}{precision(i,j) \times recall(i,j)} \tag{7.4a}$$

$$F = \sum_{i=1}^{k} \frac{n_i}{k} max\{F(i,j)\} \tag{7.4b}$$

where k is the number of clusters. F varies between 0 to 1 and larger F values indicate larger similarity between partitions.

Normalized Mutual Information (NMI) [SG02]. Let the clustering λ^a define the frequencies $\frac{n_i}{n}$, where n_i is the number of elements in cluster i and n is the total number of elements. Then, the entropy of this partition is defined as $H(\lambda^a) = -\sum_i n_i log(n_i)$. For two partitions λ^a and λ^b the mutual information is defined as in equation (7.5a), whereas the normalized mutual information is defined as shown in equation (7.5b).

$$I(\lambda^a, \lambda^b) = \sum_{i}^{k_a} \sum_{j}^{k_b} n_{ij}^{ab} log(\frac{n \times n_{ij}^{ab}}{n_i^a \times n_j^b}) \tag{7.5a}$$

$$NMI(\lambda^a, \lambda^b) = \frac{-2 \times I(\lambda^a, \lambda^b))}{H(\lambda^a) + H(\lambda^b)} \tag{7.5b}$$

Jaccard similarity measure (J) [JD88]. Let us calculate the matrix $C_{n \times n}$ where $C_{ij} = 1$ iff the objects i and j are in the same cluster, 0 otherwise

and the matrix $G_{n\times n}$ where $G_{ij} = 1$ iff the objects i and j are in the same prior class and 0 otherwise. Let a_{00} denote the number of items which are 0 in both matrices, a_{11} the number of items which are 1 in both matrices and define similarly a_{10} and a_{01}. Then

$$J(\lambda^a, \lambda^b) = \frac{a_{11}}{a_{01} + a_{10} + a_{11}} \tag{7.6}$$

Minkowski Score (MS) similar to [HY04]. Let $C_{n\times n}$ represent the matrix where $C_{ij} = 1$ iff the objects i and j are in the same cluster according to the clustering method, and let G represent the analogous definition for the true clusters i.e. using prior knowledge. Then

$$MS(G, C) = \frac{\sqrt{\sum_i \sum_j (G_{i,j} - C_{i,j})}}{\sqrt{\sum_i \sum_j G_{i,j}}} \tag{7.7}$$

In contrast to the other measures described in this section, a smaller Minkowski score indicates a larger similarity and is the only external measure of this nature used in this chapter.

7.4.1.2 Internal Goodness Measures

Internal measures are suitable for clustering since they are data driven and do not require external information such as ground truth data, i.e. the distribution of the real clusters, to be calculated. Conversely, internal measures report the quality of clustering results with respect to the data clustered, thus being a biased measure for the goodness of obtained clusters. Two internal measures are used to evaluate the goodness of clustering results for sensitivity matrices. The first one focuses on the compactness of the clusters relative to their centers and is defined using the formula

$$SSQ(X, C) = \sum_{j\in C} \sum_{i\in C_j} \|x_i - c_j\| \tag{7.8}$$

Here X represents the data and C the partition implied by the clustering. For each cluster, its respective center, the vector c_j, which has the same dimensions as the input vectors x_i is calculated. The sum of all these distances indicates how compact the clusters are; the smaller this distance, the better the clustering. The second measure, silhouette [Rou87] of objects in clusters, is more complicated and in contrast to the above measure is calculated

separately for each object in the data set. It is calculated for each object as

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{7.9}$$

where $a(i)$ is the average similarity of the object i to the objects found in the same cluster as i and $b(i)$ is the minimal average distance of object i to the other clusters. The silhouette has the property $-1 \leq s(i) \leq 1$. Values near 1 indicate that object i belongs to the right cluster, values near 0 indicate that object i belongs to the same degree to the actual cluster as well as the nearest cluster and values near -1 indicate that the object i is probably misclassified.

7.4.2 Synthetic Data Generation

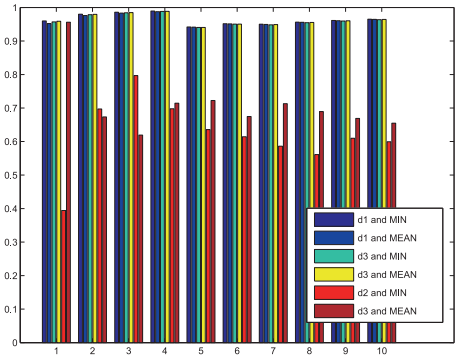
We generated several synthetic data sets for testing our approach. One of these data sets, called Butterfly 1 is visualized in Figure 7.4. The data set is generated as follows. First, four points (-5,5),(5,5),(-2,-2) and (2,-2) serving as cluster centers are selected. Then 250 points are generated from the two dimensional Gaussian distribution for every cluster center, thus generating four Gaussian clusters. Two of these clusters intersect slightly with each other whereas the other two are separated from the rest. The time-varying effect for this data set is accomplished using a spiral transformation, presented by the equations 7.10, 7.11 and 7.12. These transformations need two parameters, *maxtime* which indicates the discretization level and *time* which indicates the requested time point.

$$tt = 3 \times \frac{\pi}{2} \times (1 + 2 \times \frac{time}{maxtime}) \tag{7.10}$$

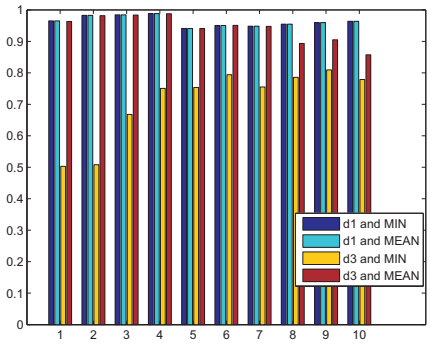
$$newX = oldX + tt \times cos(tt) \tag{7.11}$$

$$newY = oldY + tt \times sin(tt) \tag{7.12}$$

By selecting different directions for different clusters i.e. multiplying their coordinates appropriately with -1, we have created the effect that clusters are mixed temporarily with each other, as shown in Figure 7.4(c) or 7.4(e). Later on, the clusters are again separated from each other.



(a) Evaluation of distance/merge combinations for fixed $c^{(FCM)}$=2 × $c^{(Ensemble)}$



(b) Evaluation of distance/merge combinations for fixed $c^{(FCM)}$=20

Figure 7.5: Evaluation of different distance/merge combinations

7.4.3 Parameter Evaluation

In subsections 7.3.2 and 7.3.3, we have presented several alternatives for calculating distances and merging them together. Since the clustering of data is an explorative process, these alternatives, together with other parameters of the clustering (such as number of clusters of the ensemble $C^{(Ensemble)}$, number of clusters of the individual FCMs $c^{(FCM)}$, maximum number of iterations for individual FCMs) can be considered as the user input to the ensemble FCM.

To evaluate which combination of distance and merge operation achieves the best results, we conducted a set of experiments with data set Butterfly 1. The results of these experiments are shown in Figure 7.5. The experiments were carried out using $maxtime = 10$ using different combinations of distance/merge operation. The bar chart shown in Figure 7.5 shows the F-measure for the specific combination. Figure 7.5(a), which is obtained by applying Ensemble-FCM with fixed $c^{(FCM)} = 8$ and where FCM is executed until it converges, shows that combinations using distance d_2 perform badly, whereas the rest has comparable results. In Figure 7.5(b) we increased $c^{(FCM)}$ to 20 to see how it affects the results of clustering. It turned out that combinations with d_3 were more affected in this case. From the experiments, the combinations d_1-MIN and d_2-MEAN deliver good results; d_2-MEAN is more robust to abrupt changes in data and as such this combination will be used for the evaluation in the next subsection.

As already mentioned in subsection 7.3.1, the number of iterations needed for the convergence of FCM depends on the size and complexity of the data set. In this context, two options were tested regarding the convergence of

Table 7.1: Cluster results for synthetic data set Butterfly 1

	Worst Case			Average Case			Best Case		
	Km.	FCM	**Ens.**	Km.	FCM	**Ens.**	Km.	FCM	**Ens.**
MS	0.998	0.893	**0.711**	0.504	0.244	**0.170**	0.000	0.000	**0.000**
F	0.542	0.666	**0.752**	0.853	0.945	**0.971**	1.000	1.000	**1.000**
J	0.491	0.553	**0.597**	0.744	0.887	**0.928**	1.000	1.000	**1.000**
NMI	0.528	0.672	**0.735**	0.858	0.931	**0.947**	1.000	1.000	**1.000**

Table 7.2: Cluster results for synthetic data set Butterfly 2

	Worst Case			Average Case			Best Case		
	Km.	FCM	**Ens.**	Km.	FCM	**Ens.**	Km.	FCM	**Ens.**
MS	1.115	0.953	**0.798**	0.671	0.449	**0.419**	0.000	0.000	**0.000**
F	0.552	0.623	**0.719**	0.804	0.901	**0.918**	1.000	1.000	**1.000**
J	0.364	0.440	**0.517**	0.642	0.783	**0.800**	1.000	1.000	**1.000**
NMI	0.507	0.577	**0.653**	0.765	0.847	**0.851**	1.000	1.000	**1.000**

FCM: 1) FCM is executed until it converges and 2) the number of iterations is fixed (usually less than the number of iterations needed to converge) resulting in a semi-weak partitioning of the data [TJP03]. During our tests, the second option generated comparable results with the first one in less computational time and consequently it will be used for the following evaluations.

7.4.4 Evaluation of the fuzzy cluster ensemble

The Ensemble-FCM was evaluated on several time-varying data sets. The results of applying the Ensemble-FCM on the Butterfly 1 data set for $maxtime = 100$ are shown in Table 7.1. The table represents all the introduced measures of quality for K-Means, simple FCM and Ensemble-FCM. The two first methods consider each point of time as a separate data set. Ensemble-FCM, on the other hand, accumulates the timely evolution in a single clustering. For all three methods, the worst, average and best case are calculated for every measure. The ground truth data comes with the data generation process because cluster centers were selected beforehand. From the table we see that all three methods generate the perfect clusters in the best case; this can be explained by the fact that the data originally represents four Gaussian clusters.

Ensemble-FCM, besides having the best results overall, has also a smaller variance compared to the two other methods.

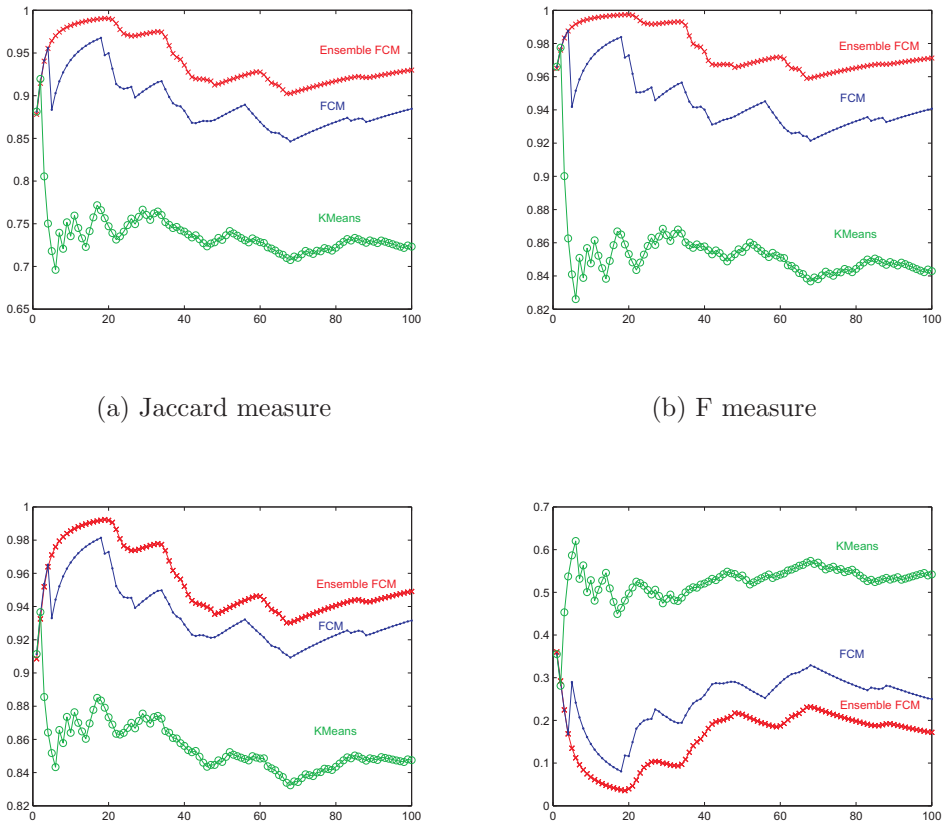
Furthermore, a more complicated data set named Butterfly 2 is generated using (-1.5,-2) and (1.5,-2) instead of (-2,-2) and (2,-2) as cluster centers and by selecting the points from two dimensional Gaussian distribution $N_2(center_i, \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix})$ for each $center_i$), obtaining thus clusters which are even more intersected with each other. The parameter *maxtime* was set to 100. The results of Ensemble-FCM are presented in Table 7.2. The overall results are worse than with the data set Butterfly 1 because of the larger intersection effect. However, Ensemble-FCM has again the overall best results and the most stable ones.

Both Table 7.1 and 7.2 show that the proposed fuzzy clustering ensemble approach outperforms the single K-means and FCM algorithms. For example, in the case of the Butterfly 1 data set, the overall average F-measure of the ensemble is 0.971, whereas the K-means and FCM have achieved an overall average F-measure of 0.853 and 0.945, respectively (see Table 7.1, row 2, column named "Average Case"). The same conclusion can be drawn for the other measures, indicating the robustness of our proposal in the average case. Figure 7.6 shows how the average accuracy measures evolve over time when $maxtime = 100$ for the Butterfly 2 data set. From the figure, it is clear that K-Means has the worst results, and in addition its variance is higher than the variance of the other two methods. Ensemble-FCM has a similar course as FCM, which is understandable because it is derived to a certain degree from FCM. However, the results of Ensemble-FCM do not deteriorate instantly as it is the case with FCM, making Ensemble-FCM stable and robust for analyzing time-varying data.

7.4.5 Cluster Analysis of Time-Varying Sensitivity Matrices

Number of clusters. Performing cluster analysis via partitional methods requires a pre-specified parameter, namely the number of clusters. Estimating the "right" number of clusters is a hard problem and many approaches exist for dealing with this problem. Usually, a goodness measure is fixed (as for example SSQ above) and its plot for a different number of clusters is examined. The point of maximum curvature in this plot, which is called the *knee* of the curve, is chosen as the number of clusters, based on the assumption that a larger number of clusters would bring minimal benefits.

This idea has been formalized in the gap statistic [TWH00]. Such approaches as the gap statistic however are computationally expensive, and are usually applied on sampled subsets of the original data sets. In the case of



(a) Jaccard measure

(b) F measure

(c) NMI measure

(d) Minkowsi measure

Figure 7.6: The evolution of measure coefficients for Ensemble-FCM, FCM and K-Means for 100 consecutive time points with data set Butterfly 1

sensitivity matrices of the E. coli model presented in Section 6.8, the SSQ error is calculated for numbers of clusters in the range [2..10]. The results are presented in Figure 7.7. The different plots reflect SSQ for a subset of 20 time points from 143 time points. All these curves have the maximum curvature for $k = 3$, and for this reason the number of clusters is chosen as 3. This can be explained by considering for example Figure 6.20(d); three large groups are distinguished at the left, at the right and in the middle of the reorderable matrix.

Performance of FCE with sensitivity matrices. The time-varying sensitivity matrices generated during the simulation of the E. coli model presented in Section 6.8 have dimensions of 116 parameters $\times$18 metabolites for

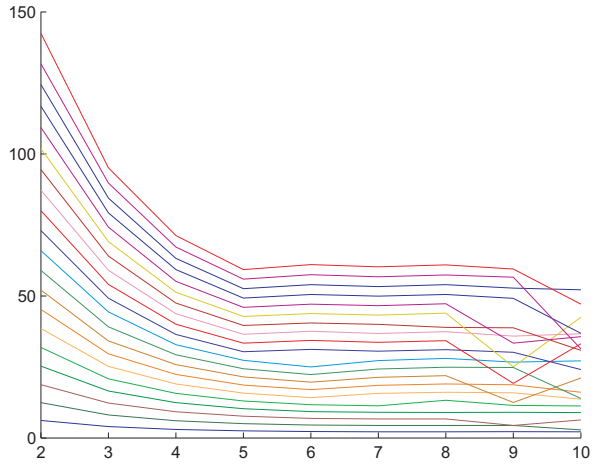


Figure 7.7: SSQ measure for different number of clusters, for a sample of time points

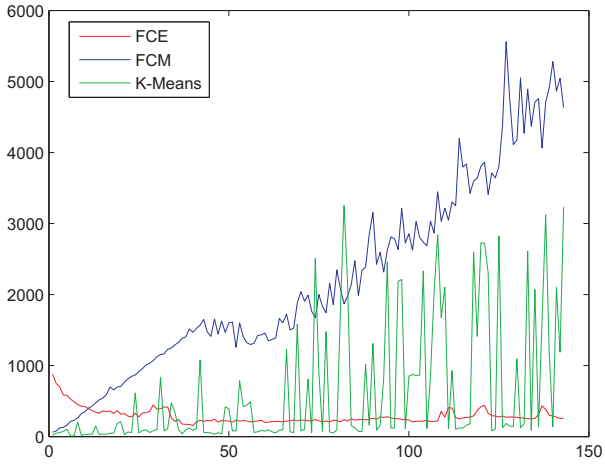


Figure 7.8: Comparison of execution times for FCE, FCM and K-Means

each time point. In total there are 143 time points. FCE considers each point of time individually, and exhibits a nearly constant execution time, independently of the number of time points. For FCE and K-Means the overhead increases with every time point added; in the last step of the algorithm the data would have dimensions of 116×2574. The execution times in milliseconds, are shown in Figure 7.8, and they reflect the above mentioned fact.

Figure 7.9 presents the FCE silhouette profiles for the 116 parameters, as they evolve over time. High profiles of silhouettes indicate accurate clustering

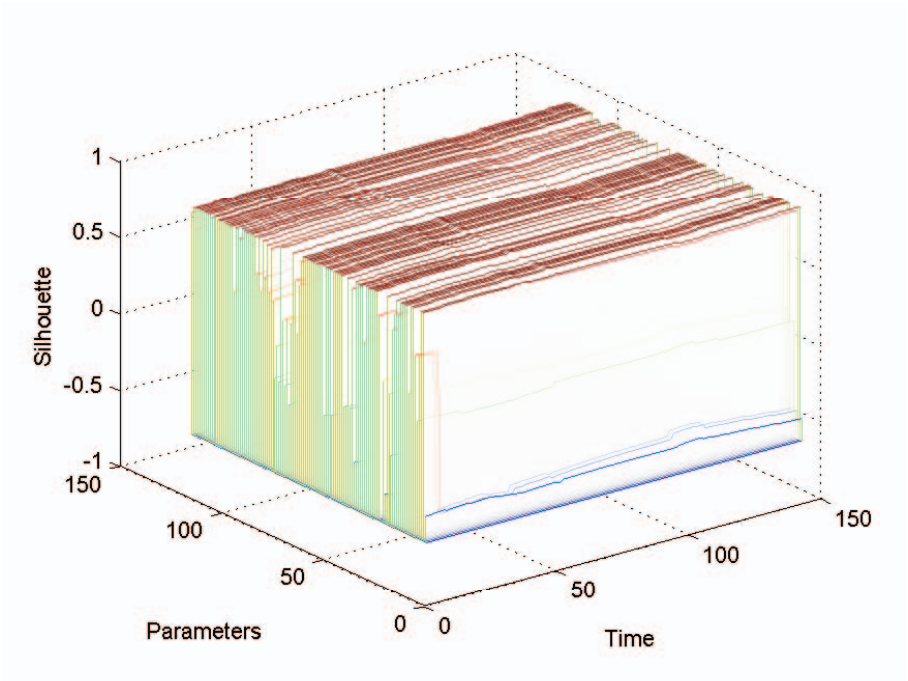


Figure 7.9: Evolution of FCE silhouettes for all parameters and all time points

of parameters, whereas low ones indicate bad classification of parameters. Most of the parameters of this model are clustered correctly, since they are described by high silhouette values.

7.5 Summary

In this chapter, we have presented a novel fuzzy cluster ensemble for analyzing time-varying multidimensional data. The approach is based on the idea of combining the fuzzy partitions created during the simple fuzzy clustering process. Several issues associated with such a fuzzy cluster ensemble approach were addressed. Experimental results for different synthetic and real data sets were carried out showing competitive results. The experiments have shown that the proposed approach has a higher degree of robustness compared to FCM or K-Means due to its ability to memorize the past relationships such that it will not be strongly affected in the negative direction when abrupt bad results are obtained by FCM.

8
Large-Scale Visualization

8.1 Introduction

Traditional visualization techniques, as presented in Chapter 6 function well as long as the underlying data sets have acceptable dimensions and size. For example, the reorderable matrix works well with data dimensions of up to 400×400.

Visualization techniques which deal with larger data sets need workarounds, which can solve the problems raised by the limited screen space. However, such solutions depend quite often on the specific visualization method. In this context, we will focus on the following ideas:

- Filtering approaches for visualization. The simplest approach to reduce the data dimensions is to enable the user to interactively filter the wanted information. This filtering can be interactive i.e. by interacting with the visualization, or query based i.e. by enabling the user to query the visualization using a specific query language. Both methods have their advantages and drawbacks depending on the context.

- Implementing Overview+Detail features for visualization. In this way, a raw overview of the data is always in the focus of the user and furthermore if details are wanted, they are shown in a separate visualization window.

- Distortion of the views in order to enable Focus+Context visualizations, where details and overview are merged into one window, which displays both global and local details.

- Tiled displays, where the responsibility for visualizing a large data set is distributed between a set of monitors (or computers).

This chapter is concerned with solutions to these problems in the context of tabular visualization techniques, such as the reorderable matrix. The chapter is organized as follows. Section 8.2 gives a survey of related work in the field. Section 8.3 describes a filtering approach for the MetVis toolkit, which enables command-line based filtering of the data. Section 8.4 focuses on overview+details techniques for visualizing large data sets. Section 8.5 is concerned with distortion techniques for the reorderable matrix. Section 8.6 discusses tiled displays for large-scale visualization in the context of the reorderable matrix. Section 8.7 concludes the chapter.

8.2 Related Work

The related work regarding filtering, Overview+Details and Focus+Context techniques was discussed in the Chapter 3, especially in Section 3.6.1, when the technical background of information visualization was given. Consequently, we will focus here on related work in visualization using tiled displays, and in the following sections will mention approaches related to the ideas presented in this chapter.

Tiled displays are used extensively in large scale visualization systems. The basic idea is to integrate a set of displays, organized in a grid, such as their entirety forms a virtual display with a larger size. This approach has both additional hardware requirements (e.g. additional graphics cards), and software requirements for accomplishing the integration and alignment between displays. The displays in this context can be monitors or projection-based devices.

Tiled display systems are widely used in large scale visualization, e.g. the PowerWall [oM06], scalable display wall [Uni06], and others [SFF+00, Lab06b, Lab06a].

For further information on the construction of such systems, Hereld et al. [HJS00] describe concepts and technologies on how to build tiled displays.

In the context of information visualization, Wei et al. [WSK+00] discuss experiences in visualizing massive telecommunication data sets with large scale displays. A 4×2 projector system, called InfoWall, with more than 10 million pixels was built. Data collected from AT & T networks and services is visualized in real time on the InfoWall platform.

Tiled displays and the software related to their functioning, provide a virtual layer which is independent of the application field. Thus, the techniques represented above are generic in nature and could be used for any visualization technique. On the other side, they are expensive to construct and

maintain.

In our case, the generality of such approaches is traded off for the possibility to construct a light weight system using Java RMI, which allows for tiled displays in the context of the colored reorderable matrix presented in Chapter 6. The details are given in Section 8.6.

8.3 Filtering Approaches for MatVis

Filtering enables the user to reduce the quantity of information in a display based on the user's interests. Filtering can be achieved by selecting appropriate subsets of the data (browsing the data), or by specifying attributes that the subset must fulfill (querying the data). Both Overview+Details and Focus+Context techniques presented below provide browsing capabilities to their respective visualization techniques. However, for very large data sets, browsing might be difficult or impossible to achieve.

8.3.1 Interactive Filtering

Interactive filtering follows a middle way between browsing and querying and is extensively used in the process of data exploration. Strongly related with filtering is the concept of *brushing*, which implies selection and highlighting of objects in visualization according to the interests of user. Figure 8.1 and Figure 8.2(a) illustrate the idea of brushing as implemented in MatVis. Parameters can be selected either in the dimension reductioning view (as illustrated in Figure 8.1) or they are automatically selected in the other respective views as, for example, in the reorderable matrix as in Figure 8.2(a). These parameters are highlighted using a different color of the respective cell boundaries. Furthermore, the selected parameters can be excluded from further consideration on the user's request. Figure 8.2(b) shows the same matrix as in Figure 8.2(a) up to the highlighted parameters which are excluded from the visualization.

8.3.2 Query-Based Filtering

Interactive selection and filtering enriches the visualization for better data exploration. However, there are situations when command line based approaches are more effective for filtering purposes. This is especially the case when objects and their features have meaningful names, which can be used to address them in a query expression.

For this purpose, a robust language allowing "AND/OR" logic and parenthesis constructions is defined for achieving filtering of the data set, in our

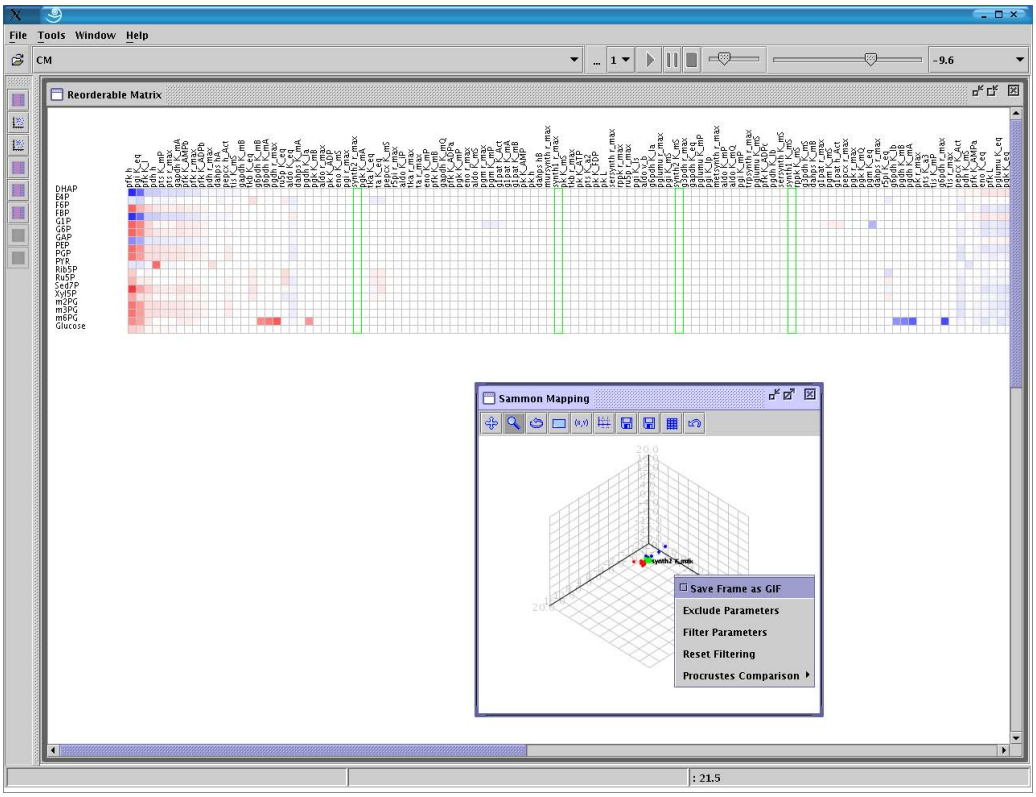
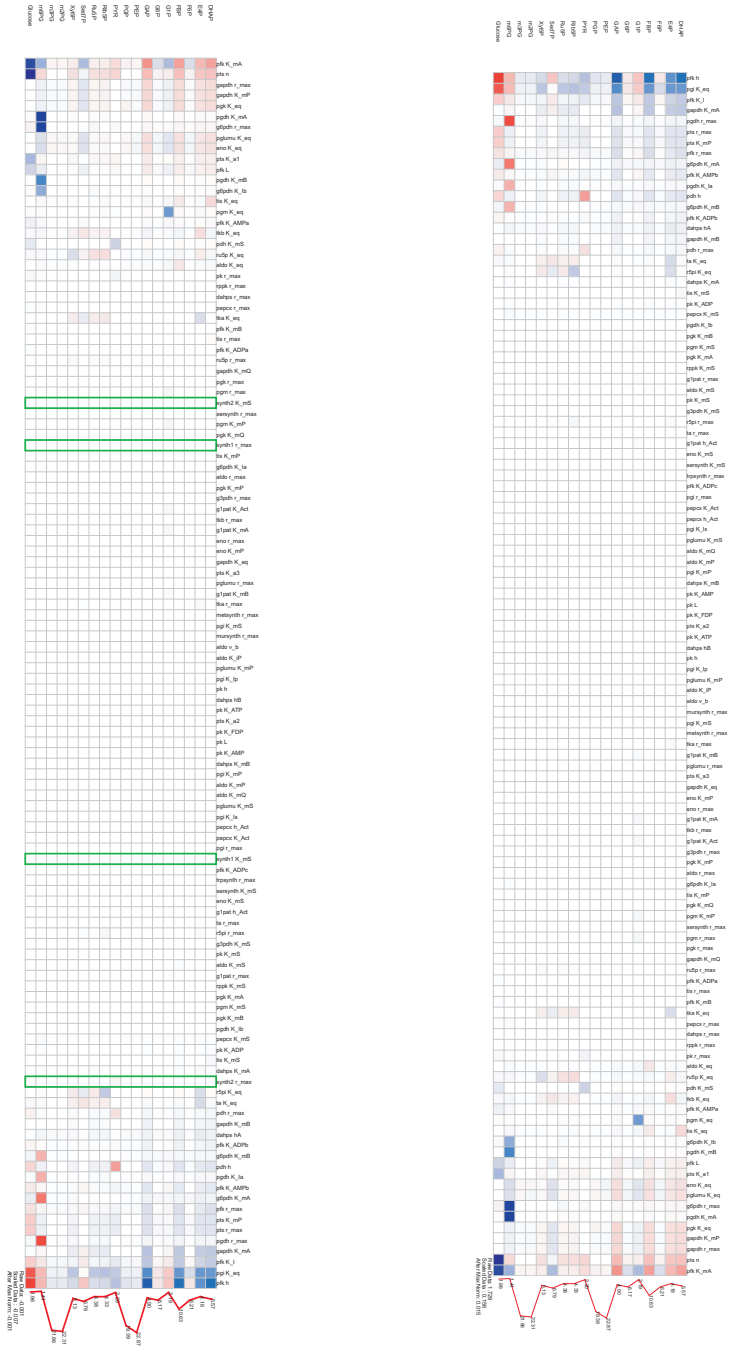


Figure 8.1: Linking and brushing between views in MatVis. The selected objects in the dimension reductioning view are highlighted in the colored reorderable matrix.

case sensitivity matrices, in MatVis. Considering that MatVis is built in Java, JavaCC [jav06] comes in handy as a tool which allows to express the definition of the language in the BNF (Backus-Naur Form), generating thus a Java program which parses input in the defined language. The parsing process itself bears a certain resemblance to the event based parsing of XML files, which was described in Chapter 4. JavaCC, which stands for *Java Compiler Compiler*, accepts as input the definition of a language in BNF and emits Java code for speeding the development of parser logic for the specified language. It was developed in analogy with YACC (Yet Another Compiler Compiler), the C-based tool developed by AT&T for the purposes of building parsers for C. The whole definition of the query language proceeds inside a template file with the extension *.jj*, which when compiled with JavaCC generates the real parser classes of the defined language in Java. The whole process is similar to the querying approach provided in [Bre04]. The template needs to specify the following components: 1) the context of parsing, 2) definition of the query language tokens and whitespace, 3) definition of the syntax of the language and 4) definition of behavior of parser.



(a) Brushing parameters in the Reorderable Matrix

(b) Visualization without the filtered parameters

Figure 8.2: Brushing of the reorderable matrix and its visualization after filtering of parameters

Query Specification The two dimensions of the reorderable matrix can be considered as two imaginary tables for the purpose of filtering. Furthermore, each specific visualization technique can be considered as a separate database; this can help, for example, in case we want to differentiate filtering in the different visualization techniques i.e. apply different filters in different views.

Consequently, the user may enter filtering expressions of the form *(metabolite = "Pyr" OR metabolite ="Gluc") AND parameter = "V_max*"* for considering the sensitivity of *Pyr* and *Gluc* for all parameters starting with *V_max*. In the above expression, keywords *AND* and *OR* as well as parentheses were used. In the language of JavaCC, they would be specified in the block:

```
SKIP:
{
  " "
  "\n"
  "\r"
  "\t "
}
TOKEN:
{
  <AND: "and">
  | <OR: "or">
  | <PARAMETER: "parameter">
  | <METABOLITE: "metabolite">
  | <LPAREN: "(">
  | <RPAREN: ")">
  | <EQUALS: "=">
  | <NOTEQUAL: "<>">
}
TOKEN:
{
  <STRING : (["A"-"Z","a"-"z","0"-"9"])+ >
| <QSTRING: "\"" (~["\"","*","?"])* (["*" , "?"])? "\"">
}
```

First, the white space characters, i.e. characters that are ignored, are defined. Then, the specification of the token elements of the language follows. This includes both tokens which define the language (such as AND, OR, EQUALS, etc.) as well as literals (STRING and QSTRING). The definition of literals proceeds using *regular expressions*. On the other hand, regular expressions are allowed as literals for the query language itself.

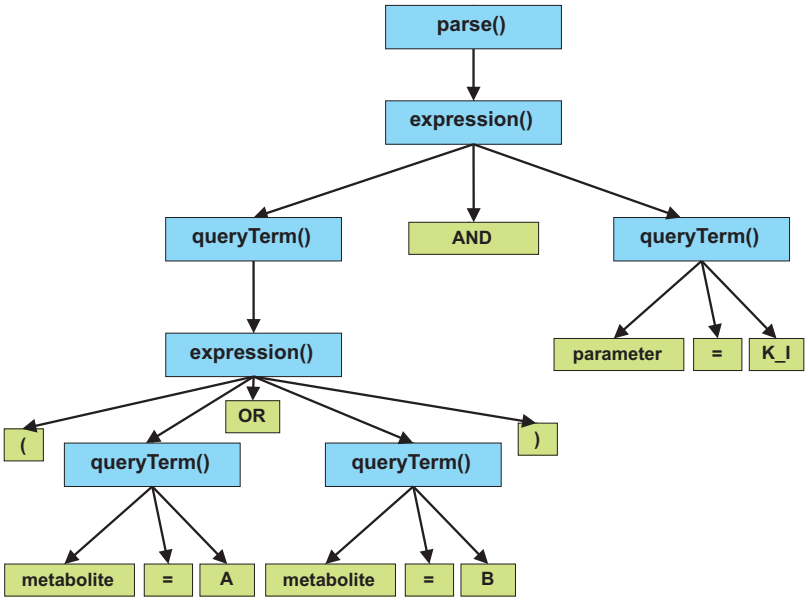


Figure 8.3: An example query and its parsing tree

Query Processing The template for processing queries is defined based on the tokens defined above and on parsing rules which decompose the query into its components. During the query decomposition, three main methods, namely:

parse()

expression()

queryTerm()

accomplish the processing of the query parts.

Figure 8.3 presents the parsing tree generated for the filtering expression *(metabolite = A OR metabolite =B) AND parameter = K_I*. Thus, *parse*() method is produced out of *expression*() methods, which by themselves are composed of *queryTerm*() methods combined with AND or OR. The *queryTerm*() methods are produced out of an elementary piece of the query (e.g. *parameter = A*) and an *expression*(). The same filtering achieved interactively in Figure 8.2(b) can be achieved also using the query expression *parameter = "synth2*" OR parameter ="synth1*"*.

8.4 Overview+Detail

Overview+detail interfaces convey the information to the user by means of multiple views, where some views are specialized to show detailed information about the object of visualization (e.g. through a larger zooming factor), whereas other views concentrate on overviews of the information space

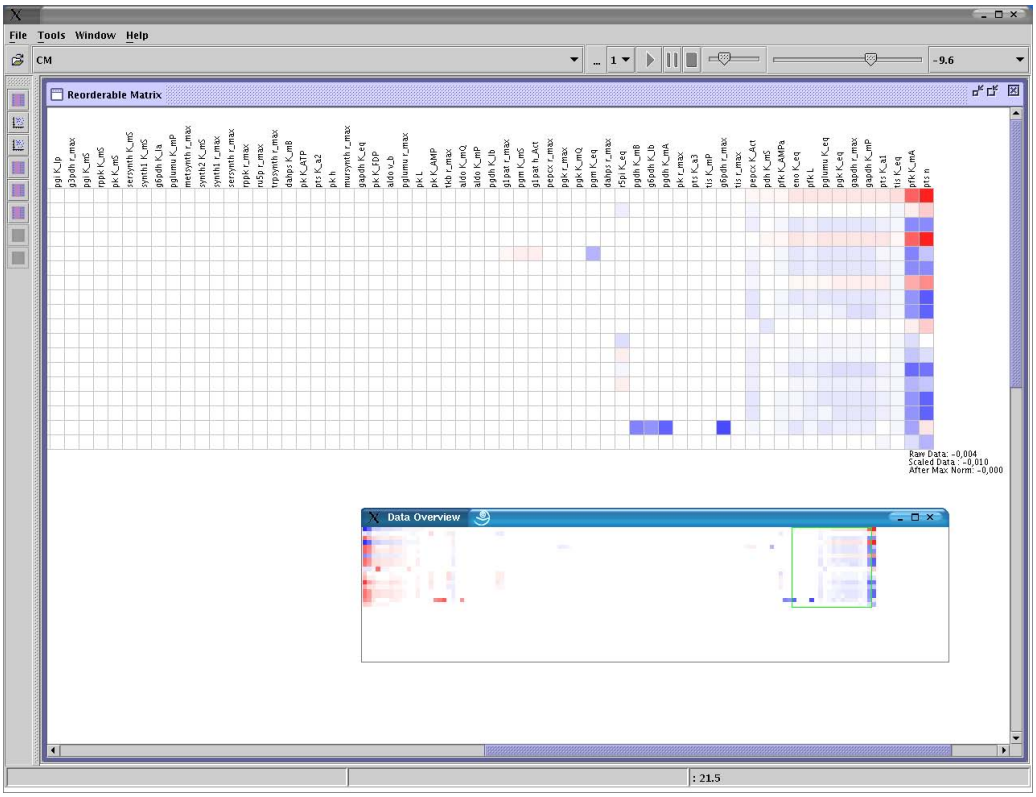


Figure 8.4: Overview+Details approach for MatVis. The view on the foreground enables the user to navigate in large scale visualizations.

(e.g. using a smaller zooming factor). The overview+detail paradigm is used extensively in rendering of maps, image collections, text collections, etc. Using overview+detail interfaces is associated with several benefits, namely efficient navigation, because users may navigate using the overview window rather than using the detail window and keep track of their current position in the information space. Furthermore, the overview window itself might give users task-relevant information on a higher level. The overhead caused by the visual search in the user's memory for keeping the overview and details views synchronized with each other on the other side represents a drawback of overview+detail interfaces.

To properly use overview+detail interfaces, taxonomies and guidelines have been developed to make their usage easier [HBP02]. In this context, the overview and detail views need to be tightly coupled to each other and the ratio of level of details i.e. zooming factors in both views should not exceed a certain bound (e.g. 25-30). In addition, depending on the tasks, proper sizes of the overview and detail windows need to be selected. Some tasks require a large detail window coupled with a small overview window whereas some other tasks require a large overview window with a small details

window.

The overview+detail paradigm is used in the context of the MatVis platform to enhance the reorderable matrix visualization method by extending its capacity of visualizing larger data sets. Consequently, for large dimensions, overview windows which represent zoomed out versions of the full space are estimated and shown to the user. The detail window shows a part of the matrix visualized using normal zooming levels, allowing for proper analysis of details. Figure 8.4 shows an example of the overview+detail paradigm in action. The overview window is shown on the foreground and the detail window in the background. The detail window is focused on the part of the matrix which corresponds to the selected region in the overview window. The ratio of the zooming levels in the two windows is four in this case, but higher zooming ratios are applied for larger data sets.

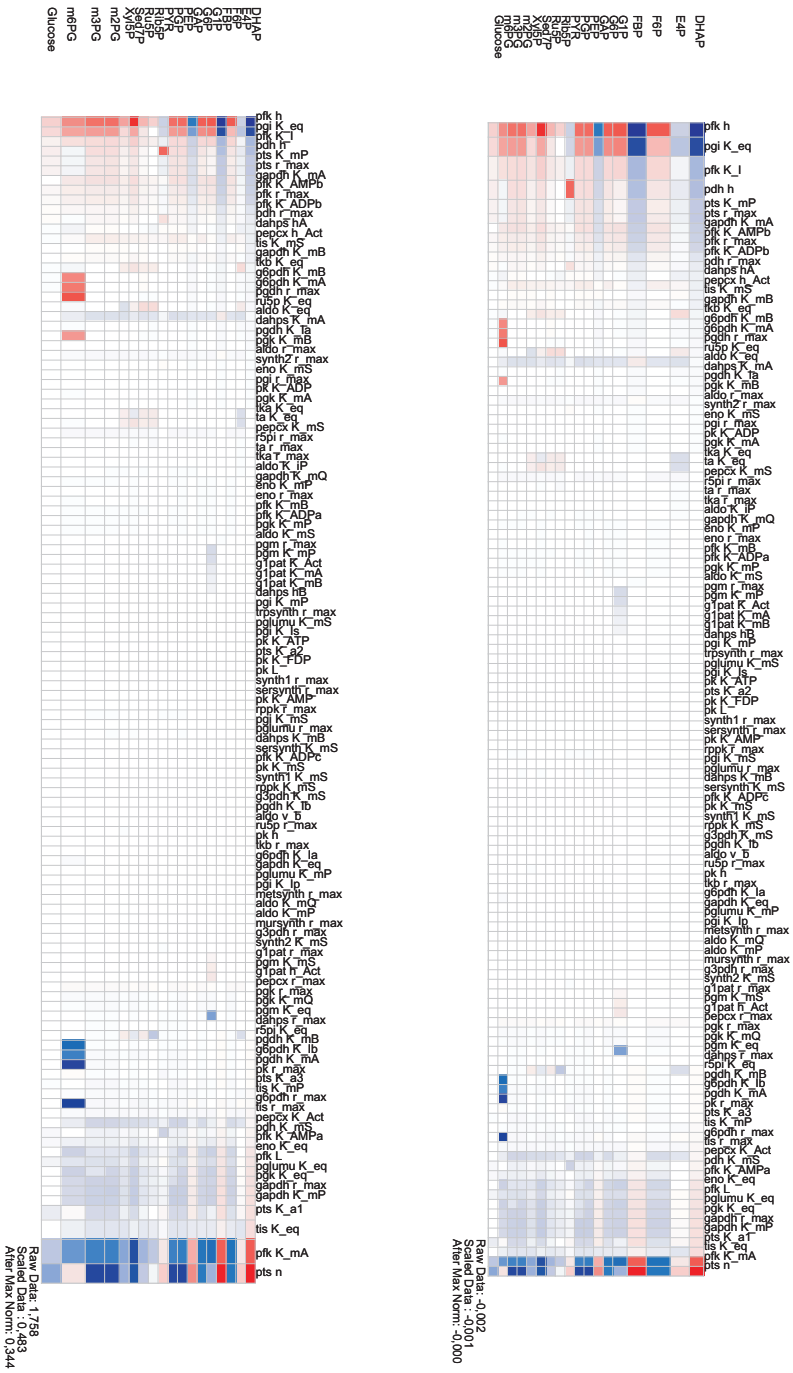
8.5 Focus+Context Techniques

Focus+context techniques were reviewed shortly in Section 3.6.1.1. Distortion techniques represent a large group of focus+context techniques. They allow an exploration process which provides means for focusing both on details and on the overview of the data in the same interface, in contrast to overview+detail techniques where the two are separate from each other. Distortion techniques are based on the simple idea of using a variable level of details in the same visualization window, which means usually a high level of detail for selected portion(s) of data and a lower level of detail for the rest.

It is obvious that distortion techniques are effective when used in an interactive context. The application of distortion techniques depends on the context of visualization. Thus, for visualizing trees and graphs, a transformation in another space, e.g. hyperbolical space, can be used [LRP95]. In other contexts, as for example in the visualization of maps, a lens-like distortion procedure can be used. Furthermore, the different distortion techniques distinguish themselves from each other on the number of foci and on how the level of details is decremented from the focus to the normal visualization area.

Returning to the context of MatVis and the reorderable matrix visualization, distortion can help in extending the capabilities of tabular visualizations. Subsequently, a robust distortion procedure is implemented which provides:

- Simultaneous row and columnwise distortion. The focus in this case is interactively set on a specific cell of the reorderable matrix. This cell and its respective row and column define the center of the distortion. This cell has the highest level of detail i.e. the highest zooming level. The other cells have the same width or height with the center of the



(a) Focus is on the lower right part of the window

(b) Focus is on the upper left part of the window

Figure 8.5: Illustration of distortion in the reorderable matrix

distortion if they are in the same column respectively row. For the rest, the zooming level is either minimal if they are far from the distortion center or reduced proportionally with the distance to the focused cell.

- One-dimensional distortion of either the rows or columns. If the user wants to consider whole objects (e.g. parameters) with their features (e.g. metabolites), then the one-dimensional distortion makes more sense. Here the focus of the user is not on a single cell any more but on a column or in a row. The distortion takes place only in one dimension, and the zooming level on the other dimension stays constant.

Figure 8.5 shows two reorderable matrices with different distortion centers. In Figure 8.5(a) the focus is on the lower right part of the matrix whereas in Figure 8.5(b) the focus is on the upper left part of the matrix. Consequently, the user can keep a certain part of the matrix in his/her focus while always having a global view of the whole data set available in the same interface.

8.6 Tiled Displays

Distributed computing is extensively used in the context of visualization mainly to enable/simplify computations needed for the visualization itself or to construct visualization environments consisting of several displays acting as a large single virtual display.

Tiled displays and the related software provide a virtual layer which is independent of the application field. Thus, the techniques represented above are generic in nature and could be used for any visualization technique. As was revealed in the related work in Section 8.2 these approaches are expensive to build. As such, a light weight architecture is presented in this section, which allows for tiled displays in the context of colored reorderable matrix visualizations.

This approach has the following advantages:

- It expands the capability of the reorderable matrix by allowing different parts of the matrix to be visualized in different displays.

- It can be used as a tool for allowing collaborative visualization of the same data set. In this way, different users sitting in different computers can explore the same data set in cooperation with each other.

- It can be used as a tool for comparison purposes by visualizing different parts of the same data set, without restricting the space for the visualization itself.

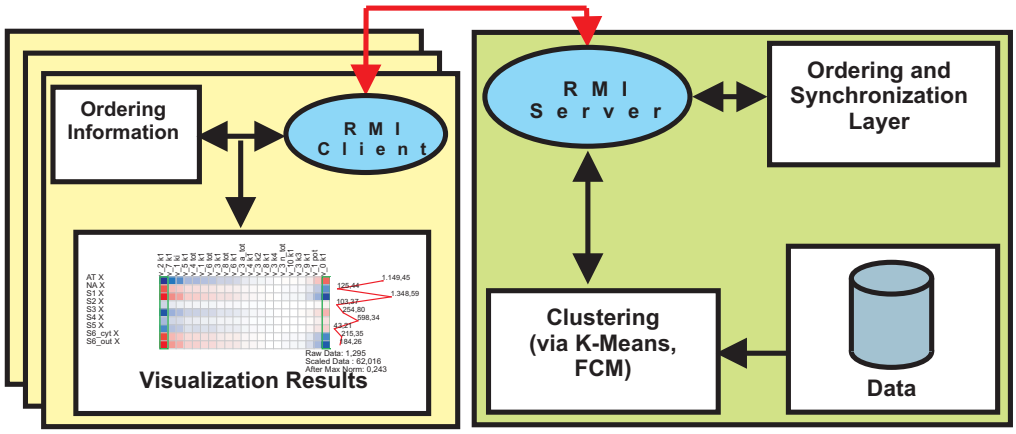


Figure 8.6: Schema of the distributed tiled displays approach

8.6.1 Architecture of the Approach

The architecture for enabling tiled and distributed displays was structured as an interactive environment, where visualization results can be viewed from one user with several displays or from several users in a collaborative way.

So far, the algorithms presented in MatVis work in a standalone way. The tiled displays are organized in a client-server configuration, several analysis clients can share a single analysis server. Remote clients can request operations on the server and visualize the results. Furthermore, they are instructed by the server which part of the visualization they will show, when they need to visualize different parts of a large visualization.

Figure 8.6 shows the architecture of the approach. The Client is a MatVis implementation which does not proceed with local analysis of data but it connects to the Server using Remote Method Invocation (RMI) and submits the necessary information for further processing. The server then passes the raw data to an analysis engine which performs the requested analysis. Once the analysis is complete, the Server returns the results back to the Client, where these results are visualized in the same way as if they were performed locally. The interactive requests the Client makes range from ordering of matrices to clustering using different algorithms or dimension reductioning.

8.6.2 Visualizing over Multiple Displays

The user may either visualize a full copy of the data set or just parts of it. For some of the visualization techniques presented in Chapter 6, e.g. dimensionality reduction techniques, it does not make sense to divide their visualization over multiple displays as these are presented as plots of points and their division over multiple displays would simply disorient the user.

In these cases, only the computation results of the dimension reductioning process are obtained from the server.

For the reorderable matrix, the distribution among several displays comes as a natural extension of the normal visualization method. Here, two cases are distinguished:

- The matrix is asymmetrical (raw data matrix) and one dimension is much smaller than the other. This scenario happens often in reality as usually analysts consider data sets with just a few features (from two to several tens of features usually) but many objects (from tens of objects to thousands of objects). In this case, the different displays can be arranged linearly and are considered by MatVis as extensions to each other. Figure 8.7 presents such an example where the visualization of a large data set is divided between two displays.

- The matrix is symmetrical or with nearly equal dimensions. This scenario happens in cases when proximity matrices such as correlation or distance matrices are considered. The displays in this case need to be arranged in a (possibly square) grid, similar to the PowerWall or other approaches for tiled displays.

For both cases, the synchronization is simplified by the fact that the RMI Server needs to maintain which parts of the permutation vector are displayed by the respective display. In the example shown in Figure 8.7, half of the data set is visualized in each display. The permutation of rows is the same over all displays. Consequently, interactive changes into one display are reflected into the other displays while the permutation of columns is local for each display.

Considering that the code is in Java using RMI, clients can run on different platforms. The two clients presented in the figure are executed in a SUSE Linux and Windows XP machine respectively. The possible reordering of columns is performed on the server side, thus reducing the load on the client. The algorithmical side of visualizations is currently performed on the Server side on a single machine, but it may be extended further to allow for the distribution of the computations on a network of computers or even a computational grid.

8.7 Summary

This chapter focused on extensions of visualization techniques presented in Chapter 6 dealing with large scale visualizations. First, issues related to filtering of data for the purpose of reducing unnecessary information were discussed. Both interactive and query-based filtering were discussed and a

Figure 8.7: Visualization of matrices with two displays

simple but robust query language for filtering the visualizations was introduced.

Furthermore, the reorderable matrix was extended with two techniques for browsing and exploring large data sets:

- An overview+detail interface, which allows navigation in the visualization by providing two different levels of details in two different windows. One is used as the master view and controls the other view which displays the details for the user.

- A focus+context based distortion view, which allows the integration of both the overview as well as details in the same interface. Both two-dimensional and one-dimensional distortion were considered in the case of the reorderable matrix.

Finally, an architecture for distributed visualization of large data sets in a (possibly) tiled display system was described. This architecture enables the extension of the capabilities of tabular visualization techniques by dividing the visualization into multiple displays. Moreover, it allows MatVis to be used as a tool for collaborative visualization and analysis of data.

9
Concluding Remarks

9.1 Summary of Contributions

In this dissertation, different techniques for analyzing data in the context of metabolic modeling were discussed. The main contributions of this thesis are organized along four areas of information visualization and data mining topics and are summarized as follows:

- Novel techniques for the visualization of metabolic networks and the related simulation data.
  - Visualization of the networks together with simulation-generated time series data, for a thorough exploration of the network dynamics. Bottlenecks and active parts of the network can be easily distinguished. Effectors, expressing activation and inhibition are also considered. The dynamic changes in concentrations of the metabolites and reaction rates in the network are emphasized using the visual variable size.
  - Prototypical visualization of metabolic networks in 3D, allowing cross-free drawing of edges representing reactions in the network.
  - Steerable drawing of complex metabolic networks, which aggregates the benefits of automatic drawing and manual drawing, for creating high quality drawings of these networks quickly. The steerability allows the user to guide the drawing process to meet biological conventions, allowing him or her to decompose the network into parts, draw the parts automatically when possible and merge them interactively into the main drawing.

All these approaches are bundled together in the visualization tool named MetVis.

- Comparison of semistructured data (XML files) in the context of SBML files storing metabolic network models. Consequently, the CustX-Diff algorithm for customizable comparison of XML files was introduced. The customization of the comparison process proceeds through specification of XPath expressions, which specify the parts of SBML document to be considered during the change detection process. The approach is generic in that it can be used in other contexts, i.e. non-SBML documents. Additionally, parts of this approach can be used to efficiently filter XML trees using the Aspect-Oriented Programming (AOP) paradigm, as described in [QF06].

- Visualization of time-varying sensitivity matrices generated during the simulation of metabolic network models. Different techniques for visualizing and analyzing them were explored, which include:

  – Tabular visualization techniques such as the reorderable matrix. A colored heatmap-like version was introduced and automatic reordering techniques both for static data as well as time-varying data were explored. Both local reordering and global reordering problems belong to the NP-hard class of problems, and consequently different heuristics for solving these problems were proposed. Local ordering was approached using Weighted Ordering, Weighted Spectral Seriation, Exhaustive Spectral Seriation, and TSP(Traveling Salesman Problem) based local search, whereas global ordering was approached using Spearman Footrule Aggregation, Barycenter Heuristic and again a TSP-based heuristic. Evaluation results of these heuristics were presented in Chapter 6. Furthermore, symmetrical derivatives of the reorderable matrix were introduced for exploring (interactively) proximity data, such as correlation matrices or the cluster membership evolution matrix. The latter was introduced to observe the dynamic evolution of the memberships of clusterings of parameters over the time.

  – Dimension reduction techniques were used to allow the visual exploration of relationships between parameters in low dimensions. Multi-dimensional Scaling (MDS) and the Sammon Mapping, combined with clustering methods, were presented as a means for creating understandable view of the high-dimensional sensitivity matrices.

  – Approaches for visualizing large-scale data were the focus of Chapter 8. Filtering methods for reducing unnecessary information were

discussed. Moreover, overview+detail interfaces and focus+ context based distortion were developed to expand the capabilities of the reorderable matrix. Finally, an architecture for distributed visualization of large data sets in a (possibly) tiled display system was introduced.

- Clustering of multidimensional time-varying data

  The techniques developed in Chapter 6 motivated us to develop a relationship-based clustering framework in Chapter 7, which relies on the accumulation of evolving pairwise similarities. It is based on the idea of combining the fuzzy partitions created during the simple fuzzy clustering process at each point of time. Issues related to the technical implementation of the approach were discussed and evaluation results both with synthetic data sets as well as time-varying sensitivity matrices were presented. This technique provides robustness and can be parallelized easily in a distributed computing environment.

9.2 Open Issues and Further Work

There are several directions for future research on issues treated in this dissertation. This section highlights some of the promising directions by discussing possible improvements of the presented visualization techniques and ensemble clustering algorithms. Furthermore, intersection points with other application areas are mentioned.

Visualization of Metabolic Networks. Open issues related to the visualization of metabolic networks are as follows:

- New approaches for visualization of networks with associated time series, for example new metaphors which can be used to animate the state of the network.

- Further development of 3D visualization techniques, focusing possibly on 3D virtual reality interfaces.

- Improvement of the steerable drawing approach. This can be achieved by incorporating data from available databases, such as KEGG [KG00]. Furthermore, more efforts should be put on the biological decomposition of metabolic networks for the purpose of drawing.

Customizable Comparison of XML Structures. Further work in this context should be focused on two tracks: extension of the subset of XPath

allowed for customizing the comparison process, e.g. by allowing predicates in the XPath expressions, and optimizing the aggregation of XPath expressions, when more than one expression is used for filtering.

Visualization of Sensitivity Matrices. Although a great part of this thesis dealt with algorithmic and visualization issues for time-varying multidimensional data in the context of sensitivity matrices, there are several areas for future research.

First, it would be of a special interest to study the behavior of the proposed methods with other kinds of data, for example gene microarrays. The current research in this direction is focused on static analysis of such data; the synergy of static methods such as clustering, classification, etc. and interactive techniques of information visualization remains to be researched. Another kind of data where the techniques, especially the reorderable matrix, could be applied, are network data representing social networks, parts of the Internet, etc. These kind of data is rarely visualized using tabular techniques and represents an interesting future research area.

Second, algorithmic problems such as reordering the matrices, either locally or globally, are NP-hard problems represent an interesting track for further research. Especially their analogy with ranking problems in web search remains a very attractive research direction.

Third, the structure of the techniques presented in Chapter 6 permits their exploitation in order to construct distributed versions of at least some of the techniques, e.g. the dimension reductioning techniques. However, special attention is needed, because the usage of distributed versions is justified only for very large data sets. Dimension reductioning techniques on the other hand, work on distance matrices, which are quadratic in size, and there is a certain overhead for transmitting large size data sets and their solutions. Consequently, distributed versions would need to consider these two contradictory facts, in order to provide a quality gain over the non-distributed versions.

Cluster Ensembles for Time-Varying Data. A distributed version of the approach, executable on a network of computers, could be of interest, since the cluster ensemble approach represents an "embarrasingly" parallel problem. Furthermore, the merging of similarities in Subsection 7.3.3 allows for unweighted merging of similarities during a time horizon. There are cases, however, when the user might be interested only in specific time frames, and for such cases a weighted approach would be more appropriate. Moreover, the exploration of filtering approaches for automatic exclusion of bad cluster results is another direction for future research. In this context, techniques related to information theory for defining the information content of clustering partitions is of special interest.

Application of Fuzzy Cluster Ensembles (FCE) in other fields, such as clustering of web documents, is also interesting. Here, fuzzy clustering could be more appropriate than hard clusters because of the nature of this data, where objects do not belong fully to a cluster but rather to a certain degree to different clusters.

Large-Scale Visualization. The object of future work in large-scale visualization techniques presented in Chapter 8 will be the evaluation of the different techniques by completing user studies for measuring empirically the goodness of approaches. Additionally, for the distributed tiled display approach, further development of the backend component by including distributed computing platforms is intended.

Bibliography

[ABH94] R.D. Appel, A. Bairoch, and D.F Hochstrasser. A New Generation of Information Retrieval Tools for Biologists: the Example of the ExPASy WWW Server. *Trends Biochem. Science*, 19:258–260, 1994.

[AHJY05] C. C. Aggarwal, J. Han, J.Wang, and P. S. Yu. On High Dimensional Projected Clustering of Data Streams. *Data Mining and Knowledge Discovery*, 10(3):251–273, 2005.

[AHWY03] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A Framework for Clustering Evolving Data Streams. In *Proc. of Int'l. Conf. on Very Large Data Bases*, pages 81–92, 2003.

[AHWY04] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A Framework for Projected Clustering of High Dimensional Data Streams. In *Proc. of Int'l. Conf. on Very Large Data Bases*, pages 852–863, 2004.

[AJL+02] B. Alberts, A. Johnson, J. Lewis, M. Ra, K. Roberts, and P. Walter. *Molecular Biology of the Cell*. Garland Publishing, 4th edition, 2002.

[And72] D.F Andrews. Plots of High Dimensional Data. *Biometrics*, 28:125–136, 1972.

[BBD05] T. Biedl, F. J. Brandenburg, and X. Deng. Crossings and Permutations. In *Graph Drawing*, pages 1–12. Springer, 2005.

[BDS03] U. Brandes, T. Dwyer, and F. Schreiber. Visualizing Related Metabolic Pathways in Two and a Half Dimensions. In *Graph Drawing*, pages 111–122, 2003.

[Ber81] J. Bertin. *Graphics and Graphic Information Processing*. Walter de Gruyter, 1981.

[Ber83] J. Bertin. *Semiology of Graphics*. University of Wisconsin Press, 1983.

[Bez76] J.C. Bezdek. A Physical Interpretation of Fuzzy ISODATA. *Transactions on Systems, Man and Cybernetics*, 6:387–389, 1976.

[BH02] J.C. Bezdek and R.J. Hathaway. VAT: A Tool for Visual Assessment of (Cluster) Tendency. In *Proceedings of the International Joint Conference on Neural Networks(IJCNN)*, pages 2225–2230. IEEE, 2002.

[Bio06] Biospace. Biospace Open Source Biology. `http://biospice.lbl.gov/`, 2006.

[BJGJ01] Z. Bar-Joseph, D.K. Gifford, and T.S. Jaakkola. Fast Optimal Leaf Ordering for Hierarchical Clustering. *Bioinformatics*, 17(Suppl.1):S22–S29, 2001.

[BJM97] F.-J. Brandenburg, M. Jünger, and P. Mutzel. Algorithmen zum Automatischen Zeichnen von Graphen. *Informatik Spektrum*, 20(4):199–207, 1997.

[BM95] I. Bloch and H. Maitre. Fuzzy Distances and Image Processing. In *Symposium on Applied Computing*, pages 570–574. ACM Press, 1995.

[BM98] B. Bederson and J. Meyer. Implementing a Zooming User Interface: Experience Building Pad++. *Software Practice and Experience*, 28(10):1101–1135, 1998.

[BMG00] B. B. Bederson, J. Meyer, and L. Good. Jazz: an Extensible Zoomable User Interface Graphics Toolkit in Java. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 171–180, New York, NY, USA, 2000. ACM Press.

[BR01] M. Y. Becker and I. Rojas. A Graph Layout Algorithm for Drawing Metabolic Pathways. *Bioinformatics*, 17(5):461–467, 2001.

[Bre04] J. Brereton. Use JavaCC to Build a User Friendly Boolean Query Language. http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0401brereton/index.html, 2004.

[Bro97] A. Broder. On the Resemblance and Containment of Documents. In *SEQUENCES '97: Proceedings of the Compression and Complexity of Sequences 1997*, page 21, Washington, DC, USA, 1997. IEEE Computer Society.

[But04] David Buttler. A Short Survey of Document Structure Similarity Algorithms. In *International Conference on Internet Computing*, pages 3–9, 2004.

[CAM02] G. Cobena, S. Abiteboul, and A. Marian. Detecting Changes in XML Documents. In *ICDE '02: Proceedings of the 18th International Conference on Data Engineering (ICDE'02)*, pages 41–52. IEEE Computer Society, 2002.

[CB95] Athel Cornish-Bowden. *Fundamentals of Enzyme Kinetics*. Portland Press, 2nd edition, 1995.

[CE99] F. Curbera and D. Epstein. Fast Difference and Update of XML Documents. In *Xtech*, 1999.

[CGM97] S. S. Chawathe and H. Garcia-Molina. Meaningful Change Detection in Structured Data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 26–37. ACM Press, 1997.

[CH88] S. Chatterjee and A.S. Hadi. *Sensitivity Analysis in Linear Regression*. John Wiley & Sons, 1988.

[Che73] H. Chernoff. The Use of Faces to Represent Points in k-Dimensional Space Graphically. *Journal of the American Statistical Association*, 68:361–368, 1973.

[CM88] W. C. Cleveland and M. E. McGill. *Dynamic Graphics for Statistics*. CRC Press, Inc., Boca Raton, FL, USA, 1988.

[CMS99] S. K Card, J. D. Mackinlay, and B. Shneiderman. *Information Visualization: Using Vision to Think*. Morgan Kaufmann, 1999.

[CNRS+01] C. Chassagnole, N. Noisommit-Rizzi, J. W. Schmid, K. Mauch, and M. Reuss. Dynamic Modeling of the Central Carbon

Metabolism of Escherichia coli. *Biotechnology and Bioengineering*, 79(1):53–73, 2001.

[CRGMW96] S. S. Chawathe, A. Rajaraman, H. Garcia-Molina, and J. Widom. Change Detection in Hierarchically Structured Information. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 493–504, 1996.

[DG77] P. Diaconis and R. Graham. Spearman's Footrule as a Measure of Disarray. *Journal of the Royal Statistical Society*, 39(2):262–268, 1977.

[DH98] M. Deza and T. Huang. Metrics on Permutations: A Survey. *Journal of Combinatorics, Information & System Sciences*, 23:173–185, 1998.

[DKNS01] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank Aggregation Methods for the Web. In *WWW10*, pages 613–622, 2001.

[DRS04] T. Dwyer, H. Rolletschek, and F. Schreiber. Representing Experimental Biological Data in Metabolic Networks. In *CRPIT '04: Proceedings of the Conference on Asia-Pacific Bioinformatics*, pages 13–20, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.

[eco] The E. coli Index. `http://ecoli.bham.ac.uk/`.

[Fal02] S. Falk. High Quality Visualization of Biochemical Pathways in BioPath. *Silico Biology*, (6), 2002.

[FB99] J.P. Fritz and K.E. Barner. Design of a Haptic Data Visualization System for People with Visual Impairments. *IEEE Transactions on Rehabilitation Engineering*, 7(3):372–384, 1999.

[FB03] X. Zh. Fern and C. E. Brodley. Random Projection for High Dimensional Data Clustering: A Cluster Ensemble Approach. In *Proc. of Int'l. Conf. on Machine Learning*, pages 186–193, 2003.

[FB04] X. Zhang Fern and C. E. Brodley. Solving Cluster Ensemble Problems by Bipartite Graph Partitioning. In *Proc. of Int'l. Conf. on Machine Learning*. ACM Press, 2004.

[Fel97] D. A. Fell. Systems Properties of Metabolic Networks. In *Proceedings of the International Conference on Complex Systems*, 1997.

[FJ02a] A. L.N. Fred and A. K. Jain. Data Clustering Using Evidence Accumulation. In *Proc. of Int'l. Conf. on Pattern Recognition*, pages 276–280. IEEE Press, 2002.

[FJ02b] A.L.N. Fred and A. K. Jain. Evidence Accumulation Clustering Based on the K-Means Algorithm. In *Proc. of Structural, Syntactic, and Statistical Pattern Recognition : Joint IAPR Int'l. Workshops SSPR 2002 and SPR 2002*, pages 442–451. Springer Verlag, 2002.

[FJ05] A. L.N. Fred and A. K. Jain. Combining Multiple Clusterings Using Evidence Accumulation. *Transactions on Pattern Analysis and Machine Intelligence*, 27(6):835–850, 2005.

[FMM+02] S. Flesca, G. Manco, E. Masciari, L. Pontieri, and A. Pugliese. Detecting structural similarities between XML documents. In *Proceedings of the Fifth International Workshop on the Web and Databases (WebDB 2002)*, June 2002.

[FMMP05] S. Flesca, G. Manco, E. Masciari, and L. Pontieri. Fast Detection of XML Structural Similarity. *IEEE Transactions on Knowledge and Data Engineering*, 17(2):160–175, 2005.

[Fod02] I.K. Fodor. A Survey of Dimension Reduction Techniques. Lawrence Livermore National Laboratory (LLNL) Technical Report, UCRL-ID-148494, 2002.

[Fre01] A. L.N. Fred. Finding Consistent Clusters in Data Partitions. In *Proc. of Multiple Classifier Systems, LNCS Volume 2096*, pages 309–318. Springer Verlag, 2001.

[Fri02] M. Friendly. Corrgrams: Exploratory Displays for Correlation Matrices. *The American Statistician*, pages 316–324, 2002.

[Fur86] G. W. Furnas. Generalized Fisheye Views. In *CHI '86: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 16–23, New York, NY, USA, 1986. ACM Press.

[GCE98] N. Gershon, S. Card, and S. G. Eick. Information Visualization Tutorial. In *CHI 98 Conference Summary on Human Factors in Computing Systems*, pages 109–110, New York, NY, USA, 1998. ACM Press.

[GF00] P. J. F Groenen and P. H. Franses. Visualizing Time-Varying Correlations Across Stock Markets. *Journal of Empirical Finance*, 7:155–172, 2000.

[GFC04] M. Ghoniem, J.-D. Fekete, and P. Castagliola. A Comparison of the Readability of Graphs Using Node-Link and Matrix-Based Representations. IEEE Symposium on Information Visualization 2004, pages 17–24. IEEE, 2004.

[GHS99] I. I. Goryanin, T. C. Hodgman, and E. Selkov. Mathematical Simulation and Analysis of Cellular Metabolism and Regulation. *Bioinformatics*, 15(9):749–758, 1999.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 2nd edition, 1979.

[GMM+03] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering Data Streams: Theory and Practice. *Transactions Knowledge and Data Engineering*, 15(3):515–528, 2003.

[GMT05] A. Gionis, H. Mannila, and P. Tsaparas. Clustering Aggregation. In *Proc. of Int'l. Conf. on Data Engineering*, pages 341–352. IEEE Press, 2005.

[GN00] E. R. Gansner and S. C. North. An Open Graph Visualization System and its Applications to Software Engineering. *Software — Practice and Experience*, 30(11):1203–1233, 2000.

[GW01] R. C. Gonzales and R. E. Woods. *Digital Image Processing*. Prentice Hall, 2001.

[Ham87] R.W. Hamming. *Numerical Analysis for Scientists and Engineers*. Courier Dover Publications, 1987.

[Hau06] M. D. Haunschild. *Methoden und Konzepte zur Unterstützung der Experimentellen Modellbildung im Metabolic Engineering*. PhD thesis, University of Siegen, 2006.

[HBP02] K. Hornbæk, B. B. Bederson, and C. Plaisant. Navigation Patterns and Usability of Zoomable User Interfaces With and Without an Overview. *ACM Trans. Comput.-Hum. Interact.*, 9(4):362–389, 2002.

[HFS+03] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J.C. Doyle, H. Kitano, A.P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, E.D. Gilles, M. Ginkel, V. Gor, I. I. Goryanin, W. J. Hedley, T. Charles Hodgman, J. H. Hofmeyr, P. J. Hunter, N.S. Juty, J. L. Kasberger,

A. Kremling, U. Kummer, N. Le Novère, L.M. Loew, D. Lucio, P. Mendes, E. Minch, E. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J.C. Schaff, B.E. Shapiro, T.S. Shimizu, H.D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, and J. Wang. The Systems Biology Markup Language (SBML): a Medium for Representation and Exchange of Biochemical Network Models. *Bioinformatics*, 19(4):524–531, 2003.

[HFTW05] M. D. Haunschild, B. Freisleben, R. Takors, and W. Wiechert. Investigating the Dynamic Behavior of Biochemical Networks Using Model Families. *Bioinformatics*, 21(8):1617–1625, 2005.

[HFWT02] M.D. Haunschild, B. Freisleben, W. Wiechert, and R. Takors. Distributed Simulation of Metabolic Networks with Model Variants. In *Proceedings of 16th European Simulation Multiconference*, pages 436–440. SCS Press, 2002.

[HJS00] M. Hereld, I. R. Judson, and R. L. Stevens. Tutorial: Introduction to Building Projection-based Tiled Display Systems. *IEEE Computer Graphics and Applications*, 20(4):22–28, 2000.

[HMPB+04] H. Hermjakob, L. Montecchi-Palazzi, G. Bader, J. Wojcik, L. Salwinski, A. Ceol, S. Moore, S. Orchard, U. Sarkans, C. von Mering, B. Roechert, S. Poux, E. Jung, H. Mersch, P. Kersey, M. Lappe, Y. Li, R. Zeng, D. Rana, M. Nikolski, H. Husi, C. Brun, K. Shanker, S. G. Grant, C. Sander, P. Bork, W. Zhu, A. Pandey, A. Brazma, B. Jacq, M. Vidal, D. Sherman, P. Legrain, G. Cesareni, I. Xenarios, D. Eisenberg, B. Steipe, C. Hogue, and R. Apweiler. The HUPO PSI's Molecular Interaction Format–a Community Standard for the Representation of Protein Interaction Data. *Nature Biotechnology*, 22(2):177–183, February 2004.

[Hun93] Lawrence Hunter. Molecular Biology for Computer Scientists. In Lawrence Hunter, editor, *Artificial Intelligence and Molecular Biology*. MIT Press, 1993.

[HW79] J.A. Hartigan and M.A. Wong. A K-means Clustering Algorithm. *Applied Statistics*, 28:100–108, 1979.

[HW03] M.D. Haunschild and W.Wiechert. Sensitivity Analysis of Metabolic Network Models. In *ASIM 2003, 17. Symposium Simulationstechnik*, pages 415–420, 2003.

[HY04] X. Hu and I. Yoo. Cluster Ensemble and Its Applications in Gene Expression Analysis. In *Proc. of Asia-Pacific Bioinformatics Conference*, pages 297–302. ACM Press, 2004.

[ID90] A. Inselberg and B. Dimsdale. Parallel Coordinates: A Tool for Visualizing Multidimensional Geometry. In *IEEE Conference on Visualization*, pages 361–378. IEEE, 1990.

[IS03] B. P. Ingalls and H. M. Sauro. Sensitivity Analysis of Stoichiometric Networks: An Extension of Metabolic Control Analysis to Non-Steady State Trajectories. *Journal of Theoretical Biology*, 222(1):23–26, 2003.

[ITT89] J. Bartholdi III, C.A. Tovey, and M.A. Trick. Voting Schemes for Which It Can Difficult to Tel Who Won the Election. *Socian Choice Welfare*, 6:157–165, 1989.

[JAKN03] S. Joshi, N. Agrawal, R. Krishnapuram, and S. Negi. A Bag of Paths Model for Measuring Structural Similarity in Web Documents. In *KDD '03: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 577–582, New York, NY, USA, 2003. ACM Press.

[jav06] Java Compiler Compiler. `https://javacc.dev.java.net/`, 2006.

[JD88] A. K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[JM97] D.S. Johnson and L.A. McGeoch. *Local Search in Combinatorial Optimization*, chapter The Traveling Salesman Problem: A Case Study, pages 215–310. Wiley and Sons, 1997.

[JMBO01] H. Jeong, S. P. Mason, A.-L. Barabasi, and Z. N. Oltvai. Lethality and Centrality in Protein Networks. *Nature*, 411:41–42, 2001.

[JMF99] A. K. Jain, M.N. Murty, and P.J. Flynn. Data Clustering: A Review. *ACM Computing Surveys (CSUR)*, 31(3):264–323, 1999.

[JTA+00] H. Jeong, B. Tombor, R. Albert, Z. Oltvai, and A. Barabasi. The Large-Scale Organization of Metabolic Networks. *Nature*, 407(6804):651–654, 2000.

[Kat05] W. Katja. SimWiz3D - Visualising Biochemical Simulation Results. In *International Conference on Medical Information Visualisation–BioMedical Visualisation*, pages 77–82, 2005.

[KBH04] R. Kosara, F. Bendix, and H. Hauser. TimeHistograms for Large, Time-Dependent Data. Proceedings of Joint Eurographics - IEEE TCVG Symposium on Visualization (VisSym 2004), pages 45–54, 2004.

[Ken71] D. G. Kendall. Seriation From Abundance Matrices. In F. R. Hodson, D. G. Kendall, and P. Tantu, editors, *Mathematics in the Archeological and Historical Sciences*, pages 215–251. Edinburgh University Press, 1971.

[KG00] M. Kanehisa and S. Goto. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Research*, 28(1):27–30, 2000.

[KGV83] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983.

[Kit02] H. Kitano. Systems Biology: A Brief Overview. *Science*, 295(5560):1662–1664, 2002.

[KK98] G. Karypis and V. Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal of Scientific Computing*, 20(1):359–392, 1998.

[Koh02] M. C. Kohn. Use of Sensitivity Analysis to Assess Reliability of Metabolic and Physiological Models. *Risk Analysis*, 22(3):623–631, 2002.

[KP94] P. D. Karp and S. Paley. Automated Drawing of Metabolic Pathways. In *International Conference on Bioinformatics and Genome Research*, pages 225–238, 1994.

[KSGG03] S. Klamt, J. Stelling, M. Ginkel, and E.D. Gilles. FluxAnalyzer: Exploring Structure, Pathways, and Flux Distributions in Metabolic Networks on Interactive Flux Maps. *Bioinformatics*, 19(2):261–269, 2003.

[KU05] W. Katja and K. Ursula. A New Dynamical Layout Algorithm for Complex Biochemical Reaction Networks. *BMC Bioinformatics*, 6(1), August 2005.

[LA94] Y. K. Leung and M. D. Aerley. A Review and Taxonomy of Distortion-Oriented Presentation Techniques. *ACM Transactions on Computer-Human Interaction*, 1(2):126–160, 1994.

[Lab06a] MIT Media Lab. Data Wall. `http://vlw.www.media.mit.edu/groups/vlw/DataWall-overview.htm`, 2006.

[Lab06b] Argonne National Labs. Active Mural. `http://www-fp.mcs.anl.gov/fl/activemural/`, 2006.

[LDS99] J.W. Lengeler, G. Drews, and H.G. Schlegel. *Biology of the Prokaryotes*. Blackwell Publishing, 1999.

[Lev66] V.I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics- Doklady*, 10:707–710, 1966.

[Lit83] R.J. Littlefield. Using the GLYPH Concept to Create User-Definable Display Formats. In *Proceedings of the National Computer Graphics Association '83*, pages 697–706, 1983.

[LRP95] J. Lamping, R. Rao, and P. Pirolli. A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies. In *Proceedings Conf. Human Factors in Computing Systems, CHI*, pages 401–408. ACM Press, 1995.

[LWW90] J. LeBlanc, M. O. Ward, and N. Wittels. Exploring N-dimensional databases. In *VIS '90: Proceedings of the 1st conference on Visualization '90*, pages 230–237, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.

[Mac67] J. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the 5th Berkeley Symposium on Mathematics Statistics and Probability*, 1967.

[Mar79] K.V. Mardia. *Multivariate Analysis*. Academic Press, 1979.

[MAR97] K. Mauch, S. Arnold, and M. Reuss. Dynamic Sensitivity Analysis for Metabolic Systems. *Chemichal Engineering Science*, 52(15):2589–2598, 1997.

[Men93] P. Mendes. GEPASI: A Software Package for Modelling the Dynamics, Steady States and Control of Biochemical and Other Systems. *Computer Applications in the Biosciences*, 9(5):563–571, 1993.

[Men00] P. Mendes. Advanced Visualization of Metabolic Pathways in PathDB. In *Proceedings of the 8th Conference on Plant and Animal Genome*, 2000.

[MFB98] C. J. Morton-Firth and D. Bray. Predicting Temporal Fluctuations in an Intracellular Signalling Pathway. *Journal of Theoretical Biology*, 192:117–128, 1998.

[MHOT06] J. B. Magnus, D. Hollwedel, M. Oldiges, and R. Takors. Monitoring and Modeling of the Reaction Dynamics in the Valine/Leucine Synthesis Pathway in Corynebacterium glutamicum. *Biotechnology Progress*, 22(4):1071–1083, 2006.

[Mic93] G. Michal. Biochemichal Pathways. `http://www.expasy.org/cgi-bin/show_thumbnails.pl`, 1993.

[MS00] E. Mäkinen and H. Siirtola. Reordering the Reorderable Matrix as an Algorithmic Problem. In *Diagrams 2000*, pages 453–467, 2000.

[MS03a] D.J. Marchette and J.L. Solka. Using Data Images for Outlier Detection. *Computational Statistics & Data Analysis*, 43:541–552, 2003.

[MS03b] W. Müller and H. Schumann. Visualization Methods for Time-Dependent Data: An Overview. In S. Chick, P.J. Sanchez, D. Ferrin, and D.J. Morrice, editors, *Winter Simulation Conference*, pages 737–745, 2003.

[MS04] G. Miklau and D. Suciu. Containment and Equivalence for a Fragment of XPath. *J. ACM*, 51(1):2–45, 2004.

[MTU00] H. Maruyama, K. Tamura, and N. Uramoto. Digest Values for DOM (DOMHASH), 2000.

[Mun97] T. Munzner. H3: Laying Out Large Directed Graphs in 3D Hyperbolic Space. In *IEEE Symposium on Information Visualization (InfoVis '97)*, pages 2–10. IEEE Computer Society, 1997.

[MW98] M. C. Minnotte and R. Webster. The Data Image: A Tool for Exploring High Dimensional Data Sets. In *Proceedings of the ASA Section on Statistical Graphics*, pages 25–33, 1998.

[NJ02] A. Nierman and H. V. Jagadish. Evaluating Structural Similarity in XML Documents. In *Proceedings of the Fifth International Workshop on the Web and Databases (WebDB 2002)*, Madison, Wisconsin, USA, June 2002.

[NJW02] A. Y. Ng, M. Jordan, and Y. Weiss. On Spectral Clustering: Analysis and an Algorithm. In T. G. Dietterich, S. Becker, and Zoubin Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[Noa05] Stephan Noack. Visualisierung und Analyse biochemischer Netzwerkmodelle. Master's thesis, Dresden University of Technology, 2005.

[NWH+06] S. Noack, A. Wahl, M. Haunschild, E. Qeli, B. Freisleben, and W. Wiechert. Visualizing Regulatory Interdependencies and Parameter Sensitivities in Biochemical Network Models. In *5th Vienna Symposium on Mathematical Modelling*. ARGESIM Verlag, (30,1: Abstract Volume, 30,2: Full Papers-CD), 2006.

[oM06] University of Minnesota. Power Wall. `http://www.lcse.umn.edu/research/powerwall/powerwall.html`, 2006.

[ONW+06] M. Oldiges, S. Noack, A. Wahl, E. Qeli, B. Freisleben, and W. Wiechert. From Enzyme Kinetics to Metabolic Network Modeling - Visualization Tool for Enhanced Kinetic Analysis of Biochemical Network Models. *Engineering in Life Sciences*, 6(2):155–162, 2006.

[PRW94] P. Pardalos, F. Rendl, and H. Wolkowicz. The Quadratic Assignment Problem: A Survey and Recent developments. In Panos M. Pardalos and Henry Wolkowicz, editors, *Quadratic assignment and related problems: DIMACS Workshop, May 20–21, 1993*, volume 16 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 1–41, Providence, RI, USA, 1994. American Mathematical Society.

[Pur02] H. C. Purchase. Metrics for Graph Drawing Aesthetics. *Journal of Visual Languages and Computing*, 13(5):501–516, 2002.

[PWR04] W. Peng, M. O. Ward, and E. A. Rundensteiner. Clutter Reduction in Multi-Dimensional Data Visualization Using Dimension Reordering. In *INFOVIS*, pages 89–96, 2004.

[QF06] Ermir Qeli and Bernd Freisleben. Filtering XML Documents Using XPath Expressions and Aspect-Oriented Programming.

In *DocEng '06: Proceedings of the 2006 ACM symposium on Document engineering*, pages 85–87, New York, NY, USA, 2006. ACM Press.

[QGF06] Ermir Qeli, Julinda Gllavata, and Bernd Freisleben. Customizable Detection of Changes for XML Documents Using XPath Expressions. In *DocEng '06: Proceedings of the 2006 ACM Symposium on Document Engineering*, pages 88–90, New York, NY, USA, 2006. ACM Press.

[QWF03] Ermir Qeli, Wolfgang Wiechert, and Bernd Freisleben. Metvis: A Tool for Designing and Animating Metabolic Networks. In *The 2003 European Simulation And Modelling Conference*, pages 333–338. EUROSIS, 2003.

[QWF04a] Ermir Qeli, Wolfgang Wiechert, and Bernd Freisleben. 3D Visualization and Animation of Metabolic Networks. In *18th European Simulation Multiconference (ESM 2004)*, pages 258–262. SCS European Publishing House, 2004.

[QWF04b] Ermir Qeli, Wolfgang Wiechert, and Bernd Freisleben. Visualization of Sensitivity Matrices Generated During Simulations of Metabolic Network Models. In *The IASTED International Conference on Applied Simulation and Modelling (ASM 2004)*, pages 583–589. ACTA Press, 2004.

[QWF04c] Ermir Qeli, Wolfgang Wiechert, and Bernd Freisleben. Visualizing Time-Varying Matrices Using Multidimensional Scaling and Reorderable Matrices. In *Proceedings of the 2004 International Conference on Information Visualization (IV2004), London, UK*, pages 561–567. IEEE, 2004.

[QWF05a] E. Qeli, W. Wiechert, and B. Freisleben. Visual Exploration of Time-Varying Matrices. In *Proceedings of the 2005 International Conference on Information Visualization (IV2005), London, UK*, pages 889–895. IEEE, 2005.

[QWF05b] Ermir Qeli, Wolfgang Wiechert, and Bernd Freisleben. The Time-Dependent Reorderable Matrix Method for Visualizing Evolving Tabular Data. In *Visualization and Data Analysis (VDA 2005)*. SPIE, 2005.

[RC02] I. Rojdestvenski and M. Cottam. Visualizing Metabolic Networks in VRML. In *IV*, pages 175–180, 2002.

[RK04] U. Rost and U. Kummer. Visualisation of Biochemical Network Simulations with SimWiz. *IEE Systems Biology*, 1(1):184–189, 2004.

[Rob98] J. C. Roberts. On Encouraging Multiple Views for Visualisation. In *Proceedings of the International Conference on Information Visualisation*, pages 8–15. IEEE Computer Society, 1998.

[Roj03] I. Rojdestvenski. Metabolic Pathways in Three Dimensions. *Bioinformatics*, 19(18):2436–2441, 2003.

[Rou87] P. Rousseeuw. Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Journal of Computational and Applied Mathematics*, 20(1):53–65, 1987.

[Sam69] J.W. Sammon. A Nonlinear Mapping for Data Structure Analysis. *IEEE Transactions on Computers*, 18(5):401–409, 1969.

[Sau00] H.M. Sauro. *Animating the Cellular Map 9th International BioThermoKinetics Meeting*, chapter Jarnac: a system for interactive metabolic analysis, pages 221–228. Stellenbosch University Press, 2000.

[SBG90] S. Smith, R. D. Bergeron, and G. G. Grinstein. Stereophonic and Surface Sound Generation for Exploratory Data Analysis. In *CHI '90: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 125–132, New York, NY, USA, 1990. ACM Press.

[Sch35] I.J. Schoenberg. Remarks to M. Fr'echet's article Sur la d'efinition axiomatique d'une classe d'espaces vectoriels distanci'es applicables vectoriellement sur l'espace de Hilbert. *Annals of Mathematics*, 36:724–732, 1935.

[SCS00] A. Saltelli, K. Chan, and E.M. Scott. *Sensitivity Analysis*. John Wiley & Sons, 2000.

[Sel77] S.M. Selkow. The Tree-to-Tree Editing Problem. *Information Processing Letters*, 6(6):184–186, 1977.

[SFF+00] D. Schikore, R. A. Fischer, R. Frank, R. Gaunt, J. Hobson, and B. Whitlock. High-Resolution Multiprojector Display Walls. *IEEE Computer Graphics and Applications*, 20(4):38–44, 2000.

[SG02] A. Strehl and J. Ghosh. Cluster Ensembles - A Knowledge Reuse Framework for Combining Multiple Partitions. *Journal of Machine Learning Research*, 3:583–617, 2002.

[SGM02] A. Strehl, J. Ghosh, and S. Merugu. A Consensus Framework for Integrating Distributed Clusterings Under Limited Knowledge Sharing. In *NSF Workshop on Next Generation Data Mining*, pages 99–108, 2002.

[SHM] C.L. Blake S. Hettich and C.J. Merz. UCI Repository of machine learning databases. `http://www.ics.uci.edu/$\sim$mlearn/MLRepository.html`.

[Shn96] B. Shneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *IEEE Visual Languages*, pages 336–343, College Park, Maryland 20742, U.S.A., 1996.

[Shn98] B. Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley, 1998.

[Sib78] R. Sibson. Studies in Robustness of Multidimensional Scaling: Procrustes Statistics. *Journal of the Royal Statistics Society, Series B*, 40(2):234–238, 1978.

[Sii03] H. Siirtola. Combining Parallel Coordinates with the Reorderable Matrix. In *Coordinated and Multiple Views In Exploratory Visualization (CMV'03)*, pages 63–74. IEEE, 2003.

[SL99] J. Schaff and L. M. Loew. The Virtual Cell. In *Pacific Symposium on Biocomputing*, pages 228–239, 1999.

[SL05] L. Stromback and P. Lambrix. Representations of Molecular Pathways: an Evaluation of SBML, PSI-MI and BioPAX. *Bioinformatics*, 21(24):4401–4407, 2005.

[SLN05] P. Saraiya, P. Lee, and C. North. Visualization of Graphs with Associated Timeseries Data. In *INFOVIS '05: Proceedings of the 2005 IEEE Symposium on Information Visualization*, page 30, Washington, DC, USA, 2005. IEEE Computer Society.

[Spä85] H. Späth. *The Cluster Dissection and Analysis Theory FORTRAN Programs Examples*. Prentice-Hall, 1985.

[SS93] G. Stephanopoulos and A.J. Sinskey. Metabolic Engineering–Methodologies and Future Prospects. *Trends in Biotechnology*, 11(9):392–396, 1993.

[Ste46] S.S. Stevens. On the Theory of Scales of Measurement. *Science*, 103:677–680, 1946.

[STT81] K. Sugiyama, S. Tagawa, and M. Toda. Methods for Visual Understanding of Hierarchical System Structures. *IEEE Transactions on Systems, Man, and Cybernetics*, pages 109–125, 1981.

[SZG+96] D. Schaffer, Z. Zuo, S. Greenberg, L. Bartram, J. Dill, S. Dubs, and M. Roseman. Navigating Hierarchically Clustered Networks Through Fisheye and Full-Zoom Methods. *ACM Transactions on Computer-Human Interaction*, 3(2):162–188, 1996.

[Tai79] K.C. Tai. The Tree-to-Tree Correction Problem. *Journal of the ACM*, 26(3):422–433, 1979.

[THT+99] M. Tomita, K. Hashimoto, K. Takahashi, T. S. Shimizu, Y. Matsuzaki, F. Miyoshi, K. Saito, S. Tanida, K. Yugi, J. C. Venter, and C. A. Hutchison III. E-CELL: a Software Environment for Whole-Cell Simulation. *Bioinformatics*, 15:72–84, 1999.

[TJP03] A. Topchy, A. K. Jain, and W. Punch. Combining Multiple Weak Clusterings. In *Proc. of Int'l. Conf. on Data Mining*, pages 331–338. IEEE Press, 2003.

[Tuf83] E. R. Tufte. *The Visual Display of Quantitative Information.* Graphics Press, Cheshire, CT, USA, 1983.

[TWH00] R. Tibshirani, G. Walther, and T. Hastie. Estimating the Number of Clusters in a Dataset via the Gap Statistic. Technical Report 208, Dept. of Statistics, Stanford University, 2000.

[Uni06] Princeton University. Scalable Display Wall. `http://www.cs.princeton.edu/omnimedia/`, 2006.

[VBP93] A. Varma, B. Boesch, and B. Palsson. Biochemical Production Capabilities of Escherichia Coli. *Biotechnology and Bioengineering*, 42(1):59–73, 1993.

[Wal00] P. Walser. idx3d Library. `http://www.idx3d.ch/idx3d/idx3d.html`, 2000.

[War00] C. Ware. *Information Visualization: Perception for Design.* Morgan Kaufmann, 2000.

[WDC03] Y. Wang, D. J. DeWitt, and J. Cai. X-Diff: An Effective Change Detection Algorithm for XML Documents. In *International Conference on Data Engineering*, pages 519–530. IEEE Press, 2003.

[Wes01] H. V. Westerhoff. The Silicon Cell, Not Dead but Live! *Metabolic Engineering*, 3:207–210, 2001.

[WF01] A. Wagner and D. Fell. The Small World Inside Large Metabolic Networks. *Proceedings of Royal Society, London*, 268:1803–1810, 2001.

[Wie02] W. Wiechert. Modeling and Simulation: Tools for Metabolic Engineering. *Journal of Biotechnology*, 94:37–63, 2002.

[WSK+00] B. Wei, C. T. Silva, E. Koutsofios, S. Krishnan, and S. C. North. Visualization Research with Large Displays. *IEEE Computer Graphics and Applications*, 20(4):50–54, 2000.

[WT04] W. Wiechert and R. Takors. *Validation of Metabolic Models: Concepts, Tools, and Problems*, pages 277–320. Horizon Scientific Press (Horizon Bioscience), 2004.

[YH38] G. Young and A.S. Householder. Discussion of a Set of Points in Terms of Their Mutual Distances. *Psychometrica*, 3:19–22, 1938.

[YK03] Y. Yang and M. Kamel. Clustering Ensemble Using Swarm Intelligence. In *Swarm Intelligence Symposium*, pages 65–71. IEEE Press, 2003.

[Zah71] C. Zahn. Graph-theoretical Methods for Detecting and Describing Gestalt Clusters. *Transactions on Computers*, 20(1):68–86, 1971.

[Zha93] K. Zhang. A new editing-based distance between unordered labeled trees. In *CPM '93: Proceedings of the 4th Annual Symposium on Combinatorial Pattern Matching*, pages 254–265, London, UK, 1993. Springer-Verlag.

[ZS89] K. Zhang and D. Shasha. Simple Fast Algorithms for the Editing Distance Between Trees and Related Problems. *SIAM J. Comput.*, 18(6):1245–1262, 1989.

[ZSS92] K. Zhang, R. Statman, and D. Shasha. On the Editing Distance Between Unordered Labeled Trees. *Inf. Process. Lett.*, 42(3):133–139, 1992.

[ZTGFR02] Y. Zeng, J. Tang, J. Garcia-Frias, and G. R. Rao. An Adaptive Meta-clustering Approach: Combining the Information from Different Clustering Results. In *Proc. of Bioinformatics Conference*, pages 276–287. IEEE Press, 2002.