Service-Oriented Discovery of Knowledge

Foundations, Implementations and Applications

Proefschrift

ter verkrijging van
de graad van Doctor aan de Universiteit Leiden,
op gezag van de Rector Magnificus prof. mr. P.F.van der Heijden,
volgens besluit van het College voor Promoties
te verdedigen op donderdag 18 november 2010
klokke 16.15 uur

door

Jeroen Sebastiaan de Bruin
geboren te Rotterdam
in 1981

Promotie Commissie

Promotor:	Prof. Dr. J.N. Kok
Overige leden:	Prof. Dr. T.H.W. Bäck
	Prof. Dr. F. Arbab
	Prof. Dr. M. Vazirgiannis, Athens University
	Dr. W.A. Kosters

The work in this thesis has been carried out under the auspices of the research school IPA (Institute for Programming research and Algorithmics).

This work was part of the BioRange programme of the Netherlands Bioinformatics Centre (NBIC), which is supported by a BSIK grant through the Netherlands Genomics Initiative (NGI).

Be epic.

Contents

Chapter 1

Introduction

Ever since scientists recognized computers as an invaluable support tool for their research there has been a rapidly increasing demand for technologies that allow for more data to be gathered, stored, and processed. Statistics in the past have suggested that worldwide data volumes are doubling every two to three years [LV03], an estimate which is still reasonnably accurate today, even more so for the scientific community, where gathering huge amounts of data seems to be more the rule than an exception.

The advances in computer science technologies gave rise to a paradigm shift in the way we perform and think about research. No longer do experiments need to be conducted in a *hypothesis-driven* fashion only, where a scientist has an idea, formulates a hypothesis, and tries to validate by experimenting. Rather, the current trend is perform science in a *data-driven* way; the scientist collects as much data as possible on a specific problem environment, looks for emerging patterns, interprets these patterns, and relates them to the current knowledge.

While this new paradigm certainly has its conveniences, it also has its share of problems and difficulties, some of which are solved, some that still need (better) solutions. One of those problems is referred to as the *data explosion*, a dramatic growth in the generation of data. This was especially noticeable in medical and physical sciences, where measurement equipment emerged that had higher resolutions and more sensitive measurement capabilities, thereby able to generate and store massive amounts of data.

As more and more data is being generated and gathered, the demand for programs and algorithms that can help interpret this data also grows. Since data volumes now span gigabytes or even terabytes, analyzing this data becomes a task that could not be done without the help of a computer. Moreover, traditional methods of analyzing data such as statistics do not always suffice anymore, since statistics do not extract hypotheses from data. The demand for such possibilities has given rise to a new field of research in computer science called *Data Mining*, more formally known as *Knowl-*

edge Discovery in Databases (KDD) [FPSS96].

KDD is the process of applying various methods from scientific fields such as artificial intelligence, statistics and data processing to data, with the intention of uncovering hidden knowledge or behavior [KS05]. In this context, the term knowledge refers to patterns, which are bits of information that summarize a larger collection of data. As data collections grow bigger, these patterns become more important and form hypotheses within the data-driven paradigm.

Given that the size and variety of machine-readable datasets have increased dramatically, it seems likely that an equal, or at least proportional increase of processing power is necessary to perform data mining on such gigantic collections, power that goes beyond a single machine. As a result, new technologies have been developed to allow parallel and remote computing, using multiple computers to work together on a single task or problem.

In this thesis we investigate how relatively new techniques in software engineering can help improve knowledge discovery (KD) in terms of performance and ease of design and use. We use a paradigm called *service orientation*, which is a relatively new technique to perform distributed computing, and demonstrate how different approaches to KD can be assisted by this technique. We further demonstrate how service orientation can speed up the creation of KD experiments as well as their execution, and improve KD results.

The rest of this chapter is organized as follows: in Section 1, we present a motivation for our research and our specific use cases. In Section 2, we give an overview of this thesis, briefly describing each chapter. Finally, in Section 3. we present a list of the author's publications, whose combined effort forms the foundation of this thesis.

1.1 Motivation

When the author started his research, the project was about research on inductive databases, thereby finding efficient ways to store, retrieve and mine on data and patterns. These inductive databases were to be used in a biological or bio-informatics setting, meaning that the research was directed especially to the problems and demands of these fields, such as dealing with huge amounts of (possibly distributed and heterogeneous) data, as well as making these databases user-friendly enough for biologists and bio-informaticians.

As more research was conducted on the problems and challenges of the bioinformatics field, the research questions slightly changed. It became clear that a single inductive database would not suffice for research problems in the bio-informatics field, certainly not for microarray and other genomics experiments, and topics such as remote processing and concurrency became integral to the research. As a result, the focus shifted from inductive database technology and research, which was already being researched by multiple institutes at that time, to applying software engineer-

ing technologies that supported remote and concurrent processing, which would allow for faster experimentation within different methods of KD, including inductive databases.

As the search for software engineering technologies progressed, it became apparent that service orientation, a relatively new paradigm within the software engineering community, was best suited for the new research focus, because it potentially fulfilled all the desired criteria, and because other upcoming technologies in the bioinformatics community started to make use of service orientation as well. Therefore, it seemed more important than ever to explore service orientation in this context, optimizing it for fast experimentation as well as ease of use.

The explorations of service orientation needed to be approached from two sides. On one side, the author wanted to present some guidelines and best-practices on how to use service orientation in the design of two different KD methods that were very actively researched at the time, inductive database and scientific workflows. On the other side, the author wanted to take a critical look at current technologies that actually used service orientation and web services, which is the standard that is currently used most, and see how and where they could be improved in terms of performance and efficiency.

A final yet vital part was to create or improve an application set in the biology or bio-informatics context, as well as finding suitable data to experiment on. The author came across the work of Igor Trajkovski and Nada Lavrac, who had both worked on an application that performed subgroup discovery on genes, and who wanted to offer it as a web service. It seemed as a good start to apply the author's research on service orientation. The author began by re-implementing the original application using web services, and gradually modified and extended it into the Fantom service, which is the web service that combines the author's research on web services and data mining in bio-informatics.

Since the author wanted Fantom to be generic, making it suitable for a range of problems and problem domains instead of specific ones, the author wanted to support a range of data sources. Therefore, one use case is a microarray experiment, and the other one is a Single-Nucleotide Polymorphism experiment. While both are different experiments, the Fantom service can work with both, since the outcome of the experiments can be transformed into a ranking of unique entities, or identifiers, with scores attached to them. Note that despite the fact that our research is primarily concentrated on biology and bio-informatics, the Fantom service is generic enough to be set in any domain, as long as a ranked list of items and ontologies are available, as well as a mapping between the items in the list and ontological concepts.

1.2 Thesis Outline

Analoguous to its subtitle, this thesis has been divided into three parts: foundations, implementations and applications. The first part, foundations, covers chapters 2, 3 and 4. These chapters explain basic techniques and terminology, and present different viewpoints on data mining that have been proposed and researched in the past few years. In these chapters we investigate how service orientation could fit into, or even improve, different data mining viewpoints and techniques.

In Chapter 2, we discuss the basics of software engineering, thereby focussing on software reuse. We present a short history and demonstrate how the need for software reuse has driven the software engineering field to its current state. We also discuss service orientation, the central paradigm of this thesis. Next, we discuss the concept of data mining, providing definitions and relations to other scientific fields, and present an overview of how a data mining process typically works. We also give an overview of subgroup discovery, and briefly discuss distributed knowledge discovery.

In Chapter 3, we investigate inductive databases. We present a framework that combines data mining, patternbases and databases into an inductive database, which is a database that supports data mining in its query language. We propose design principles for inductive querying and a framework for the fusion of databases and patternbases to transparently form an inductive database. We also present scenarios to demonstrate how inductive databases benefit knowledge discovery and give a concrete example showing an advantage of mining both the patterns and the data. Finally, we theorize on how service orientation can fit within the suggested frameworks, and what improvements are possible.

In Chapter 4 we investigate how the service-oriented paradigm benefits knowledge discovery in scientific workflows. We compare the non-service-oriented, constructed process model with the service-oriented orchestrated process model, and point out the benefits of service-oriented technology in scientific workflows. After that, we propose a guidance model for the design of a service-oriented knowledge discovery process, and provide guidelines for individual knowledge discovery service design based on the types of functionalities it requires. We also provide a use case to show the application and benefits of the proposed model and guidelines in practise.

The second part, implementations, covers chapters 5 and 6. These chapters are technical in nature since they provide implementation details on the Fantom service, as well as an overview of the applications using the Fantom web service that were created for the optimization of rule pruning and threshold determination.

In Chapter 5 we discuss the Fantom service. We give insights into its implementation, providing algorithms used in all the phases of rule generation, as well as algorithms that handle rule pruning and clustering, and ontology creation. We also discuss the diverse inputs that the Fantom service expects, what kind of scoring measures it calculates, and what kind of output it delivers. To illustrate the performance of

the Fantom service, we also present some statistics concerning speed and rule pruning, which were collected by applying the Fantom service to a well-known public microarray study.

In Chapter 6 we continue our discussion on the Fantom algorithm by embedding it into larger workflows. We present two applications that use multiple instances of the Fantom service simultaneously to perform rule optimization and threshold calculation of multi-class problems. We use the principle of statistical exact testing and perform distributed computing with Fantom to further prune rules in the output of Fantom. To illustrate the effectiveness of the distributed application of Fantom, we performed another experiment on the microarray study used in Chapter 5, and show how effective exact pruning can be on top of the pruning performed in the Fantom service.

The third and final part, applications, covers chapters 7 and 8. In these chapters we discuss the application of the Fantom service on several life-science data sets, with various settings. In each chapter we discuss the biological backgrounds of the data set, and the study that it was part of. For each of the experiments conducted, we discuss primarily performance of rule generation, pruning, and clustering, although we provide the experts' opinions on the resulting rules in lesser detail as well.

In Chapter 7 we perform experiments using the Fantom service on microarray expression data obtained from samples taken from mice with cardiac overexpression of the transcription factor TBX3. We briefly discuss the biology of genes and genomes, and provide information on the mouse heart study and microarray technology. We perform multiple types of experiments, and for each of these experiments we apply exact pruning on the results. Finally, we present performance measurements on all experiments, as well as pruning and exact pruning statistics.

In Chapter 8 we perform experiments using the Fantom service on data that was obtained from a Single-Nucleotide Polymorphism (SNP) study done on human depression. We discuss what SNPs are, and why they are important. We also discuss the human depression study, and give background information on human depression where relevant. We conduct two different experiments on the data sets available. In one experiment we let Fantom mine the SNP rankings directly, and in another experiment we let Fantom mine on gene rankings that were extracted from the SNP ranking. We present performance measurements of the Fantom service for both sets, as well as pruning and clustering statistics.

Apart from these eight chapters, there are also three appendices. In Appendix A, we discuss the formats of all the mappings and data sources that the Fantom service relies on. These include interaction mappings, key mappings, ontology mappings, and the ontology format itself. In Appendix B, we present the mathematical backgrounds of the Enrichment Score function. We define its mathematical properties, and present an algorithm to calculate the maximum potential score for a certain subgroup size of a rule. Finally, in Appendix C, we present a user manual of the Fantom application.

1.3 Publications

The chapters 3, 4, and 5 of this thesis are based on the following publications:

Chapter 3

For chapter 3 we used two articles that are both concerned with inductive databases. For the first part of the chapter we used the following paper:

> Jeroen S. de Bruin and Joost N. Kok
> *Towards a Framework for Knowledge Discovery*
> In the Proceedings of IFIP PPAI 2006, pages 219–228
> Santiago de Chile, Chile, August 2006

In this paper we proposed a general architecture for the implementation of inductive databases through combination of existing technologies. We also gave insights on how inductive databases could be combined with grid computing to achieve efficient and fast knowledge discovery. For the second part of the chapter we used the following paper:

> Jeroen S. de Bruin
> *Towards a Framework for Inductive Querying*
> In the Proceedings of ISMIS 2006, pages 419–424
> Bari, Italy, October 2006

In this paper we discussed the lower level querying and fusion component more in-depth. We also showed how inductive databases could speed up data mining processes through the use of constraint-based mining, where the constraints were derived from existing patterns.

Chapter 4

In chapter 4 we address issues in service-oriented computing, thereby focussing on service-oriented knowledge discovery. For the first part of the chapter we used the following article:

> Jeroen S. de Bruin, Joost N. Kok, Nada Lavrac and Igor Trajkovski
> *Towards Service-Oriented Knowledge Discovery: A Case Study*
> ECML/PKDD 2008, SoKD Workshop Proceedings, pages 1–10
> Antwerpen, Belgium, September 2008

In this paper we examined the differences between knowledge discovery processes

that are constructed and orchestrated, or composed. We outlined their differences, weaknesses and strengths. and indicated how web services could improve orchestrated knowledge discovery processes. To illustrate these benefits, we experimented with different web service implementations and presented a comparison in their execution times. We also indicated weaknesses of the workflow model that needed to be addressed to optimally accommodate data mining processes. For the second part of the chapter we used the following paper:

> Jeroen S. de Bruin, Joost N. Kok, Nada Lavrac and Igor Trajkovski
> *On the Design of Knowledge Discovery Services:*
> *Design Patterns and Their Application in a Use Case Implementation*
> In the Proceedings of Isola 2008, pages 649–662
> Porto Sani, Greece, October 2008

In the second article we took a more theoretical approach to data mining with web services. We presented a model for the design of the data mining process as a whole based on availability of other services as well as functional and relational requirements. We also presented design patterns for the design of individual services. As a use case, we used an existing solution for a gene mining problem and transformed it into a workable web service solution using our model and design patterns, and showed how efficiency, interactivity and performance was increased.

Chapter 5

Chapter 5 was based on a single publication that summarized the Fantom service. This article was:

> Jeroen de Bruin, Nada Lavrac, Joost N. Kok
> *The Fantom Service for Subgroup Discovery in Score Lists*
> ECML/PKDD 2009, SoKD Workshop Proceedings, pages 52–63.
> Bled, Slovenia, September 2009

In this article we discussed the Fantom service, including inputs, scoring functions, the use of ontologies, output and internal functionalities and optimizations of rule generation and rule pruning. To show how the service performed and to give an indication of the effect of optimizations in pruning, we performed experiments on public genome datasets to indicate how many rules were pruned, and what the effect of each optimization was in both speed and rule pruning.

Further publications

The author of this thesis was also involved in a number of other publications:

Jeroen S. de Bruin, Tim K. Cocx, Walter A. Kosters,
Jeroen F. J. Laros and Joost N. Kok
Data Mining Approaches to Criminal Career Analysis
In the Proceedings of ICDM 2006, pages 171–177
Hong Kong, China, December 2006

Yanju Zhang, Jeroen S. de Bruin and Fons J. Verbeek
miRNA Target Prediction Through Mining of miRNA Relationships
In the Proceedings of BIBE 2008, pages 1–6
Athens, Greece, October 2008

Yanju Zhang, Jeroen S. de Bruin and Fons J. Verbeek
Specificity Enhancement in microRNA Target Prediction through Knowledge Discovery
In Machine Learning, (ISBN 978-953-7) (In Press)

Part I

Foundations

Chapter 2

Background

In this thesis we take a software engineering approach to knowledge discovery, exploring and applying technologies and trends in software engineering and knowledge discovery, and combining them to improve the performance and ease of design of a knowledge discovery experiment. We present a general overview of software engineering and data mining and give an overview of the main technologies used in this thesis as well.

2.1 Introduction

This chapter is organized as follows. In Section 2, we briefly discuss a few basic concepts of software engineering, thereby focussing on software reuse. We present a short history and illustrate how the need for software reuse has influenced the software engineering field. In Section 3, we discuss service orientation, thereby explaining terminology and common techniques. We also give examples of successful web service architectures. In Section 4, we discuss the concept of data mining, give its definition and illustrate how various scientific fields contribute to it. We also give an overview of how a data mining process typically works, and give an overview of the most common classes of data mining algorithms. Next, we discuss a specific data mining field called subgroup discovery. We discuss its qualities, and the common theory and techniques that it is based on. Finally, in Section 5, we briefly discuss distributed knowledge discovery.

2.2 Software Engineering

The development of software has never been a trivial task. At the beginning of software programming, difficulties were mostly related to computer hardware limitations; programming a piece of software was a challenge due to limitations in memory size and processing power. At that time, experts held the opinion that as computers

would grow in power, programming would no longer be a problem. As it turns out, the opposite appeared to be true.

As predicted, rapid advances in computer hardware technology led to the realization of increasingly powerful computers. This, in turn, led to a demand for increasingly larger and more complex software systems. However, as software systems grew in size, they also grew in complexity, and eventually became too complex for their creators to be fully understood. As a result of this lack of understanding, software systems became unmanageable, were frequently over budget, appeared very late on the market and were often of poor quality. This was deemed the software crisis:

> *"The major cause [of the software crisis] ... that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem."* [Dij72]

In order to counter the crisis, the first NATO conference on Software Engineering was held in 1968. The term software engineering was relatively unknown then, and the intention of the conference was to force a paradigm shift in software development, from a mere craft to a full-grown engineering discipline, hence the deliberate (perhaps even provocative) use of the term software engineering. The conference was a success in that respect: the term software engineering became popular and widely used.

Software engineering is defined as the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software [ABD+04]. The goal of software engineering is to develop and apply techniques that make it possible to create high quality software with greater ease and efficiency. In short, it is the mission of the software engineering field to provide the silver bullet[1] that puts an end to the enduring software crisis.

In the decades following this historical conference, the software engineering field turned its attention towards the formation of the Software Lifecycle Process. It was argued that in order to improve software, a full and thorough understanding of software and the software lifecycle was necessary. In 1970, Royce proposed his waterfall model [Roy70], in which the software lifecycle process is viewed as flowing steadily and discretely through the phases of requirements analysis, design, implementation, validation and testing, integration, and maintenance. Consequently, research in the related domains of project management, requirements engineering, and programming and design methodologies also received an impulse.

[1] The term silver bullet was first used by J.R. Brooks Jr [Bro87]. He compared a software project to a werewolf; both change from familiar, everyday things into true horrors in an eyewink. According to ancient folklore and Hollywood movies, silver bullets are the only possible way to kill a werewolf.

Especially research on methodologies proved to have a profound influence on the modern day technology. It resulted in concepts that are widely applied in modern day programming practices. Dijkstra proposed the structured programming methodology, a programming methodology that states that programs should be split up into smaller parts, each with a single point of entry and of exit [DDH72]. In that same period, Parnas proposed the Parnas Module [Par72], adopting the methodology of information hiding. Information hiding concerns itself with hiding of design decisions in a computer program, especially those that are most subjective to change, thereby shielding other program parts from change.

Research on design methodologies also provided some well-known best practices that are used today. Perhaps the best-known methodology in this area is the notion of high cohesion / loose coupling, proposed by Yordon and Constantine [YC75]. They argued that programs should have a structured design of modules, where each module has a clear and distinct meaning in the program, containing functions that are strongly related to each other (high cohesion). Furthermore, modules should not be connected too strongly to other modules, thereby containing the effect of change in a module (loose coupling).

Driven by the research successes in programming and design methodologies, new high-level programming languages began to appear that incorporated these best-practices, including well-known programming languages such as Pascal and C. Programming and design paradigms also shifted more towards object-oriented programming and design.

In the decade that followed, *object orientation* (OO) became the predominant paradigm in the software engineering community. The community's great interest in OO resulted in the creation of OO programming languages like C++ and Java, and OO design methods [Boo93, Jac92, RBL$^+$90], which in their turn led to the creation of the current de-facto modeling standard: UML [RJB04]. For a time, it seemed that OO was the solution to the software crisis. Unfortunately, it was not perfect yet: although OO proved to be an improvement in many aspects of software engineering, there were still some issues that needed to be resolved. Thus a new paradigm emerged: component orientation (CO).

Although intended to be highly reusable, large scale reuse of classes never occurred. A reason why classes are not very reusable lies in the fact that they have a technical nature. Often, collections of classes provide a certain functionality, and the role of an individual class within that collection is unclear to anyone other than the class implementor, which greatly restricts its (re)use.

After having studied the problems of technical reuse, the CO paradigm was created to overcome them. The paradigm sets guidelines for software components that are meant to maximize their reusability. Reusability is the ability and the extend to which a software system or parts of a software system can be reused in other software systems. The increased reusability of software components makes them accessible and more attractive to a large public, allowing for reuse on a much larger scale. As

a result, rapidly expanding component markets have formed over the last few years [BBCD+98], indicating that components succeeded where object technology failed.

Since components are meant for reusability, they have well-described interfaces that allow for reuse and composition with other components, thus composition of components into systems is also easier and faster than the creation of systems. It is exactly these properties, its composability and uniform accessibility, that made component orientation become the basis for services and the service-oriented paradigm.

2.3 Service-Orientation

The service-oriented (SO) paradigm is a paradigm that specifies the design and implementation of software through the use of *services*, which are connected to each other and interact together in a *Service-Oriented Architecture* (SOA) [Gro07]. A SOA is a distributed architecture that allows a user to build an application by means of composing individual components that exist across separate (physical or logical) domains. These components are called services [HD06]. We will first discuss services and then we continue to discuss the broader SOA framework.

2.3.1 Services

We define services in the SO paradigm as follows:

Definition A *service* is an encapsulated unit of clear and distinct functionality, independent deployment, designed for orchestration, that communicates solely through contractually specified interfaces, and only has explicit dependencies.

We now discuss each part of this definition individually:

A service is an encapsulated unit

To control access to a service, and to protect it from (potential malicious) outside interference of its functionalities, a service is encapsulated. Encapsulation is a mechanism that shields the internal properties of a software unit, so that they are not directly observable or accessible by outside clients. In services, two types of encapsulation are required:

- **Implementation encapsulation**
  Implementation encapsulation, also known as implementation hiding, is a good way to protect a service from outside modifications. Functions supported by a service are black-boxes; only their external characteristics are visible to their users. These external characteristics comprise of its interface and a description of its functionality, and both should be well-described by the service's metadata description standard.

- **State encapsulation**
  State encapsulation, also known as state hiding, is the protection of the service from uncontrolled outside deregulation. To ensure this, a service seems stateless from the outside. A service can only be identified by its name and location and, as a result, cannot be distinguished from its copies (a similar definition holds for software components. Szyperski called this ”nomen est omen”, which means ”the name is the sign” [SGM02]).

A service is a unit of clear and distinct functionality

Adhering to the Parnas module principle, a service should not contain a collection of random functionalities. Rather, each service within a broader system should have its own unique role, providing clear and well-described functionalities. No two different services within a system should provide the same functionalities. A similar argument can be made for data mining, where distinct services should offer similar functionalities in terms of process and type of knowledge discovery (e.g., no two classification services should perform the exact same classification).

A service is a unit of independent deployment

A service’s design and implementation may depend on functionalities provided by its context (i.e., other services), but not on the implementation of these functionalities. For example, a service using another service which provides a queueing functionality may make no assumptions about the implementation of the queueing algorithm. This restriction ensures that a service is a separate, self-contained entity, thereby avoiding that a service is assimilated into the system, breaking when implementations of other services change.

A service is a unit designed for orchestration

In the SO paradigm, applications are constructed by orchestrating services together in an application or framework, whereby orchestration is the automated arrangement, coordination, and management of services. Although a software system can be comprised of a single service, typically it is a combination of diverse services orchestrated together to provide some joined functionality. This means that a service should always be able to be integrated into a larger system, provided that all other services in the system use the same orchestration and communication protocols. The interfaces of a service function as connection points for other services.

A service communicates solely through contractually specified interfaces

An interface is an access point for functionality, consisting of a set of named operations accompanied by the semantics of each operation. A service can be a client or

an implementor of an interface, depending on the class of the interface. We identify two classes of interfaces:

- **Provided interface**
  A provided interface is an access point that allows other clients (i.e., other services) to access functionalities provided and/or implemented by the service.

- **Required interface**
  A required interface is an access point for the service itself, enabling it to access external functionalities (thus functionalities not implemented by the service itself), which it needs in order to function properly.

These provided and required interfaces are the only means through which a service can communicate with other services, a methodology called *design by contract* [Mey92]. In this methodology, implementation is decoupled from a program's interface, whereby an interface is an annotation of the service's functionality that serves as a contract between the service user and the service provider.

A service only has explicit dependencies

Although services are designed to be as independent as possible, some dependencies, both global and local, cannot be avoided in order to function correctly. For a service to be usable by third parties, such dependencies must be explicitly mentioned in the service description. These dependencies comprise of other functionalities that must be present within the application, but also standards concerning the environment of the application itself, such as the operating system or supported hardware.

Now that we have defined what a service is, we move on to the definition of the framework in which services function, the Service-Oriented Architecture.

2.3.2 Service-Oriented Architecture

A SOA is a framework in which services are orchestrated into applications or other services. The framework dictates protocols and standards with which services can be embedded and orchestrated, be it locally or at a remote location. As a consequence, a SOA relies heavily on standards defined for communication between, and discovery and execution of services, as well as meta-data that specifies these standards for each service. A SOA can be seen as the next evolution of the CO paradigm, in the sense that services are components that can be accessed remotely as well as locally. In Figure 2.1[2] an overview of the web service framework is presented, one of the most widely used SOA frameworks nowadays.

[2] Picture adapted from IBM, http://www.ibm.com

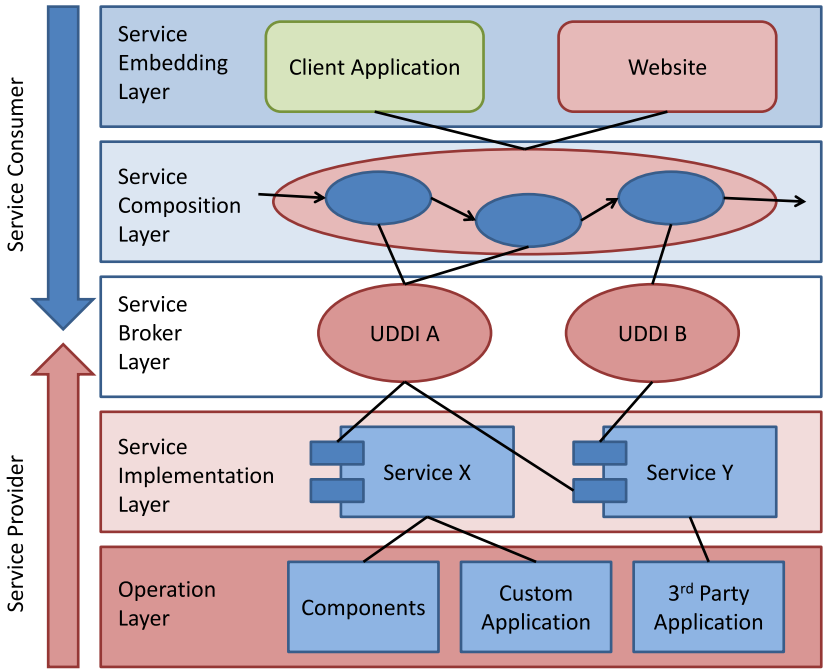


Figure 2.1: The web service framework

There are a few key points in Figure 2.1. First, in the service provider layers, services can consist of not only custom software, but also of existing solutions. This is possible because of the standardized messaging and interface formats that are part of the SOA specification.
The current standard for defining web service interfaces is the *Web Service Description Language* (WSDL) [W3C01]. WSDL is an XML-based standard that describes for each web service how the service handles incoming messages, what type of service it is, what kind of parameters it supports, and how the service interface is connected to the underlying implementation.

Another area of interest are the service consumer layers. Notice that applications are no longer constructed but instead orchestrated by putting together individual web services. This composability is partly the merit of the standardized interfaces, but also because the web service architecture is message-oriented; communication between individual components proceeds through the use of uniformly defined messages. A standard that is often used for web service message transport is the *Simple Object Access Protocol* (SOAP) [W3C07], which is an XML-based message format and transport protocol. Using both standardized ways of accessing and messaging makes an application decomposable into distinct, uniformly accessible units of com-

putation and processing, which allows for remote computing.

Finally, the last point of interest is the central layer called the services broker layer. In this layer the interfaces of the web services are offered to the consumers who search for their underlying functionality. For a user it is impossible to know the location of each service, and similarly for a provider it is impossible to know the location of all its potential users. To meet both demands, the *Universal Description Discovery and Integration* (UDDI) [MER01] was designed, which is a registry for web services offered by service providers containing all WSDL documents corresponding to interfaces of those services. In Figure 2.2[3] the web services architecture is shown.

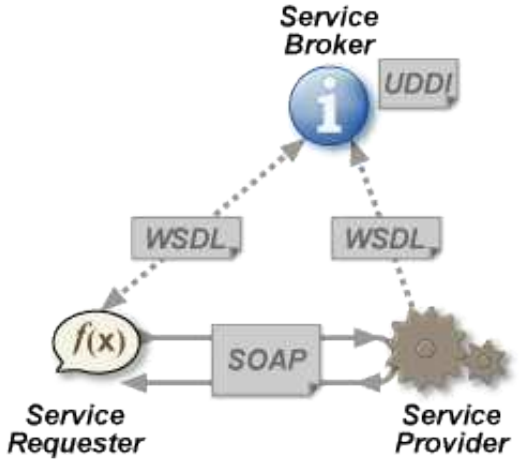


Figure 2.2: The web service architecture

As can be seen in Figure 2.2, connection of services proceeds through a UDDI service broker. The service requester sends a requester WSDL description of the service it needs. Within the UDDI, all WSDL documents of service providers are stored, and based on the requester WSDL document a list of matches is sought for, and if found, the relevant provider WSDL documents are returned to the requester. In the final stage, the service requester sends a SOAP message to the service provider based on the provider WSDL document, and after processing has taken place, the result (if any) is returned to the service requester, also through the SOAP protocol. The format of the return message is again specified in the WSDL document.

[3]Picture taken from wikipedia, http://en.wikipedia.org/wiki/Web_service

2.4 Data Mining

Data mining refers to the process of analyzing data in collections of data aiming to find *patterns*, which are bits of knowledge that summarize parts of the data [WF99]. The primary goal of data mining is to find patterns that are novel, interesting, and useful. Data mining has become increasingly important and popular since storage facilities have increased, and because data collections have become so big that it is impossible to analyze them without the help of a computer.

To uncover patters, data mining uses a variety of techniques that have roots in other disciplines such as machine learning, artificial intelligence and statistics. However, equally important is the presentation of the results, hence data mining is also influenced by computer visualization techniques.

2.4.1 Data Mining Process

In general, a data mining process can be categorized into *descriptive data mining* and *predictive data mining*. Descriptive data mining is used to generate rules that describe the data set, or subgroups of that data set, in order to gain more understanding and to formulate new theories about the data. Predictive data mining is used to generate models on the basis of known data, to formulate a prediction or theory about new data.

Originally, a data mining process was modeled as a process consisting of three sequential phases: first preprocess raw data, then mine the preprocessed data, and finally interpret the results [FPSS96]. Later, this model was modified and extended by an additional three phases in the CRoss Industry Standard Process for Data Mining (CRISPDM)[4] process model, which is shown in Figure 2.3.

As can be seen, the process is no longer linear. Research in data mining and analysis of the process uncovered that moving back and forth between different phases is inevitable, and the next phases in the process to be executed depend on the outcome of the previous ones. Furthermore, the outer circle in the figure symbolizes the cyclic nature of data mining itself, which suits the new data-driven paradigm; data mining results form new hypotheses, resulting in more business or domain understanding, and generating new questions. Hence, subsequent data mining processes will benefit from the experiences of previous ones. We present a brief overview of the individual phases below:

Business Understanding

This initial phase focuses on understanding the project objectives and requirements from a business or scientific perspective, and then converting this knowledge into a

[4] http://www.crisp-dm.org/

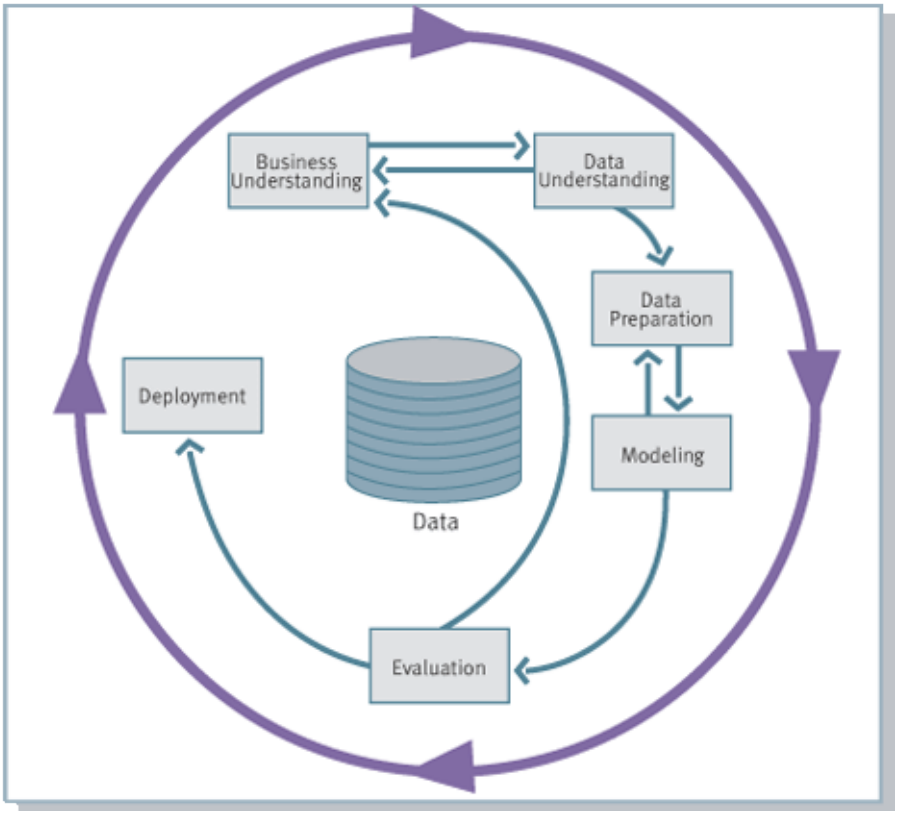


Figure 2.3: The CRISPDM process model

data mining problem definition, and an initial strategy designed to achieve the objectives.

Data Understanding

The data understanding phase comprises of data collection and familiarization with the data, to identify data quality problems, to discover first insights into the data, or to detect interesting subsets to form hypotheses for hidden information.

Data Preparation

Data preparation, also called data preprocessing, refers to the process of *cleaning*, *formatting* and *partitioning* the data.

Cleaning is the process of removing inaccurate or missing entries in the data. that might interfere with the accuracy of the experiment. Techniques such as outlier detection are commonly used in this phase.

After the data has been cleaned, often it needs to be formatted into *feature vectors*, which are vectors of (alpha-)numerical features. Usually, each entry or observation in the dataset corresponds to a single feature vector. Sometimes these vectors can get very big, in which case dimensionality reduction techniques can be used to reduce their size [LM98, GGNZ06].

Finally, the complete data set is often partitioned into a *training set* and a *test set*. The training set is used to train the algorithm (if needed), while the test set is used to verify if the patterns uncovered in the training phase are valid. The accuracy of a data mining algorithm indicates how effective it is in a certain problem domain.

Modeling

Modeling is the phase where the actual data mining takes place. Various modeling techniques are selected and applied, and their parameters are calibrated to optimal values. Typically, there are several techniques for the same data mining problem type, as we will discuss in Section 2.4.2. Some techniques have specific requirements on the form of data, which requires stepping back to the data preparation phase.

Postprocessing and Validation

In this final step of the process, patterns generated by data mining are examined for accuracy and validity. In case there are training and test sets, patterns acquired in the training set are contrasted against those resulting from the test set to see if they are present there too. When rules are specific to the training set instead of the global data set, we call this *overfitting*.

When a set of statistical inferences are simultaneously considered, errors such as hypothesis tests that incorrectly reject the null hypothesis are more likely to occur. Therefore, rules that are attributed with statistical significance or error-rates such as p-values might need to have these corrected for multiple hypothesis testing. Many of the methods [Abd07] are based on Boole's inequality, stating that if one performs n tests, each of them significant with probability p, then the probability that at least one of them comes out significant is $\leq n * p$.

Finally, when all rules are validated, a formatting phase is usually used to structure patterns, models and knowledge so that it is presented in a way that is easy to understand. Often this is done by using computer visualization techniques.

2.4.2 Data Mining Algorithms

Though there have been many data mining algorithms devised over the years, most of them fall into one or more of the following categories [FPSS96]:

- **Classification**
  Classifiers attempt to label feature vectors with classes on the basis of their

values. Classifiers are trained on the training set, and then their accuracy is measured on the test set. Since the classes of all observations are known beforehand, we call this *supervised learning*.

- **Clustering**
  Clustering has a similar goal as classification, namely to group (subgroups of) feature vectors together based on some similar entry or entries within the feature vectors. However, different from classification, classes are not known apriori, hence it is called *unsupervised learning*.

- **Regression**
  Regression analysis is a technique that tries to find a model that fits the data, e.g., a linear or hyperbolic function that fits all or most data points, minimizing the total error. Regression focusses on uncovering relationships between independent variables and dependent variables, thereby creating a model for the entire feature vector space.

- **Association learning**
  Association learning methods try to uncover relationships between (groups of) features in the feature vector. Rules uncovered usually have the form of $B \leftarrow A$, where the presence of features in group A implies the presence of features in group B. These rules usually have a *confidence* indication, though other quality measurements are also used [Omi03, AY98, BMUT97].

2.4.3 Subgroup Discovery

Subgroup discovery [Wro97, LKFT04] is a data mining method that tries to find interesting subgroups within a population of samples. It combines elements of classification and association learning [LKFT04] and regression; classification, for it tries to match a property or conjunctions of properties to a certain (sub)class, association learning because it tries to generate descriptive patterns that describe subgroups, and finally regression, because it tries to identify relations between dependent variables and independent ones.

There are also differences between subgroup discovery and classification. Classifiers usually generate rigid models for each class, that do not allow for as much flexibility in false positives as subgroup discovery does. It is also slightly different from association learning, since the rules imply subgroups and not other properties (though it can).

Patterns in subgroup discovery have the form of *Class* $\leftarrow$ *Conditions*, meaning that the description in the conditions describe (or imply) the class or subgroup. These conditions are made up of one or a conjunction of expressions that apply to all members of the class or subgroup. For example, let us assume that we have two classes,

StayIn and *GoOut*, and three properties *Weather*, *Sky* and *Wind* with diverse values. A rule could look as follows:

GoOut ← *Weather=Sunny AND Sky=Clear AND Wind=None*

In this rule it is stated that when the weather is sunny, the sky is clear and there is no wind, then people go out.

In subgroup discovery all rules are annotated with a measurement of interestingness. In [LKFT04] the *Weighted Relative Accuracy (WRAcc)* measurement is used, which is defined as follows:

WRAcc(Class ← *Condition) = p(Condition)·(p(Class|Condition)- p(Class))*

As with most measurements in subgroup discovery there are two components that try to establish a tradeoff between the generality of a rule and the deviation of the normal status or accuracy (also called ”unusualness”). In case of *WRAcc*, *p(Condition)* is the generality factor, since it indicates the relative size of a subgroup, and *p(Class|Cond)-p(Class)* is the unusualness measurement, indicating the difference between rule accuracy and expected accuracy.

Very important in subgroup discovery is efficient searching in the search space. If we use a brute force method to enumerate all the different subgroups over n properties, then the total number of enumerations would be:

$$\sum_{i=1}^{n} \frac{n!}{(n-i)!\cdot i!} = 2^n - 1$$

This means that a search quickly becomes infeasible for larger amounts of properties. To counter the explosion of the search space, usually heuristics like a beam search are used. While this is usually more efficient, the drawback is that the search is not exhaustive, leaving the chance that the optimal solution is not found.

Another important factor for efficiency is result pruning, to counter the explosion of results and redundant information. Pruning can be done in many ways, i.e., on the basis of fixed thresholds [KLJ03] or by using the properties of the measurement function [Wro97].

2.5 Distributed Knowledge Discovery

Over the last few years grid computing—the use of the memory and/or processing resources of many computers connected with each other by a network to solve computational problems—has received much attention. As more data becomes available, conventional experimentation becomes a tedious and lengthy task, often requiring hours or even days on computing a single task. To improve the speed of a computational task, grid computing is often used. It is a form of distributed computing where loosely coupled computers form a cluster to perform very large computational tasks

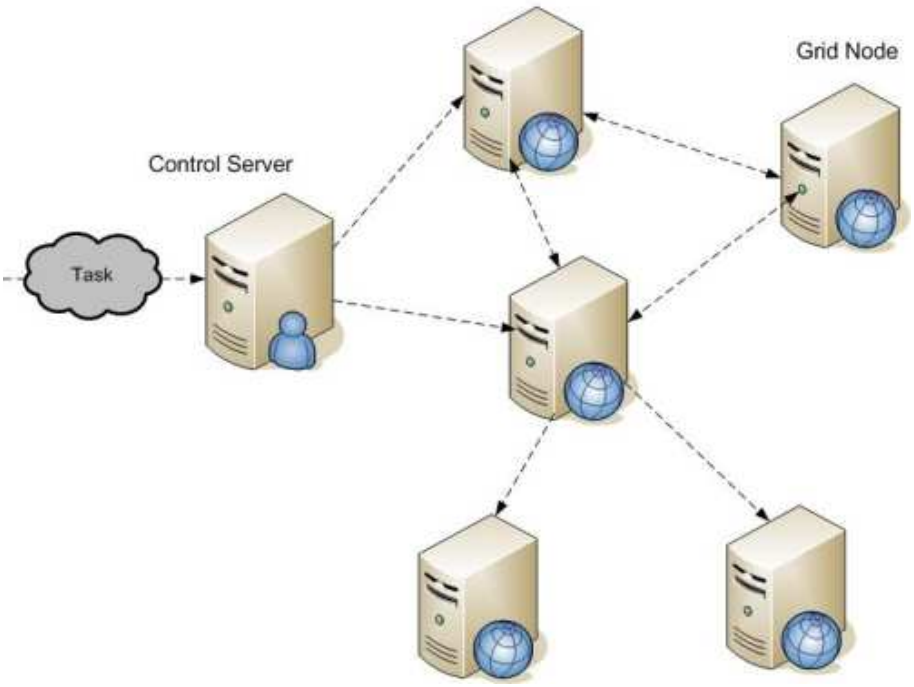


Figure 2.4: A graphical illustration of grid computing

on. A graphical depiction of grid computing can be seen in Figure 2.4[5].

Research is becoming more dependent on previous research outcomes, possibly from third parties. The complexity of modern experiments, usually requiring the combination of heterogeneous data from different fields (physics, astronomy, chemistry, biology, medicine), requires multidisciplinary efforts. This makes the quality of an *e-Science infrastructure* important. The term e-Science is used to describe computationally intensive science that is carried out in highly distributed network environments, for example experiments that deal with very large data sets, so large that grid computing is required. An e-Science infrastructure allows scientists to collaborate with colleagues world-wide and to perform experiments by utilizing resources of other organizations. A common infrastructure for experimentation also stimulates community building and the dissemination of research results. These developments apply to pure as well as applied sciences. Currently there are many efforts to construct these infrastructure, such as the Dutch Virtual Laboratory for e-science (VL-e) project[6].

Due to the increased popularity of e-Science, scientific workflows also became a

[5]Picture taken from the DAME project website, http://voneural.na.infn.it/grid_comp.html

[6]http://www.vl-e.nl/

popular topic of research. We define a *workflow* as a collection of components and relations among them, together constituting a process. Components in a workflow are entities of processing or data. They are connected by relations, which can either be data transport entities that coonects inputs and outputs from one component to another, or control flow entities that impose conditions on the execution of a component. Workflows have become increasingly popular over the last few years, since they allow a scientist to graphically construct a process of interconnected building blocks, allowing for easier experiment design and easier use of distributed resources. Taverna [MyG08] is an example of a workflow designer that allows for easy creation of workflows, possibly with remote resources. Figure 2.5 shows an example of a Taverna workflow that can be used to obtain a daily comic from a web page.

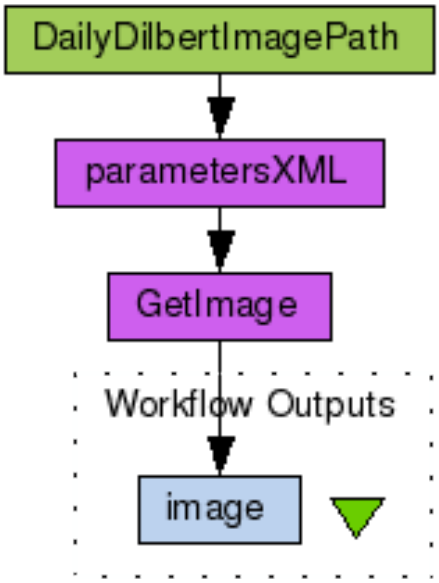


Figure 2.5: A Taverna workflow that retrieves a comic from a website

Data used in knowledge discovery is often distributed over multiple resources, which in their turn can be spread among several different logical or physical places. It is therefore important to see how current data mining algorithms can be adapted to cope with these distributions to make distributed data mining possible. This requires some form of task scheduling and runtime weighing of options, and even identification of parallelism possibilities within a process.

The problem stated above can be addressed in several ways. One way is to adapt current mining algorithms to cope with distributed data sources. Current data mining

algorithms usually address problems on a single resource, and impose a somewhat rigid structure on the input data. Relational mining algorithms, which are mining algorithms specifically developed for relational databases and thus able to work with several tables within a database, could prove to be a good basis for such adaptation.

A second way to achieve distributed data mining is through an architecture that supports a distributed environment, allowing the database itself to support and internalize remote connections to other databases. In this case, the client is unaware that the requested query or process is scheduled and executed at different locations, since to the user there appears to be only one location of data storage and processing. It is the task of the database itself to keep track of all connections and remote access protocols.

An important attribute of data mining on the grid is the ability to process data mining requests on a location other than the client or the data server(s). This poses some implications on the data mining application, since it must be able to evaluate and segment operations into sub-operations that can be simultaneously processed by multiple (distinct and/or remote) processing locations. To be able to support such parallel remote processing, it should be addressed and internalized in the distributed data mining architecture itself. The architecture should support load balancing algorithms that are efficient enough to dynamically and continuously check whether it is optimal to handle a (sub)operations locally or at another grid node.

Chapter 3

Inductive Databases

In this chapter we discuss how data mining, databases and patternbases can be integrated into inductive databases. We propose design models for the data integration part as well as the querying part of inductive databases, and reason that web services would fit well as data mining operators within the inductive querying framework. We also discuss a number of use cases in which we illustrate how knowledge discovery is performed in inductive databases, and we give concrete examples on how the use of patterns can improve data mining performance.

3.1 Introduction

The size and variety of machine-readable data sets have increased dramatically and the problem of data explosion has become apparent. Scientific disciplines are starting to assemble primary source data for use by researchers and are assembling data grids for the management of data collections. The data are typically organized into collections that are distributed across multiple administration domains and are stored on heterogeneous storage systems.

Recent developments in computing have provided the basic infrastructure for fast data access as well as many advanced computational methods for extracting knowledge from large quantities of data, providing excellent opportunities for data mining. Currently, data mining algorithms are separate software entities that extract data from databases or files, operate on the data in their own program space outside the database, and finally return results, either in a file, in a database table, or by means of a visual tool. With inductive databases, another methodology is proposed.

Inductive databases integrate databases with data mining. In inductive databases, data and patterns are handled in a similar fashion, and an inductive query language allows the user to query and manipulate patterns of interest [Rae02]. Generally these inductive query languages are seen as extensions of current query languages such as SQL or XML that, apart from atomic data operations such as insert, delete and

modify, also support data mining primitives. The challenge is to provide a persistent and consistent environment for the discovery, storage, organization, maintenance, and analysis of patterns, possibly across distributed environments.

This chapter is organized as follows. In Section 2, we discuss the principles of inductive databases and refer to related work. In Section 3, we present our framework for transparent data and pattern integration, and for inductive querying. In Section 4, we present two examples of inductive database usage, one where an inductive querying scenario will be described, another one where we will show how patterns can be used to increase data mining performance. Finally, in Section 5, we will draw some conclusions and focus on future research.

3.2 Inductive Databases

An interesting question is how the existing data mining algorithms can be elegantly integrated into current DataBase Management Systems (DBMS) without affecting performance or restricting algorithm functionality. In order to meet these requirements, the concept of so-called inductive databases [IM96] was proposed. In an inductive database it is possible to reason about and extract knowledge from the collected data in the database, as well as pose queries about inductively gathered knowledge in the form of patterns derived from that data. The subject of inductive databases has received a great deal of attention lately. A lot of research in this field is directed towards a better understanding of inductive databases [Rae02, Meo05], inductive querying and optimization [RJLM02, BKM98], and inductive query languages [BBMM04, MRB04].

An inductive database, as defined in [Rae02], is a database that stores data as well as patterns as first class objects. More formally, an inductive database $IDB(D, P)$ has a data component D and a pattern component P. Storing patterns as first-class citizens in a database enables the user to query them in a similar manner as data. The extra power lies in the so-called *crossover* queries which contain both pattern and data elements. In order to efficiently and effectively deal with patterns, researchers from diverse scientific domains would greatly benefit from adopting a Pattern-Base Management System (PBMS) in which patterns are made first-class citizens. This provides the researcher with a meaningful abstraction of the data.

The process of pattern discovery can be formalized as follows: Given a certain pattern class C and a data set D, find those patterns $p \in C$ that are sufficiently present, sufficiently true, and interesting [Meo05]. Data mining in an inductive database becomes a querying process, and the accuracy and completeness of the results, as well as the ease of finding them, depend on the expressive power of the inductive query language [IV99]. To have sufficient expressiveness in the inductive query language, it should contain primitives for data mining, data selection, pre- and postprocessing, as well as data normalization. Furthermore, it should contain opera-

tions for pattern definition and clustering, as well as constructs to extend the query language with user-made operations. A number of inductive query languages specifically targeted at association-rule mining have been proposed [IV99, BKM98].

Since all required technologies are available, our idea is to modify existing databases to support efficient pattern storage, and extend databases with an implementation of an inductive query language, thereby effectively transforming a DBMS into a DataBase Knowledge Discovery System (DBKDS). Since inductive databases provide facilities for pattern discovery as well as a means to use those patterns through the inductive query language, data mining becomes in essence an interactive querying process.

The efficiency of the data mining process also depends on the way that data is represented within the database, so a compromise must be made between efficient storage and efficient discovery. Since computer storage is becoming cheaper every day, we are inclined to prioritize a representation that facilitates efficient discovery over efficient storage. Over the past few years much research has been done on efficient pattern representation and pattern storage issues [Rae02, Meo05, BCF+08].

The studies in the PANDA project[1] have shown that the relational way of storing patterns proves to be too rigid to efficiently and effectively store patterns, since patterns often have a more semi-structured nature. To be able to support a wide variety of patterns and pattern classes, XML or variations have been explored and the results were encouraging [MP02, CMM+04]. However, more recently much work has been done on more efficient storage of patterns in relational databases [CGP06].

3.3 Inductive Database Architecture

The rationale behind a software architecture for inductive databases is clear. By creating software architectures, software becomes better, lasts longer and contains fewer errors [BCK03]. However, although much research has been done on various aspects of inductive databases, the implementation of an inductive database has received very little attention, but is still vital for performance issues (which is of paramount importance not only for inductive querying, but also KD in general), and extensibility of the database system (which has a huge impact on the data mining power of the inductive database).

Before we discuss the software architecture, we first want to address that the distinction between patterns and data is not only an intuitive one: the patterns and the data differ in a number of aspects. Raw data usually has a rigid structure, while patterns are often semi-structured. Studies in the PANDA project have shown that storing patterns in a relational way can be very inefficient, due to their semi-structured nature [CMM+04]. Therefore, we propose that an inductive database architecture that has a separate database and a separate patternbase, connected by a fusion component

[1] http://dke.cti.gr/panda/

as outlined in Figure 3.1. Note that this is a general architecture, and that there are always special cases that do not benefit from or need patternbases; a nice example are distance based methods that fit quite well with relational databases [KAH+05]).

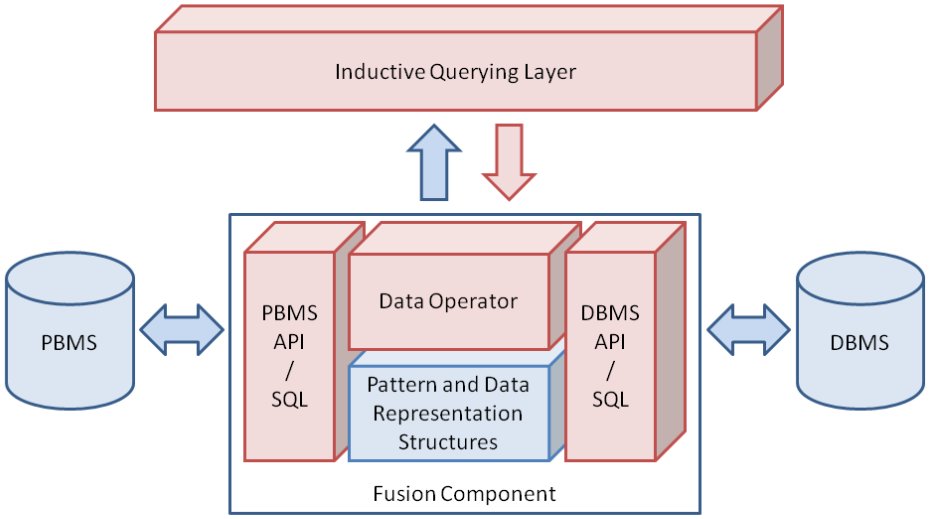


Figure 3.1: The fusion architecture

In Figure 3.1, the blue components and arrows denote data components and data flows, and the red components and arrows indicate functional components and functional flows. Let us consider a simple scenario: the user specifies a query which is processed in the inductive querying layer. As we shall see later, from here the required sub-query calls are made to the fusion layer, whereby data mining operations are supplied, as indicated by the red arrow from the querying layer to the fusion component. From here on, the necessary data and patterns are loaded through the APIs, and transformed into an internal representation. Finally, in the data operator component, the sub-query is executed.

A crucial part of this architecture are the data and pattern representation structures. According to [BCM04], a PBMS should contain three layers: a pattern layer containing the patterns, a pattern type layer containing the pattern types, and a class layer that contains pattern classes: collections of semantically related patterns. Regardless of how a pattern is represented within the patternbase, a pattern has at least the following information attached to it:

- The pattern source s, i.e., the table(s) or view(s) from which the pattern is derived.
- The pattern function f, which is the procedure used to acquire the pattern.

- The pattern parameter collection P, which is a (possibly empty) list of parameter values used by f.

The information specified above is the minimum amount of information needed to update patterns in case their source tables change. Changes can automatically be discovered and handled by database triggers supported in the DBMS query language, or by registering for them in the DBMS API. Current relational databases are unfit to represent such an architecture and XML databases have been proposed to store and represent patterns [MP02, CMM+04]. Therefore, we prefer to use an XML database for the patternbase. For representation of patterns in XML, currently the leading standard is the Predictive Model Markup Language (PMML)[2], a data mining standard for representing statistical and data mining models.

Apart from query execution, the fusion component is also responsible for the synchronization of patterns with their corresponding source data, and for maintaining data structures that allow these procedures to proceed as efficiently as possible. The fusion component should implement the following pattern and data synchronization operations:

- $Recalc(r)$, which recalculates the patterns in the patternbase affected by a change of database relation r, according to specified function f and parameter values P over source s. The function is located and known in the data mining layer.

- $Del(r)$, which deletes a pattern if (part of) its source s is no longer present in the database.

Before a query is executed, first it needs to be processed in the inductive querying layer. Currently, a few specialized inductive query languages have been proposed and implemented, such as MINE RULE [MPC98], MSQL [IV99], DMQL [HFW+96] and XMine [BCKL02]. What these languages all have in common is that they are existing SQL or XML query languages extended with data mining operators. We envision a query architecture as depicted in Figure 3.2. As can be seen in Figure 3.2, the following components are involved in the querying process:

- **Query Parser**
  All queries posed to the system first go through the query parser. Here, queries are parsed and examined, and individual relations, data mining operations and standard query types are identified and passed to the query analyzer. Identification proceeds through matching each lexical unit (e.g., a word) in the query with both the data mining operation repository and the query language typing components.

[2] http://www.dmg.org/

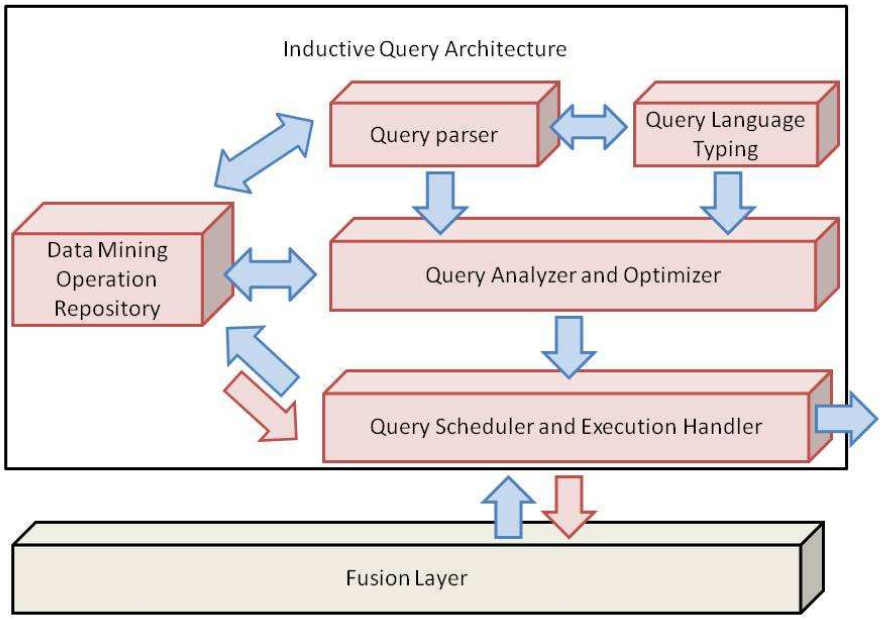


Figure 3.2: The inductive query architecture

- **Data Mining Operation Repository**
  All data mining operations are stored in the data mining operation repository. Each operation should be annotated with its lexical value (for the parser) and meta-data concerning performance and dependency indications (for the analyzer) as well as required operation parameters and output type (for the scheduler).

- **Query Language Typing**
  The query language typing component tells the parser what lexical units are part of the underlying query language of the DBMS and PBMS, so that the parser can forward those segments without having to check for data mining operations.

- **Query Analyzer and Optimizer**
  The query analyzer analyses (sub)queries to see if they can be optimized. These optimizations include logical optimizations and optimizations based concurrent execution. To optimize data mining operations, it uses meta-data provided by the operations repository.

- **Query Scheduler and Execution Handler**
  The query scheduler schedules the (sub)queries for execution, including concurrent scheduling for execution and choice of execution platform (local and/or remote), whereby it uses load balancing to come to an optimal execution profile. When this process is completed, the execution handler configures any data mining primitives according to its meta-data and send the operation and query towards the Fusion Layer (either on a remote site and/or local), where it will be handled. After receiving all the query answers, it them all to the Fusion Layer again, until a final answer is received.

As shown in Figure 3.2, the data mining operation repository is heavily involved in all steps, thereby using its meta-data to support execution, scheduling and optimization. Furthermore, data mining operations have to comply with a global typing system in order to achieve typing closure, meaning that the output type is always a subset or element of the input type. Since services are always annotated with the necessary parameters and support rigid typing schemes, we argue that services are excellent candidates for data mining operations in inductive databases. Moreover, if web services are used, remote computing could be achieved fairly easy, without having to create a custom framework.

Apart from web services as data mining operators, they can also serve as executors for other parts in the inductive querying process. For example, the query parser can be a service taking as input the query and the grammar of the inductive query language, and return errors or sub-queries. This service could be implemented remotely using a parser generator such as Bison[3].

3.4 Experimental Results

In this section we will present a use case scenario that illustrates how inductive databases can be used. We also present several experiments conducted to indicate how inductive querying can speed up a KD process through constraint-based mining [bou05].

3.4.1 Association Rule Querying Scenario

Suppose we have a database that contains transaction information on product sales for a supermarket and we want to perform some market-basket analysis to retrieve some association rules. Normally we would use the APRIORI-algorithm [AIS93] to uncover frequent item sets and association rules in the data collection. In this example we will illustrate how this can be done in an inductive database.

First, consider a query that tries to find association rules in the transaction data

[3] http://www.gnu.org/software/bison/

using a data mining operation FREQ_ITEM. When the user poses this query to the inductive database, the query scheduler uses the query analyzer and optimizer to receive an optimized set of (sub)queries.

The next task of the query scheduler is to verify whether the tables addressed in the (sub)queries are available locally at a remote site. When they are available locally, the query scheduler uses the data mining operations to execute the (sub)queries using the data retrieved from the tables. If some of the data is not available, the (sub)queries involving those data are forwarded to the remote inductive database that does have the required data, where the sub-query is sent to the query scheduler. In this example, let the required data be available locally, so the query scheduler executes the query using the data mining operation FREQ_ITEM, which results in a collection P containing frequent patterns over the transaction data set.

Let the collection P be stored in a pattern table T. Now that we have all patterns over the data set, we can use those patterns to find association rules in them using a second data mining operation ASSOC_PATTERN. This example illustrates the benefits of storing patterns as well as data. The reuse of patterns could sometimes prove to be an optimization in data mining.

Now consider the case where we want to check if the same frequent item sets hold for another transaction database. To do this, we could either formulate the same query again over the second dataset, or we can use pattern set P of the last query and apply a CHECK_PATTERN crossover data mining operation to it. This illustrates the potential power of the inductive database: intuitively, the pattern set P describes a subset of the data and thus it is more efficient to operate on those patterns instead of on the whole data set. While it is true that you can derive all patterns from the underlying data, sometimes it might be more efficient to gain patterns from the patterns already available, as is the case in the example described above.

3.4.2 Constraint-Based Inductive Querying

An inductive query is in essence a specification of constraints on data, and the resulting patterns are realizations of those constraints. Hence, by transforming patterns back into constraints, they can help improve KD performance.

Suppose we have a data set in which each entry is labeled with a class. We want to find all combinations of entries that correlate with, or are sufficiently associated to, a specific class or classes. We use the χ^2 statistic, that is we are interested in all patterns (combinations of entries, or item sets, in our case) that have a χ^2 value above a certain threshold (the threshold is to be fixed in advance). The χ^2 value is computed as follows.

Assume that there are N entries and d classes, with $(n_1 \ldots n_d)$ entries in each class. Given an item set S, construct the vector a_S=$(a_1, \ldots, a_d)$ with fractions a_i, where a_i is the fraction of examples in class i that contains S (whereby contains

indicates that the entry contains all the elements of S). Then:

$$\chi^2(a_S) = \sum_{i=1}^{d} \left(\frac{(O_{i1} - E_{i1})^2}{E_{i1}} + \frac{(O_{i2} - E_{i2})^2}{E_{i2}} \right).$$

with

- E_{i1} = $(\sum_{i=1}^{d}(a_i n_i)n_i)/N$: the expected number of elements in class i that contain S, if we assume that S and the classes are not related,
- E_{i2} = $(\sum_{i=1}^{d}((1 - a_i)n_i)n_i)/N$: the expected number of elements in class i that do not contain S, if we assume that S and classes are not related,
- O_{i1} = $a_i n_i$: the observed number of elements in class i that contain S,
- O_{i2} = $(1-a_i)n_i$: the observed number of elements in class i that do not contain S.

where n_i is the number of examples in class i and N=$\sum_{i=1}^{d} n_i$ the total number of examples.

We will compare two ways of item set generation with a χ^2 above a threshold: a brute-force method and a method that prunes the search space using patterns. In the *brute-force method* we generate every possible subset of the data. Suppose we have a table of data which contains m attributes, and for each attribute i, $1 \leq i \leq m$, where v_i is the number of possible values that attribute can have. Then the number of possible subsets s (excluding the empty set) is:

$$s = (\prod_{i=1}^{m}(v_i + 1)) - 1.$$

Since this is the number of subsets, this is also the number of times that χ^2 must be evaluated.

For the *pruning method* we use an idea presented in [NK05]. Here, it is proposed to check only frequent item sets (those item sets with a support above a class-dependent threshold) within the classes. Given a χ^2 threshold θ, a minimum frequency threshold θ_i for class i is derived:

$$\theta_i = \frac{\theta N}{N^2 - n_i N + \theta n_i}.$$

For an item set to have a χ^2 larger than θ, its threshold $a_i n_i$ should be larger than than θ_i for at least one class i, $1 \leq i \leq d$.

A χ^2 threshold is often chosen through selection of a p-value. The databases we used had either 2 or 18 classes. Table 3.1 shows the θ values for diverse p-values for

the number of classes $C = 2$ and $C = 18$.

The frequency thresholds already present two opportunities for optimization. If the frequent item sets are already present in the inductive database, then we can use the frequency threshold to prune all patterns that fall below the calculated frequencies, thereby lowering the number of times the χ^2 measure has to be calculated. If not all frequent item sets are known, then the θ_i can be used as a parameter for a frequent mining set algorithm. In our experiments we want to find out exactly how many calculations of the χ^2 measure are saved when the patterns are already present.

p-value	$C = 2$	$C = 18$
10^{-3}	10.83	40.79
10^{-5}	19.51	53.97
10^{-8}	32.84	71.41
10^{-10}	36.00	75.33

Table 3.1: θ values for a diverse number of classes and p-values

The data sets used in the experiments were obtained from the UCI [AN07]. More specific information on the data is presented in Table 3.2. For the generation of frequent item sets we used the Apriori implementation of Borgelt [Bor03].

Name	Size (N)	Classes (d)	Distribution
Mushroom	8,124	2	Classes: e(4,208) and p(3,916)
Chess KRKP	3,196	2	Classes: win(1,669) and loss(1,527)
Chess KRK	28,056	18	Classes: max(4,553) and min(27)

Table 3.2: The UCI data set descriptions

First consider the mushroom database. The data scheme contains 22 different attributes, all ranging from 2 to 12 different values per attribute. The brute-force method checks $163.5 * 10^{15}$ subsets, and thus has to do that many χ^2 evaluations. Now consider the pruning method. We first split the database into two data sets, each containing the data of one class, and then look for patterns that are significant according to the χ^2 measure. We do this for both data sets and we merge the resulting pattern sets. The resulting number of patterns is the maximum number of patterns that have to be evaluated with the χ^2 statistic (there might be patterns that are in both sets). The results of the experiments with a number of p-values, along with the reduction ratio in the number of χ^2 evaluations, are shown in Table 3.3.

As can be seen, the reduction in χ^2 evaluations is significant. Of course the number of evaluations depends on the number of classes, the number of attributes and the

number of values each attribute can take and especially the last two are quite large here. So let us analyze this method's efficiency when the number of attributes and attribute values are lower.

p-value	#patterns	#subsets	ratio
10^{-3}	525,310	$163.5 * 10^{15}$	$3.21 * 10^{-12}$
10^{-5}	466,686	$163.5 * 10^{15}$	$2.86 * 10^{-12}$
10^{-8}	408,894	$163.5 * 10^{15}$	$2.50 * 10^{-12}$
10^{-10}	399,870	$163.5 * 10^{15}$	$2.45 * 10^{-12}$

Table 3.3: The UCI mushroom data set results

Next, consider the King-Rook-King data set. This data scheme contains only six attributes, each consisting of eight possible values. For the direct method this yields a total of $531,441$ subsets. As can be seen in Table 3.4, the ratios are much higher when less attributes and attribute values are involved, but the reduction in χ^2 calculations is still a little below 90%.

p-value	#patterns	#subsets	ratio
10^{-3}	$72,990$	$531,441$	0.1373
10^{-5}	$61,996$	$531,441$	0.1167
10^{-8}	$58,018$	$531,441$	0.1092
10^{-10}	$56,388$	$531,441$	0.1061

Table 3.4: The UCI King-Rook-King data set results

Finally, we check whether many attributes with many values will result in a smaller ratio by using the King-Rook vs. King-Pawn data set. Results are displayed in Table 3.5. Since different χ^2 values yielded the same results, we also considered coverage values 0.05 and 0.1.

p-value	#patterns	#subsets	ratio
10^{-3}	254	$1.50 * 10^{17}$	$1.69 * 10^{-15}$
10^{-10}	254	$1.50 * 10^{17}$	$1.69 * 10^{-15}$
0.05	190	$1.50 * 10^{17}$	$1.27 * 10^{-15}$
0.1	126	$1.50 * 10^{17}$	$8.39 * 10^{-16}$

Table 3.5: The UCI King-Rook vs. King-Pawn data set results

Note that more optimizations are possible, both in the number of evaluations as in the evaluations themselves. In the results above the number of generated pattern were

the sum of all generated patterns per class, but we did not check if patterns occurred more than once. Another optimization is the evaluation of χ^2 itself. When generating patterns with the Apriori algorithm, the coverage, which is also used in the O_{i1} and E_{i1} of the χ^2 measure, is stored in the pattern and thus need not to be calculated again, speeding up the calculation of χ^2.

3.5 Conclusions and Future Work

We have discussed the topic of knowledge discovery in inductive databases. We provided an overview of the technology as well as definitions of the main concepts and paradigms. We also introduced an architecture for the implementation of an inductive database that is suited for computing and querying both locally and remotely and discussed its various components. More importantly, we argued that services are suited for usage in this type of knowledge discovery. We have also described how a query would be handled by the architecture, and what optimizations patterns could have for the knowledge discovery process.

In the presented use case we illustrated that constraint-based data mining in inductive databases can have a significant impact on performance. Using statistical methods to transform patterns to constraints, the number of new patterns generated in experiments could be reduced by 90% or more. Measured in time, this is not a lot when applied to the relatively small datasets of the UCI, but for bio-informatics and life-sciences, where data sets usually have a size of gigabytes or more, the impact could be significant.

We need to extensively test our architecture with various data in order to ensure it is maximized for extensibility and efficiency, two traits that can become contradicting in an architecture. Furthermore, much research still needs to be done on pattern representation and inductive query languages in areas other than frequent pattern mining.

Another area that needs to be researched thoroughly is distributed data mining within inductive databases, as well as the usage of web services within the framework. Using web services allows for easy and large-scale parallelization, but there is some overhead to be considered when executing a query on a remote site, overhead which might prove to be a burden if the query is small. Therefore, research must be done on how to represent query and data mining primitive metrics, which could help the query optimizer to make a choice between local and remote execution of queries.

Chapter 4

Service-Oriented Knowledge Discovery

Due to advances in software engineering and architecture, as well as the increased popularity of scientific workflows, new ways of performing knowledge discovery experiments can be devised. In this chapter we investigate how the service orientation paradigm and scientific workflows can improve knowledge discovery. We compare the non-service-oriented, constructed process model with the service oriented orchestrated process model, and point out the benefits of service oriented technology in workflows. After that, we propose a model for the design of a service-oriented knowledge discovery process, and provide guidelines for individual knowledge discovery service design based on the types of functionalities it requires. We also provide a use case design to show the application and benefits of the proposed model in practise.

4.1 Introduction

Despite the fact that knowledge discovery (KD) in data has proven to be valuable in many scientific fields over the last few decades, one of its main drawbacks is that setting up a KD experiment is not a simple task. Usually KD processes are very resource-intensive, requiring lots of memory space for huge amounts of data. Furthermore, they usually need one or multiple processing units to transform or mine this data. KD processes often consist of several algorithms connected together, whereby data flows from one algorithm to another.

Commonly, a KD process is created as follows: a KD researcher either implements or obtains the required algorithms and connects the in- and outputs together, executes the process, and eventually gets a result as an output. We perceive this situation as far from optimal, as it comes with quite a few problems and vulnerabilities; assuming not every scientist is a superb programmer with years of programming experience and education, implementations might suffer from errors and suboptimal

performance, and a similar argument holds for the connection of algorithms together, which mostly is done in an ad-hoc way and usually not according to a standardized format or protocol.

Instead we can consider the following scenario. Suppose a researcher wants to create a certain KD experiment involving several algorithms. The researcher only has some of these algorithms available on her own computer, knows that there are a few available at a remote location, and some that either she needs to create, or that might be found by looking for them on the internet. The ideal situation for the researcher would be to just use a search engine to look up the missing algorithms, use a tool to connect the algorithms together, and then execute the experiment.

The scenario presented above is not at all unrealistic. Due to advances in workflow management research [LLF+09, DGST09], experiments can now be designed in such a way that individual parts of the experiment can be easily connected to each other, often by using a simple graphical interface. Furthermore, the service-oriented (SO) paradigm allows for relatively simple and secure remote computing, and easy lookup of publicly available services.

In this chapter we investigate how the SO paradigm and related technologies can improve KD in scientific workflows. The SO paradigm allows users to design applications (in this context we will see a KD process as an application) in terms of individual components than can be connected to each other through standardized communication. These components can be either locally or remotely available, and can be found through public lookup facilities. We argue that combining SO with scientific workflows makes KD processes easier, faster, and better understandable.

Until recently the focus and application domain of SO technology has mostly been the commercial sector and large-scale business applications. In this part we explore the benefits and drawbacks of SO applications in KD by conducting design and implementation case studies, whereby we focus our design interests on the design of a KD service and a KD process.

This chapter is organized as follows. In Section 2, we briefly discuss some recent work related to our research. In Section 3, we examine the two scenarios in more detail, discussing their differences, weaknesses and strengths. In Section 4, we will present ideas on the design of service-oriented knowledge discovery (SOKD) services and processes, which will serve as design patterns for the implementation of our use cases, which will be discussed in Section 5. In Section 6, we will present the experimental results of the use cases, and in Section 7 we will draw a few conclusions and look at future work.

4.2 Related Work

Over the last few years distributed KD has become increasingly more popular, which generated research incentives in diverse fields of technology. In [DBG+06], dis-

tributed data mining is proposed by using peer-to-peer networks. The authors sketch a high-level introduction to peer-to-peer data mining and give some pointers and requirements for methods, as well as a theoretical example. However, a comparison with other techniques lacks, as does technological depth or formal models.

The authors of [AC06] focus on the area of text mining, and give criteria and requirements which need to be supported by good text mining tools. While they focus on their tool being embedded in other applications and address issues such as security and statelessness, they seem to only brush the topic of SO and web services as part of the tool, and present restrictions and not solutions.

In [CZW+06] the authors take a view quite similar to ours, but use Business Process Execution Language for Web Services (BPEL4WS) [BPE07] to achieve stateful long running interactions, and focus on data security through Gaussian models, while our focus lies on the design principles of web services itself within SOKD.

Finally, [GJF06] describes a framework in which web services are used for KD in databases, and outlines the framework thoroughly, as well as supported algorithms, but not the web service design and construction methodology, and thus serves as a complement to our research.

In the area of bio-informatics, workflows and service orientation have already made their introduction as well. Currently, Taverna [MyG08] is a popular workflow creation tool that supports a lot of bio-informatics services through web service APIs of the European Bioinformatics Institute (EBI)[1], BioMoby[2], Biomart[3], and the Kyoto Encyclopedia of Genes and Genomes (KEGG)[4]. There is also increased support for (distributed) algorithms in bio-informatics. The R project[5] is a free software environment for statistical computing and graphics, which supports distributed computing through a master-slave principle. Bioconductor[6] is an open source and open development software project for the analysis and comprehension of genomic data, and is largely based on the R language. Taverna provides special facilities to run R-scripts in a shell.

4.3 KD Design Scenarios

In this section we compare a first scenario where the researcher *constructs* the KD process with a second scenario where the researcher *orchestrates* a KD process using SO and workflows.

[1] http://www.ebi.ac.uk/soapalab/services/
[2] http://moby.ucalgary.ca/moby/MOBY-Central.pl
[3] http://www.biomart.org/biomart/martservice/
[4] http://soap.genome.jp/KEGG.wsdl
[5] http://www.r-project.org/
[6] http://www.bioconductor.org/

4.3.1 Scenario 1: Constructed KD Process

In this scenario the researcher constructs her own KD experiment by obtaining all required algorithms and connecting them manually. In practice, this usually means accessing diverse resources, for example the internet, to find all required algorithms and KD packages from diverse sources, and then run them one by one. If the algorithms are contained in a KD package like WEKA [Gar95], the researcher does not need to worry about intermediate data representation as the package does that for her, but if this is not the case, the researcher has to program this representation as well.

Apart from this tedious construction process, there are also a number of weaknesses that can easily break this scenario:

- **Algorithm versioning**
  When a new version of an algorithm or KD package appears, the version that the scientist uses is often not automatically updated. Instead, the scientist has to keep track of modifications herself by regularly visiting the website. Another problem with versioning is that it sometimes breaks compatibility with previous versions, which may corrupt the entire KD process.

- **Algorithm connection**
  While standard KD packages perform well on basic KD tasks, they usually lack algorithms that are tailored to a specific field of research like bio-informatics. When this is the case, the scientist has to connect all individual components of the KD process herself. This connection is often constructed in an ad-hoc fashion, making it less likely to cope with changes that might occur. Furthermore, the intermediate representations used in the connections are often suited only for the process they were designed for, so if two processes were to be combined, it is likely that one representation would need change.

- **Algorithm availability**
  It is not unlikely that some algorithms or specific algorithm implementations are not publicly available, especially when implementations are managed by commercial institutes. They want to keep their implementation proprietary knowledge, and are unwilling to distribute it or only do so at a high cost. This leaves the researcher the choice to either implement the algorithm herself, or to revise the experiment.

- **Performance**
  KD processes sometimes involve terabytes of data. When performing KD on such high volumes of data, each performance increase is important, and ideally the scientist uses the fastest machines with the best algorithms in their optimal implementation. However, the optimal implementations of algorithms are not always available or suitable for the platform that the scientist uses, making the

KD process slower. Furthermore, some algorithms run faster on specialized hardware, that is also not always available to the scientist.

4.3.2 Scenario 2: Orchestrated KD Process

In this scenario the scientist orchestrates her experiment through the use of a scientific workflow. Components in the orchestrated KD process can be present either locally or at a remote location, whereby they interact with each other in the service-oriented architecture (SOA) chosen for the application. In this case, the workflow implementation application can be seen as a SOA itself since it dictates communication and orchestration standards. The combined use of SO and scientific workflows addresses the weaknesses of the first scenario as follows:

- **Algorithm versioning**
  Keeping track of newer versions of an algorithm is no longer an issue for the scientist, since it is automatically updated on the side of the service provider. A scientist can be notified of an update, but updating can also proceed transparently. Compatibility with previous versions is also guaranteed, for the service has to adhere to a certain interface, an annotation of the service's functionality that serves as a contract between the service user and the service provider. A widely used standard for annotating web services at the moment is the Web Service Definition Language (WSDL).

- **Algorithm connection**
  Data transport between components in a scientific workflow proceeds in a standardized way, normally by using a structured transport protocol. For web services, the Simple Object Access Protocol (SOAP) is the standard protocol. The KD process will not break as long as the components adhere to the message format, which is guaranteed by the component's interface.

- **Algorithm availability**
  Since implementations of algorithms are now managed by the service provider and executed on the server end, it is safe for the providers to offer their services without running the risk of losing proprietary knowledge. These services can be polled and found by the scientist through the service provider's Universal Description, Discovery and Integration (UDDI) facility. As an example, Taverna workflows are shared in the myExperiment community[7], and are free to use for all who register there.

- **Performance**
  SOA and all related protocols discussed above are platform-independent technologies. This makes it easier for the service provider to use the programming

[7] http://www.myexperiment.org/

language and platform that is best suited, which usually leads to a performance increase. Moreover, if two services can be executed independently and are located on different machines, they can be executed in parallel in a scientific workflow, speeding up the entire KD process even more.

4.4 Service-Oriented KD Design

In this section we discuss the SO design model and patterns that we used to create the case study described in Section 4.5, and examine how these SO principles influence individual services and KD processes on a whole. First we discuss WSDL a bit more and explain how the standard influenced the design of the web service. After that, we discuss a design model for SOKD processes. Finally, we present guiding principles for individual KD service design, which we also used to design the use case.

4.4.1 WSDL and Design Implications

There are many views on the design of a KD process, ranging from a global view stating what functionalities a service in a process has, to micro-details such as what message format to use. Since the use case was designed by using WSDL, this influences our further design of a web service and a KD process as a whole. In this paper we focus on the different operation types that are defined in WSDL, and their influence on the design of KD process and a KD service:

- **Request/Response**
  In this case, the client sends a message to the service, and the service sends a message to the client in response. This is the message equivalent of a function call.

- **Solicit/Response**
  This is the reverse case of the request/response type. The service sends a message to the client, and the client sends a message to the service in response. This is often used when a service needs to poll clients.

- **Client messenger**
  The client sends a message but does not expect a message in return.

- **Server notification**
  Server notification is the exact opposite of the client messenger type. In this case, the service sends a notification to the client without expecting or waiting for an answer.

As we shall see in the remainder of this section, operation types have an impact on the entire KD process, so selecting the right type of operation is important in order to obtain a process with optimal performance.

4.4.2 Service-Oriented KD Process Design

A KD process can be seen as a workflow, whereby data flows from one unit of processing to another. Conceptually we try to map these units of processing to web services. How successful this can be done depends on the understanding of the process and the functional discreteness of individual steps. We see the design of a KD process as a three-dimensional challenge containing the logical, functional and relational views, that al influence each other. We propose the following KD process design model for SO that incorporates all these views, which is illustrated in Figure 4.1. By applying this model in the design of a SO KD process, a better understanding of the process is achieved, which leads to a better design, until both understanding and design are optimal.

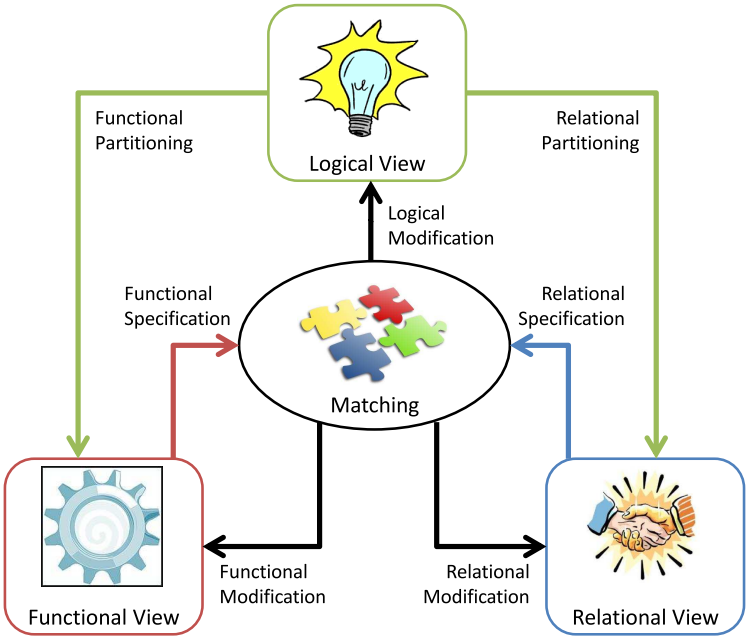


Figure 4.1: The KD process design model

We will now describe each of the views individually.

- **Logical View**
  In this view, the entire KD process is being examined to identify all services and relations in the process. This logical view is not only guided by the designer's expertise, but also on the services already available, for example, services built earlier or services publicly available through a UDDI. Ideally all

services fit together perfectly and are all available, but this is rarely the case. Therefore, choices have to be made if readily available services should be used, and how the unavailable process parts should be logically partitioned. Since a different partitioning of a KD process yields different services and relations, the partitioning will affect the functionalities of each service as well as the relations among them.

- **Functional View**
  For each service identified in the logical view, all functionalities are recorded. These functionalities will serve as a guideline for interface design and operation type selection, and will determine the nature of the relations with other functionalities. In this stage similarities between services and dissimilarities within services can be uncovered on the basis of functionality, leading to a possible joining or splitting of services.

- **Relational View**
  In this aspect of design, relations should be identified for each service with other functionalities in other services. These relations should be annotated in two dimensions: direction and usage type. The direction indicates if messages will be flowing from a service or to a service, the usage type indicates if the relation is used only once, or continuously until processing is done. Both dimensions will influence the functionality of a service, the operation type of the functionality's interface, and the content and format of the messages that will be transported. Similarities and dissimilarities in relations amongst services might also lead to a revision of the service partitioning.

- **Matching**
  This dimension is the feedback step of the model, and matches the outcome of all other phases to one another. It serves as a feedback phase for the design, and indicates if service partitionings, functionalities or relations should be modified or adapted in case of a mismatch.

4.4.3 Service-Oriented KD Service Design

In this part we focus on the functional design of a KD service, and how the design choices are expressed in the WSDL operation types.

As stated earlier, KD processes can be time-consuming, especially when large data volumes are involved. This means that any error may result in the loss of a great amount of time. Therefore, individual KD services should be designed for interaction; a scientist should get regular feedback on the progress of the process, and should at all time be able to interact with the process.

We also mentioned that a KD process is often perceived as a workflow, a sequence of computational steps whereby data flows from one step to another. This does not

mean, however, that one step should be completed in order for the next step to begin; the results that come from these actions sometimes can already be transferred to the next process phase without waiting for the service to finish processing all the data. To optimize performance as well, KD services functionality should be designed for streaming data where possible.

Having observed the facts stated above, we divided the functionalities of a KD service into three categories: Initialization, Feedback and Enactment. This classification forms a guideline for the design of a service's functionality using WSDL. So we have:

- **Initialization**
  Procedures designed in this class are expected to handle a continuous stream of messages that initialize this part of the experiment. Client messengers are usually best suited for these functions, unless initialization requires critical feedback, in which case Request/Response should be used.

- **Feedback**
  In this category methods need to be designed that provide feedback to the service client. Both Server notification or Solicit/Response method types can be used here, depending on whether the feedback is used purely for informative purposes or used to steer an interactive experiment through client intervention. Feedback is often provided iteratively, sending messages whenever an event occurs.

- **Enactment**
  This category combines the actual functionalities of the service with the feedback functionalities that report on the service's progress. Since an experiment usually is expected to return a result, a Request/Response type method is usually chosen. However, if one does not need to wait on this service in order to continue with other processing steps, a combined Client messenger and Server notification procedure could be used to let the service run asynchronously. Note that enactment can be done both atomically or iteratively, as we will see in the next section.

4.5 Experimental Setting

In this section we discuss the designs of our use cases. In all use cases, we used the same KD scenario and the same experimental hardware and software to get a fair and complete comparison between all our different implementations. The first use case was designed to compare performance between constructed processes and various implementations of composed processes. The second use case used the design patterns discussed in the previous section to re-implement the whole scenario.

4.5.1 Algorithms

For our case study we used a KD scenario described in [TZTL06]. In this scenario, microarray data is processed to identify differentially expressed genes based on a threshold score computed by the Student's t-test. This set of differentially expressed genes, together with a selection of their non-differentially expressed counterparts (both expressed in ENTREZ identifiers [MOPT05]), are then annotated with terms from the Gene Ontology (GO) [ABB+00]. In the final step these annotations, together with information about interaction amongst genes, are represented as facts and supplied to the Relational Subgroup Discovery algorithm (RSD) [LZF02].

The RSD algorithm takes a set of labelled data items and a class (in this case the classes differentially expressed and non-differentially expressed) and tries to find descriptions of subgroups of target class examples that are as large as possible, and have a significantly different distribution. However, to avoid that a certain set of data items dominates the entire rule-space, an iterative weighted covering algorithm is used to decrease weights of those items once they are collected in a rule. Based on the number of items in that rule belonging to each class and their individual weights, the quality of a rule is measured.

As a postprocessing step, rules are uniformly formatted using the GO descriptions, improving readability for expert reviewers. As an example, consider the rule below:

```
Rule 1:  Support 5, Weight: 12.0
Differential participants: [119391,1375,5287,1021]
Non-differential participants: [9950]
molecular_function(A,catalytic activity),
cellular_component(A,cytoplasmic part)
```

In this rule, a total of five genes were involved; four of them were differentially expressed, one was not, giving the rule a total weight of twelve (weights of individual genes are not mentioned in the rule). The rule itself states the common factors of all genes, whereby the genes are designated as group 'A'. The different predicates like 'molecular_function', as well as descriptors like 'catalytic activity' are all defined in GO. In this case, the correct interpretation of the rule would be:

"Subgroup 'A' of the gene collection resides in the cytoplasmic part of the cell and has as primary function catalytic activity, whereby 'A' consists of genes with ENTREZ ids 119391, 1375, 5287, 1021 and 9950"

4.5.2 KD Service Design

For our first use case, we created several different implementations for the same program. We made a web service in C# that takes as an input the parameters of the

program or executable that needs to be executed, and simply forwards them to the command-line interpreter. While this is the easiest implementation of a web service and introduces the possibility of remote processing, it still shares some of the weaknesses of a constructed KD process, since the underlying program remains prone to unexpected change in versioning, thereby possibly breaking the web service shell.

In our second implementation we modified the original program code as little as possible, to show that any program can be transformed into a web service within its own implementation domain. For the Python web service implementation we used the Python Web Service Module, and coupled the input of both services to this module. Furthermore, we extended the algorithm to use the KEGG ontology [OGS+99] as well, to show how updates can be performed without modifying the interface of a web service, making users completely oblivious of the service's implementation and updates.

For our last implementation we re-implemented all the Python code into C++ to show how web services can increase performance. The performance gain is in several dimensions here. First, applications made in programming languages like C++ and C# tend to execute faster than those made in scripting languages due to rigorous compile-time optimizations. Second, the authors are more proficient with C++ than with Python, which also yields a performance increase.

4.5.3 KD Process Design

Through our defined design pattern we redesigned the KD process using our three views:

Logical View
We identified two different web services that are used together to provide one composite service. The first service is the **GeneSelector** service that is used to compute a t-score for all genes in the microarray data, and place it either in the differential or non-differential collection. The second service is the **GeneRuleInducer** service which takes the two lists and produces rules that describe subsets of these lists that share the same terms in the GO and KEGG ontology, which are also provided in the rule.

Functional View
For each service we identify functionalities divided in the three aforementioned functional categories: Initialization, feedback and enactment. Each category of each service has its own table stating the name and the description of the functionality. The *GeneSelector* service design is shown in Table 4.1, and the *GeneRuleInducer* service design is displayed in Table 4.2.

Initialization functionalities	
Service name	Description
Probe mapper	Maps microarray probes to ENTREZ gene.
Class mapper	Maps experiment labels to classes.
Cutoff initializer	Initialises the t-score cutoff value for genes.

Feedback functionalities	
Service name	Description
Probe feedback	Provides feedback on unmapped probes.
Class feedback	Provides feedback on unmapped labels.

Enactment functionalities	
Service name	Description
t-test calculator	Calculates t-scores.

Table 4.1: The GeneSelector service design.

Initialization functionalities	
Service name	Description
Gene loader	Loads genes and their scores in the ontology structure.
Support initializer	Initializes the minimum and maximum support constraints.
Ontology loader	Loads the supplied or system-standard ontologies.
Gene-Ontology mapper	Loads the data that maps gene identifiers to ontology keys.
Gene-Interaction mapper	Loads the data that specifies interaction between genes.

Feedback functionalities	
Service name	Description
Gene list feedback	Sends a message if (part of) the list failed to load.
Rule feedback	Presents feedback on the progress of the rule miner.

Enactment functionalities	
Service name	Description
Rule miner	Initiates the rule-mining.

Table 4.2: The GeneRuleInducer service design.

Relational View
For all logical partitions we uncovered the messages that had to be sent back and forth. We specify all the relations between the individual component interfaces and the client, and whether they are iterative or not, whereby iterative relations are denoted with a *. All relations are displayed in Table 4.3.

Client to GeneSelector relations	
Message name	Description
Probemap input	Message containing probe and gene identifiers.
Classmap input	Message containing classes and labels.
Cutoff input	Message containing the user-defined t-score cutoff.
Data input*	Message containing a probe identifier and expression score per label.

GeneSelector to Client relations	
Message name	Description
Probemap output*	Message that returns a problem with the probe mapping.
Classmap output*	Message that returns a problem with the class mapping.

GeneSelector to GeneRuleMiner relations	
Message name	Description
t-score output*	Message that returns genes together with their score and class label.

Client to GeneRuleMiner relations	
Message name	Description
(Non-)Differential support input	Message that supplies the rule cutoff for (non-)differential genes.
Enactment input	Message that enacts the mining process.

GeneRuleMiner to Client relations	
Message name	Description
Genelist output*	Message specifying feedback if anything goes wrong loading the specified lists.
Miner output	Message that contains rules uncovered by the algorithm.

Table 4.3: The Relational View of the service design.

A complete overview of service connectivity and data flow is presented in Figure 4.2. Note that only those functionalities that require interaction with the user or another service are displayed.

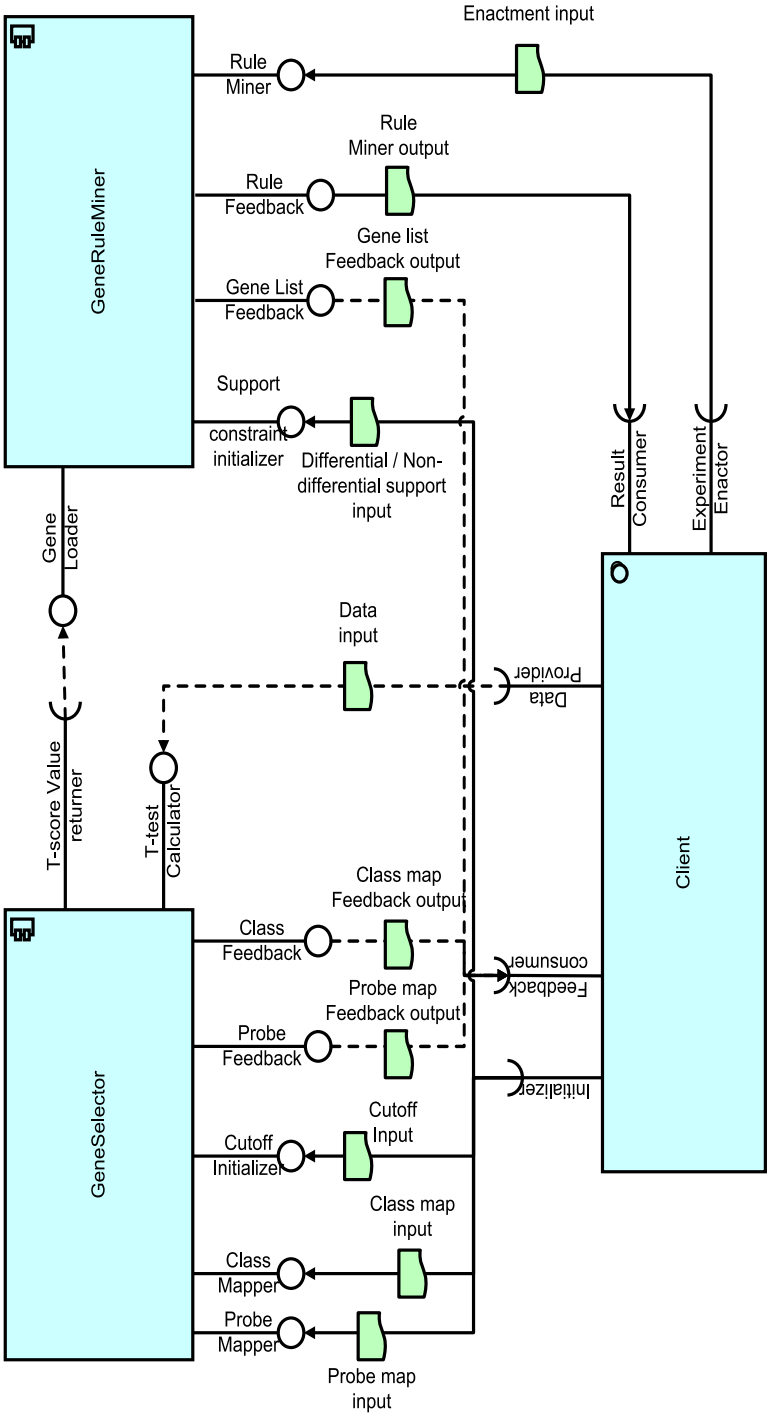


Figure 4.2: The SOKD use case design

4.5.4 Scientific Workflows

A number of workflow designer tools have been developed over the last few years, such as the orange toolkit [JMD+05] and Taverna [MyG08]. For our case study, we chose Taverna because it has the capability to execute web services. In Taverna, components are called processors, and apart from local services and WSDL services, Taverna also supports BioMoby [WL02] and SoapLab [KFH+06] web service interfaces. Connections in Taverna are pretty straightforward; data connections are called data links, and control connections are called control links. After a process has been designed and composed, the user can supply the input parameters of the process and execute it. When the process is done, Taverna will present the results to the user, or give an error message if something went wrong.

4.5.5 Implementation

Implementation of the original algorithms was done in the Python language and run on Python 2.5.2. The web service implementations were done in Microsoft C++ .Net 2005 and Microsoft C# 2005. All experiments were performed on Microsoft Windows XP using an Intel centrino duo processor 1.66GHz, and 1GB of main memory.

4.6 Experimental Results

4.6.1 KD Service Design

For our first use case, we compared three different implementations. To compare performance, we took the microarray data used in [GST+99] and reran the experiment that was done in [TZTL06], whereby Acute Lymphoblastic Leukemia (ALL) was contrasted against Acute Myelogenous Leukemia (AML). The results of the benchmark test can be seen in Table 4.4. All measurements are in seconds and are the averages of 50 consecutive runs.

	Selection service	Mining service
Original implementation	3.08s	99s
Shell implementation	3.20s	102s
Python implementation	3.23s	100s
C++ implementation	1.53s	66s

Table 4.4: Web service implementation benchmarks

Figure 4.3 shows the workflow we constructed using Taverna. To set up the workflow, the user loads the data files to the inputs created, and then runs the workflow. When successful, the workflow will display the returned file. Because all parameters

are combined in one SOAP message, we used input-splitters and output-splitters to join and split them.

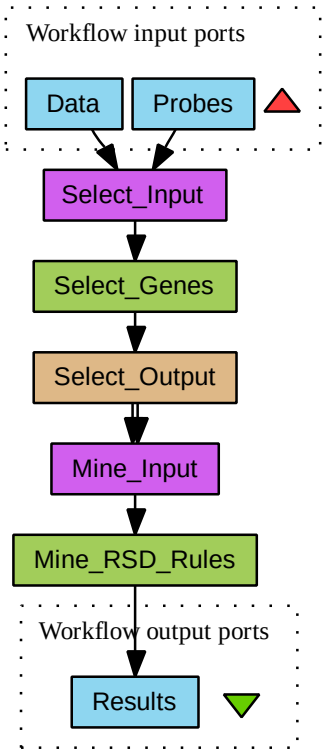


Figure 4.3: The Taverna ALL Vs. AML workflow

Based on the composition and execution of this workflow, we have a few observations and remarks on this workflow:

- **Monolithic sequential processing**
  Subprocesses that are in sequential order in Taverna need to finish first before the next subprocess can start; when sending data to a remote component, all the data is uploaded first before data mining can begin. While it is very intuitive to separate these processes completely, it can be faster to let the processing begin while the data transfer is still in progress. For example, our Selection service can already calculate and return t-values of individual genes while the upload is still in progress.

- **Stateless services**
  When executing our mining service, first all ontologies have to be loaded into memory, then annotation of the gene lists is performed, and finally the Relational Subgroup Discovery algorithm starts processing. Every time the web

service is executed, the same process is repeated. This is because this service is stateless, meaning that the web service has no knowledge of previous executions. We argue that if the service would preserve some form of state, execution could proceed faster. For example, if the mining service would keep the ontologies in memory and only change the gene assignments, the process would speedup considerably, being freed from most startup delays after the first run.

- **String representation**
  Message contents in SOAP are usually represented in text form. While this suffices for small messages, it is a redundant representation for large volumes of data. We argue that compression and decompression of data segments in the SOAP messages could improve the performance of a KD process .

Notice that these observations may also hold for the KD construction scenario, and that solutions to these problems require reimplementation of services and underlying algorithms, and possibly revisions or extensions to the SOAP protocol and Taverna messaging.

4.6.2 KD Process Design

The original process was divided into the same service partitioning as the one that our model yielded, but processing of individual services was done one by one, and results did not transfer before processing was completed. Furthermore, in the original implementation feedback was not supplied upon occurrence of an event, but as a return value after processing. A complete list of differences per service is presented in Table 4.5.

GeneSelector Service		
Category	Original process	Use case process
Service feedback	as return values	iterative on occurrence of event
Service initialization	supplied as a whole	iterative data supply
Service processing	all	per element
Service outputs	at end of processing	continuous outputting per element

GeneRuleMiner Service		
Category	Original process	Use case process
Service feedback	as return values	iterative on occurrence of event
Service initialization	supplied as a whole	iterative data supply
Service processing	all	all
Service outputs	at end of processing	at end of processing

Table 4.5: Design differences in GeneSelector and GeneRuleMiner services

Performance statistics of the original and the re-designed process are displayed in Table 4.6. As input for the selector we took t-score cutoffs of 15, 10 and 8. For the GeneRuleMiner, we took supports of at least 10% differential genes and at most 5% non-differential genes. The measurements of each phase in the table indicate the time (in miliseconds) since the entire *process* started up to the end of that phase, and are averaged over 50 consecutive runs. Since there was no performance increase to be gained in the GeneRuleMiner processing phase (all rules are returned at once when processing is finished), we only show the benchmarks of the phases preceding the GeneRuleMiner processing phase.

Original Process			
Phase	Cutoff 15	Cutoff 10	Cutoff 8
GeneSelector initialization	74	69	73
GeneSelector processing	1331	1291	1328
GeneRuleMiner initialization	3462	7122	13318

Re-desgined Process			
Phase	Cutoff 15	Cutoff 10	Cutoff 8
GeneSelector initialization	73	69	74
GeneSelector processing	1269	1228	1257
GeneRuleMiner initialization	2179	5841	11998

Table 4.6: Benchmarks of the original process and re-designed process

To make the difference between the original process and the re-designed use case more clear, consider Figure 4.4, which illustrates the scenario for cutoff 15.

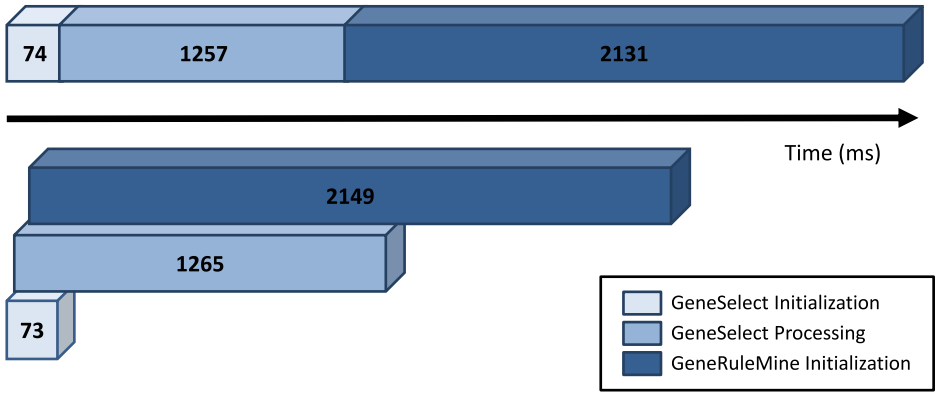


Figure 4.4: Benchmark comparison with a t-score threshold of 15

The top part displays how the original process parts are processed consecutively, whereas the bottom part shows how the re-designed process parts are processed iteratively, and in parallel where possible; data flows continuously from the GeneSelector component to the GeneRuleMiner component, thereby gaining a considerable performance increase.

4.7 Conclusions and Future Work

In this chapter we discussed the transition from construction of a KD process to the orchestration of a KD process through the use of SOA and workflows. We have contrasted two scenarios that indicate the weaknesses of process construction and execution on a single machine, weaknesses that were addressed in the second scenario by orchestrating processes in a SO fashion.

By using web services, versioning, connectivity, availability and performance of individual services are improved; versioning is improved by transparent updating of algorithms and the use of standardized WSDL interfaces, connectivity is improved by using the standardized SOAP transport protocol, availability is improved by better guarantees in service safety and the availability of the UDDI lookup service, and performance is improved by parallel execution and platform specific implementations.

Apart from the improvements on individual services, the design and performance of the entire KD process can be improved by using scientific workflows, which can be designed and executed with the Taverna workbench. By using workflows, design of KD processes is easier and more intuitive, since it splits KD processes in processing components and connections. A KD process designer just needs to import the components and connect them together in order to gain a valid KD process, which can then immediately be executed to gain a result.

By designing a SOKD process workflow using a design model that combines logical, functional and relational views, a better understanding of a KD process can be gained iteratively due to the matching and mismatching of entities in these views, whereby each iteration yields a better SOKD process design and a closer match of relations, services and functionality. An important factor that influences the partitioning of services are the services already available, thereby promoting software reuse.

When designing individual services in KD, interaction and feedback are important aspects to keep in mind. Interaction and regular feedback are important for the scientist to steer the KD process in a correct way, since KD processes are often time-consuming and thus any process incorrectly set up could possibly result in a considerable loss of time. Another important aspect is performance through parallelization. Since web services can be distributed across different logical or physical platforms, their execution could possibly proceed in a parallel fashion. To support parallelism, streaming data is preferred over monolithic data transport where possible. By combining these aspects and the functionality types in WSDL, we created guidelines for

the design of a KD service's functionality that is optimized for streaming data where possible and incorporates the need for feedback.

To illustrate the merits discussed above and to show how web services can be implemented, we created three different use cases. We showed how each use case contributed to the KD process, and showed how web services can yield a performance gain, sometimes cutting execution time by 50%. In case of redesign, processing times for the initialization of a process were further reduced up to 37%, and with 22% on average.

The design principles stated in this chapter are but a minor step to incorporating SO technology in the field of KD. However, by assuring that the design of a KD process and individual services is optimized for feedback and parallelism, a researcher can enact a process and conclude it successfully with minimal error and maximal performance. For further research we would need to study more use cases to ensure the research principles have maximum support in the KD scientific field. Furthermore, this design needs to be supported by graphical workflow tools that support iterative relationships instead of just monolithic data transport. Apart from SOAP extension, we also see future work in the combination of web services and databases. Our vision is to use web services to access databases to perform remote data mining and to construct queries.

Part II

Implementations

Chapter 5

The Fantom Subgroup Discovery Service

We propose a subgroup discovery service called Fantom that finds subgroups given a set of scored, ranked elements. The subgroups are described by conjunctions of ontological concepts and are given a measure of interestingness based on an idea from bio-informatics. For the generation of interesting subgroups we apply subgroup discovery by using frequent itemset mining, which exhaustively searches for all relevant subgroups above a minimal interestingness.

5.1 Introduction

Consider the following generic problem in data mining: as input we have a number of data elements with a score for each element. We want to find all subgroups that

- have a high score for the subgroup (i.e., a score for a set of elements based on the score of the individual elements in the set) indicating that this subgroup is interesting;
- can be described by a conjunction of predicates which all elements of the subgroup have in common.

Hence we need a procedure to score a set of elements. Furthermore, we need to decide which predicates can be used. A logical choice would be to use ontologies that describe (part of) the research domain. Moreover, we want to perform some pruning on the output, since predicates can imply other predicates, given the hierarchical structure of ontologies. We strive for most specific conjunctions of predicates as well as the highest scores.

We propose a service called *Fantom* that finds all subgroups described by a conjunction of predicates above a certain size and above a minimum score. It is built in

a generic way so that we can apply it in a variety of fields. As input, it uses a set of identifiers coupled to a set of scores and allowed predicates. As output, it presents the user with a set of rules and an appreciation of those rules in the form of a score measure. Fantom stands for Frequent pAtterN Tree-based Ontology Miner, a name that is self-explanatory: Fantom mines frequent patterns on the basis of ontologies.

This chapter is organized as follows. In Section 2, we will discuss work related to our Fantom approach, and discuss various knowledge sources that are used in Fantom as well. In Section 3, we will discuss Fantom itself, providing a detailed overview of inputs and outputs, pseudocode for parts of the algorithm and optimizations. In Section 4, we will present initial performance indications of Fantom, and provide statistics on our pruning strategies. Finally, in Section 5, we will make some preliminary conclusions, and discuss research and improvements that can be done in future work.

5.2 Related Work

In this section we present work related to the Fantom service, as well as work related to structured knowledge sources, knowledge mappings and other sources of information used in Fantom.

5.2.1 Ontologies

An *ontology*, as seen in information science, is the hierarchical structuring of knowledge about things by subcategorizing them according to their essential (or at least relevant and/or cognitive) qualities [Onl07]. Over time, many efforts have been made by the computer science community together with field experts to create and reason about ontologies for diverse scientific fields such as chemistry [OAM+03], web-mining and the semantic web [Dav06] and bio-informatics [ABB+00, OGS+99].

Due to the increased attention in data mining with ontologies, related technologies such as representations of ontologies, description logic and ontology reasoning have been given much attention as well. Currently, there is a wide range of (ontology) description languages available, and each of them has their own specific role. For representation of ontology elements and data, usually a form of the XML is applied, sometimes together with the Resource Description Framework (RDF) [W3C04a]. For representation of relations among the data elements and extensions to allow reasoning over these entity-relationship models, currently the Web Ontology Language (OWL) [W3C04b] and the older F-Logic [Bal95] are commonly used. A good overview of ontology languages is provided in [TS06].

Well-known ontologies in bio-informatics are GO and KEGG. GO, the Gene Ontology, aims to standardize the representation of gene and gene product attributes across species and databases. The three predicates that GO consists of are cellular component, biological process and molecular function. A gene product might be associated with or located in one or more cellular components; it is active in one or more

biological processes, during which it performs one or more molecular functions. A cellular component is a component of a cell, and part of some larger object, either an anatomical structure or a gene product group. A biological process is a series of events accomplished by performing one or more molecular functions in a specific order. A molecular function describes activities that occur at the molecular level.

KEGG, the Kyoto Encyclopedia of Genes and Genomes, aims to uncover higher-order systemic behaviors of the cell and the organism from genomic and molecular information. It is a database of biological systems, consisting of genetic building blocks of genes and proteins (KEGG GENES), chemical building blocks of both endogenous and exogenous substances (KEGG LIGAND), molecular wiring diagrams of interaction and reaction networks (KEGG PATHWAY), and hierarchies and relationships of various biological objects (KEGG BRITE).

5.2.2 Annotations and Mappings

Within Fantom we use ontological terms as rule predicates. This would not be possible if a mapping that associates or correlates an element with one or more ontology terms was not available. In the field of bio-informatics, there are many identifiers that can be used in genomics and proteomics [MOPT05, PTM07, MCOW05, WLDP02] and they are usually accompanied by mappings between those identifiers and ontologies, or those identifiers and other identifiers. For example, for GO terms there are several mappings (their default identifier is ENTREZ) that are updated either daily or monthly [Con09]. KEGG maps work with KEGG orthologies (genes in different species that are deriveed from a common ancestor), but also with HUGO gene symbols [Lab09].

Fantom provides an option to generate interaction association rules. By using a data source that states interactions between identifiers, Fantom can uncover patterns that describe these indirect relations. These interaction patterns have the form of ”interacts with(rule)”. In genomics, interactions can for example be obtained from the GeneRIF project [Gen09] and Reactome [VDS+07].

5.2.3 Related Algorithms

The Fantom algorithm is based on the SEGS algorithm described in [TLT07] and its predecessor [TZTL06]. While SEGS uses a similar method to Fantom, it is restricted to only one entry per (sub)ontology, is tailored specifically to microarray experiments, and does not prune rules that provide redundant information, nor does it provide clustering of similar rules. In [LRS+08] subgroups are matched to a subset of GO terms in a probabilistic way, which induces a greater portion of error and false discoveries than an exhaustive search through the search-space. The GOEAST [ZW08] algorithm also checks for gene enrichment in GO terms, but only checks for a single GO term, and takes as input raw microarray data, which again restricts its

applicability.

Another tool that mines gene lists is DAVID [HST+07], the Database for Annotation, Visualization and Integrated Discovery. Its functionality is similar to Fantom: it can classify large gene lists into functional related gene groups by relating them to ontologies (so far only the GO ontology was seen in the outputs), rank the importance of the discovered gene groups and summarize the major biology of the discovered gene groups. It also has capabilities to visualize genes and their functional annotations in a group.

The main difference is that DAVID does not allow for scored lists of genes. It solely acts on the genes that are entered in a list, and thus treats each gene as equally important. Furthermore, the number of genes allowed to experiment on is restricted to 3000, which is not much considering microarray experiments can easily comprise tens of thousands of genes. Furthermore, rule mining based on interaction associations is not available, and clusterering is done by using fuzzy heuristic partitioning instead of a similarity measurement. Finally, DAVID is not available as a web service but as a website, making it more complex to integrate in a workflow.

A lot of work has also been done on scoring functions. Typically, there is not just one scoring function that is considered the best, it all depends on what research is being conducted and what properties are considered interesting. In the case of Fantom, the aim is to have a score for a subset of elements from a ranking of identifiers and scores; the group score is thus dependent on the score of individual elements. In bio-informatics, well-known algorithms that perform this kind of scoring are found in [GST+99, LB06].

5.3 The Fantom Service

Fantom is a service that relates subgroups of identifiers to ontological knowledge that these subgroups have in common, or to ontological descriptions of identifier groups that they interact with. It takes as input a set of identifiers and their scores, background knowledge in the form of ontologies, mappings and interaction data, a scoring function, and thresholds for rule generation, and through rule generation and pruning delivers a non-redundant set of rules that describe subgroups of the input set. In this section we will discuss the inputs, internal mechanics and outputs of Fantom. For an overview of all the formats we refer the reader to Appendix A.

5.3.1 Inputs

In this section we describe the inputs of the Fantom algorithm. These include context parameters, a list of scored identifiers, user-defined ontologies, mappings between identifiers and ontologies and interaction data between identifiers, a scoring function, and functional thresholds and parameters.

Context Parameters

The context parameters are used to define the experimental context needed for the algorithm to function correctly. Based on these context parameters the correct versions of mappings and ontologies will be selected. For example, in our bio-informatics case studies, it would include the class of experimental input identifiers (Genes, or Single-Nucleotide Polymorfisms (SNP) identifiers), the type of the identifier (ENTREZ, SYMBOL) and the species this experiment concerns. Based on these parameters, translations will be made from one identifier type to another in both input and output, and correct mappings and ontology versions will be loaded, since ontologies can differ from species to species.

Set of Identifiers

The set of identifiers represents the elements in the experiment. All types of identifiers are allowed, as long as they can be related to some external source of knowledge, be it a set of uniquely identifiable car models, patients, or in our bio-informatics case studies, gene or SNP identifiers.

Each identifier in the set has a unique name, and each name has to correspond to an identifier in the provided mappings of the selected ontologies. If a mapping between identifier and ontology does not exist, it will be discarded. The same principle holds when mapping between identifier types.

An identifier is also accompanied by its score, or weight. There are no limits to these weights, but Fantom assumes that a higher weight means a higher importance, correlation, or effect. Scores can be negative as well, which is assumed to mean a negative correlation or effect.

Ontologies

The Fantom service aims to find groups of identifiers that relate to a conjunction of terms within an established knowledge base. We use ontologies as that knowledge base, since they reflect expert knowledge of a scientific (problem)domain. As defined earlier, an ontology is a hierarchical structuring of knowledge, whereby nodes annotated with broader, more general terms form the root of the ontology, and children of those nodes are annotated with specifications and differentiations of the parent term. We call these ontological terms *concepts*.

We place a restriction on the ontologies used in Fantom, namely that they must be organized in a directed acyclic graph, whereby each connection between concepts has a specialization (”is a”) or aggregation (”part of”) relationship. This restriction allows us to formalize the participation of identifiers in ontologies.

Given identifier i and concepts c, d, a collection of identifiers associated with a concept x named I_x, and a relationship $Parent(x, y)$ that denotes concept x as a parent of concept y, the following statement holds:

- $i \in I_c$ and $Parent(d, c) \rightarrow i \in I_d$.

Following from this we can also state:

- $Parent(d, c) \rightarrow I_c \subseteq I_d$, $|I_c| \leq |I_d|$.

Within Fantom, identifiers are associated with the concepts provided in the mapping, and all parents of those concepts. As an illustration, a part of the GO ontology is shown in Figure 5.1[1].

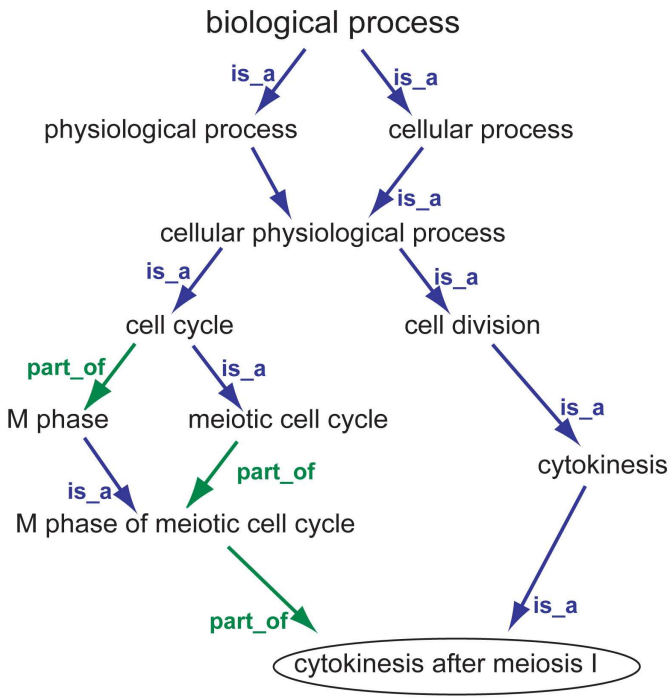


Figure 5.1: Part of the GO ontology structure

As can be seen in this example, concepts relate to each other through specific hierarchical relationships. Here the most general concept is *Biological Process*, which is the common root node for all biological processes described in GO. As we move lower in the graph, concepts become more specific.

[1]Image taken from http://www.yeastgenome.org/help/GO.html

Mappings and Interactions

Mappings and interactions are used to relate identifiers to associated concepts, or to relate identifiers to other identifiers. In the Fantom service we strive to keep one identifier class central, and use mappings to map the central class to other classes, and vice versa. For example, in case of our bio-informatics experiments, we keep mappings of ENTREZ gene identifiers to ontologies like GO and KEGG, and use mappings from SYMBOL to ENTREZ in order to translate the input ranking and output rules. The same is true for interaction definitions; all interacting genes are expressed by ENTREZ identifiers.

Scoring Functions

The scoring functions are those functions that take as input a subset of identifiers corresponding to a rule along with their individual scores, and have as output a single numeric value indicating the *interestingness* of the rule, a value that lies between 0 and 1, 0 being most uninteresting and 1 being most interesting. By default, the Enrichment Score (ES) function [GST+99] is selected, which calculates the score of a subgroup based on the score of its individual members as well as the members not in the subgroup:

- Sort the list of N identifiers according to their score with score function s, whereby $s_j = s(id_j)$. For the resulting list $L = (id_1, id_2, ..., id_N)$ it holds that $s_1 \geq s_2 \geq \ldots \geq s_N$.

- For each position i in L, evaluate the identifiers in the rule subgroup S, and the identifiers not in S but still present in L at a higher position than i:

$$P_{included}(S,i) = \sum_{\substack{j=1,...,i \\ id_j \in S}} |s_j| \,/ \sum_{\substack{k=1,...,N \\ id_k \in S}} |s_k|$$

$$P_{excluded}(S,i) = \sum_{\substack{j=1,...,i \\ id_j \notin S}} \frac{1}{N - |S|}$$

- The ES is the absolute maximum deviation from zero of $P_{included} - P_{excluded}$, where i varies from 1 to N.

Despite the fact that this measurement was devised to express the interestingness of gene sets, we think that it is in no way restricted to that purpose. We argue that any ranked set with a individual scores that indicate correlation or effect with respect to a certain experiment can be used in this scoring measurement.

Thresholds and Functional Parameters

Fantom allows the user to work with two thresholds: minimum support and minimum score. Minimum support indicates how much genes a subgroup should contain at least, and can heavily influence the duration of an experiment, depending on the data set used. The reason for introducing the minimum support threshold was to ensure that the single-identifier groups with the highest score do not dominate the list, since the default score function does not take support into account.

The second threshold indicates the minimum score a rule should have. It is primarily used for pruning rules in a postprocessing step, discarding rules that do not meet the specified interestingness criterion, but as we shall see later, it can also be used to optimize the rule generation process.

Apart from thresholds there are also a few functional parameters that specify the generality of the ontologies and the rules. The ontology parameter allows the user to influence the specificity of terms in a rule, while the rule parameter allows the user to constrict the number of concepts used in a rule to one per ontology predicate. Both parameters can positively influence the speed of the mining process at the expense of a smaller rule search space.

5.3.2 Output

As output, Fantom generates a text file that contains all the rules that remain after pruning. Furthermore, rules with the same subset of identifiers are clustered together, improving readability. An example rule looks like this:

```
Rule 1
Score: 0,761302007553004
Participants: [Epha1, Epha2, Ephb2, Ephb3, Ephb4, Ptk2]

All genes in the subgroup
 have the following properties:
  molecular_function(protein tyrosine kinase activity),
  molecular_function(ATP binding),
  biological_process(protein amino acid phosphorylation),
  biological_process(tyrosine kinase signaling pathway),
  KEGG_pathway(Axon guidance)
```

The example rule describes a certain subgroup containing the genes *Epha1*, *Epha2*, *Ephb2*, *Ephb3*, *Ephb4*, *Ptk2* and relates them to GO and KEGG terms, a process we call Knowledge Fitting. By relating subgroups of the input to established knowledge sources, we strive to increase the interpretability of knowledge.

The Fantom method can also generate rules that take into account gene interactions. When interaction association rules are generated, the conjunctions of ontology

terms do not describe the subgroup mentioned in the participants header, but rather an anonymous subgroup that all the subgroup participants interact with. The next rule would be an example of such an interaction rule:

```
Rule 1
Score: 0,915794797276831
Participants: [Acat1, Acat2, Cycs, Dld, Mdh2, Uqcrq]

All genes in the subgroup have interaction with genes
 that have the following properties:
  molecular_function(ATP binding),
  biological_process(tricarboxylic acid cycle),
  KEGG_pathway(Alzheimer's disease)
```

5.3.3 Algorithms and Structures

The Fantom service consists of several methods, whereby three phases are repetitively executed until all options are exhausted: generation, appreciation and pruning of rule candidates. The pseudocode is shown in Algorithm 1.

Algorithm 1 Fantom Main Body

```
RuleCandidates C ← ∅
SubsetCollection S ← ∅
C ← Preprocessing()
InsertCandidates(S, C)
while C ≠ ∅ do
    C ← GenerateCandidates(S)
    C ← AppreciateCandidates(C)
    InsertCandidates(S, C)
    PruneCandidates(S)
    C ← ReturnCandidates(S)
end while
C ← ReturnRules(S)
print Postprocessing(C)
```

In the remainder of this subsection all methods will be discussed individually.

Preprocessing

In the preprocessing phase all inputs are loaded. First the ranked list of identifiers is loaded, and translated to the default identifier if necessary. Next, the selected ontologies are loaded into the system. These ontologies are represented in a dictionary and indexed on their unique identifier. Each ontology predicate gets its own dictionary

to avoid the case of having equal index names between ontologies and to optimize memory usage and search-time, which is $O(\log(n))$ on average, and $O(n)$ in the worst case, where n is the amount of items in the dictionary. In the final phase of the loading stage, the mappings from identifiers to ontology concepts are loaded.

After all mappings and ontologies are loaded, the ontology dictionaries are annotated by the identifiers. There are two ways of annotating the ontologies, depending whether interaction rules need to be generated or not. If the experiment is set to generate interaction rules, then the interaction file will be used as an intermediate step to annotate the ontologies. If not, then solely the map from identifiers to concepts is used. The complete algorithm pseudocode is shown in Algorithm 2.

Algorithm 2 *AnnotateOntology(o, i, m, L)*

Require: An ontology o.
Require: An identifier list i.
Require: An identifier to ontology mapping m.
Require: (optional) An interaction list L.

```
for all id ∈ i do
  if L ≠ ∅ then
    indirect_set ← GetInteractions(id, L)
    concept_set ← ∅
    for all indirect_id ∈ indirect_set do
      concept_set ← concept_set ∪ GetConcepts(indirect_id, o, m)
    end for
  else
    concept_set ← GetConcepts(id, o, m)
  end if
  for all concept ∈ concept_set do
    AnnotateConceptAndParents(concept, id)
  end for
end for
```

Algorithm 2 works as follows. If an interaction list was supplied, the algorithm fetches all identifiers that *id* interacts with, and fetches the concepts associated with those identifiers. Then, each of those concepts plus their parents are annotated with identifier *id*. If there are no interactions specified, the concepts directly related to *id* as well as their parents are annoteded with *id*.

After all ontologies have been annotated, the preprocessing phase moves into its final stage: preparation for candidate generation. In this stage, all ontology concepts that participate in this experiment are compared to the minimum support and score thresholds, and those with sufficient support will be used for further generation of rules. At the same time, the ontologies will also be cropped and optimized for use in

the candidate pruning stage. The ontologies are pruned and optimized in a number of ways:

- **Removal of unused concepts**
  After annotation of the ontologies, all concepts that remain unannotated with identifiers will not participate in any further actions taken by the Fantom service and thus can be unloaded from the ontology.

- **Cropping of uninformative concepts**
  When using all ontology concepts, chances are that some of the rules generated are not useful because they contain concepts that are too general to be informative. To counter uninformative rules, Fantom allows the user to provide a functional parameter that indicates the specificity of the terms to be used in an ontology. This parameter indicates a percentage of the maximum depth of the pruned ontology, whereby depth is the maximum number of parents a concept has to the root of the ontology, including the root itself. For example, if the maximum depth of an ontology in a specific experiment is 7, and the user supplied 0.5 as a parameter, then the minimum depth for a concept to be allowed to be included in the experiment is 4.

Candidate Generation

Candidate generation in the Fantom service is based on the Apriori algorithm, which is frequently used in itemset mining [AIS93] and refined many times since its conception [HPY00, Bod03]. Given that it is a proven technique that has been used in many fields, it was considered generic enough to be used in Fantom. Within the Fantom service, the algorithm repeatedly executes three stages: candidate generation, candidate appreciation and candidate pruning. A central structure that manages rules in all these procedures is the *SubsetCollection* structure, which provides methods for inserting, retrieving and discarding rules and candidates. Note that until it is reported as output, no rule is fixed yet, and therefore we shall refer to them as *rule candidates*, or merely *candidates*.

A rule candidate is a data structure with two lists; one list contains the participants in the candidate, i. e., the identifiers that are associated with all the concepts, which are present in the second list. Candidate generation proceeds on the basis of subset combination; two candidates that contain a conjunction of m concepts are combined with each other to form a candidate that has a conjunction of $m + 1$ concepts. When combining two candidates, the resulting identifier lists will be the conjunction of identifiers that were in the two lists of the original candidates.

The algorithm works as follows. Before each cycle, all candidates are inserted in the SubsetCollection structure. In case of the first cycle, when all candidates contain only one concept (these are called the *atomic candidates*), all candidates are gathered in one list. However, assume now that we are in a certain cycle m, which contains

candidates with m concepts. For those candidates to be combined into a candidate of $m+1$ elements, they have to share $m-1$ elements. Thus for every candidate of $m > 1$ concepts, m subgroups are created and the candidate is inserted m times into a hash table, each entry hashed by a different subgroup. Notice that each cycle has its own hash table except for the first one, which has no subsets.

After all candidates of length m are inserted, generation of candidates of length $m+1$ commences. If there is no hash table present, that means that there is a single list of atomic candidates. If this list contains n candidates, then the maximum number of new candidates is $n * (n-1)/2$. However, if the two concepts that are combined are hierarchically related, then the combination is invalidated, and the candidate is discarded.

Now consider the case where we have candidates of $m > 1$ concepts. Suppose that there are k hashes, the i-th hash containing S_i candidates, $1 \leq i \leq k$. Then the number of generated candidates would be at most:

$$\sum_{i=1}^{k} \frac{S_i * (S_i - 1)}{2}$$

Combination of two candidates within a hash entry proceeds in a straightforward fashion. The concepts that do not belong to their common subgroup of $m-1$ concepts are tested for hierarchical relatedness, and discarded if there is such a relation. A complete overview of this part of the algorithm is shown in Algorithm 3.
After all combinations have been generated, they are returned by the function and passed on to the candidate appreciation phase.

Candidate Appreciation

The candidate appreciation phase is the phase where all generated candidates in the previous phase are being assigned a score on the basis of the selected score function. Two scores are being calculated: the score of the candidate with its current set of participating identifiers, and the maximum score that the minimum number of elements involved in this rule could possibly have, whereby this minimum is equal to the minimum support parameter. If this maximum score is below the score threshold, that means that this rule can be pruned in the next phase.

In Fantom, the ES measurement is used as a default scoring mechanism. ES gives high scores to groups that contain many identifiers at the top of the identifier ranking (high positive correlations or effect) or at the bottom (high negative correlations or effect). As an illustration, consider Figure 5.2, which shows the score calculation result after each sample, and was generated in one of our case studies. As can be seen, the score increases fast at the beginning, indicating that a lot of genes with high positive effect were participants in the rule under consideration. If participants were less concentrated on either side, but instead spread out over the entire ranking, a smaller maximum score would be obtained.

The ES is an indication of the interestingness of the current candidate based on

Algorithm 3 *GenerateCandidates(S)*

Require: SubsetCollection Structure S.

```
NewCandidates ← ∅
if Subsets(S) ← ∅ then
  C ← GetAllCandidates(S)
  for i ∈ {1, ..., |C| − 1} do
    for j ∈ {i + 1, ..., |C|} do
      if not Related(Concepts(C[i]), Concepts(C[j])) then
        NewCandidates ← NewCandidates ∪ MakeCandidate(C[i], C[j])
      end if
    end for
  end for
else
  Hashtable H ← GetLatestCandidatesSubsets(S)
  for all Key ∈ Keys(H) do
    C ← H[Key]
    for i ∈ {1, ..., |C| − 1} do
      for j ∈ {i + 1, ..., |C|} do
        if not Related(Concepts(C[i]), Concepts(C[j])) then
          NewCandidates ← NewCandidates ∪ MakeCandidate(C[i], C[j])
        end if
      end for
    end for
  end for
end if
return NewCandidates
```

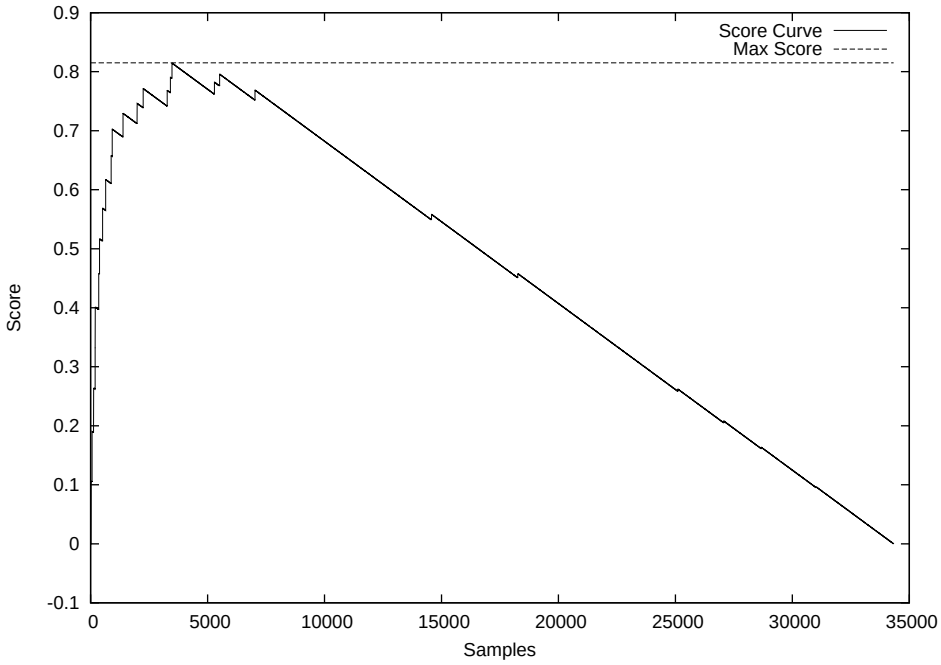


Figure 5.2: Example of ES determination

its identifier subgroup M. We cannot, however, use this ES to prune if it is lower than the given score threshold t, for combination of this candidate with another one could still yield an interesting new candidate with a score above t. However, we can find $L \subseteq M$, the subgroup of M for which the ES is maximal, whereby $|L|$ is the minimum support of the rule. We call this score ES_{max}. The number of subgroups to consider would be $\binom{|M|}{|L|}$, which is an infeasible amount to process for each candidate for large subgroups. However, it can be done in $O(|M|)$ by using the following backwards induction algorithm devised by Muskulus & de Bruin, which is presented in Algorithm 4.

A brief summary of the algorithm is: for each identifier position i in M we assume that ES_{max} is achieved there. For a given minimum threshold q, we delete the $|M| - q$ participants with the biggest absolute score after the position of i in the ranking. In case there are $|M| - q$ values or more, we are done and we recalculate the score of this set. In case there are less than $|M| - q$ values to remove, then we delete the remaining values that are directly before i. Since ES is the maximum deviation from zero, we also need to take into account the occasion that the minimum ES is at position i. However, to calculate this, we invert the order of all identifiers in their ranking as well as their scores, and again calculate ES_{max}. The maximum score of the candidate is the maximum over all $2|M|$ values calculated. For a com-

Algorithm 4 *MaxEnrichmentScore(M, q)*

Require: Identifier Set M.
Require: Minimum Support q.

for $i \in \{1, \ldots, |M|\}$ **do**
  $T \leftarrow \{1, \ldots, |M|\} \setminus \{i\}$
  for $j \in \{1, \ldots, (|M| - q)\}$ **do**
    $T_+ = \{t \in T \mid t > i\}$
    if $T_+ \neq \emptyset$ **then**
      $t_0 \leftarrow \min_{t \in T_+} t$
    else
      $T_- = \{t \in T \mid t < i\}$
      $t_0 \leftarrow \max_{t \in T_-} t$
    end if
    $T \leftarrow T \setminus \{t_0\}$
  end for
  $M' \leftarrow \{M[t] \mid t \in T\} \cup \{M[i]\}$
  $c_i \leftarrow$ *CalcEnrichmentScore(*M'*)*
end for
return $\max_{1 \leq i \leq |M|} c_i$

plete overview of the algorithm and its correctness we refer the reader to Appendix B.

Candidate Pruning

Candidate pruning is done to make sure only the most interesting rules are reported back once the experiment is over, to make sure redundant information is left out, but also to minimize the amount of candidates generated in the experiment. Within each stage, candidate pruning is done in three phases. First, candidates generated and appreciated in the previous two phases are now compared to the input constraints; all candidates whose number of participants do not satisfy the support constraint, or whose ES_{max} is below the minimum score constraint, are pruned. The remainder will be inserted in the subset collection structure, where further pruning will take place.

Within the subset collection structure, pruning proceeds in a bi-dimensional way, hence we called it *Bi-Cohortal Pruning*. First we prune the rules in a *horizontal* cohort, comparing candidates of the same dimensionality (by dimensionality we mean the number of concepts that an candidate contains). The remaining candidates are inserted one by one. The algorithm is shown in Algorithm 5.

As discussed earlier, for every candidate of $m > 1$ concepts, m subgroups are created and the candidate is inserted m times into a hash table. Upon each insertion in

Algorithm 5 *InsertCandidates(S, C)*

Require: SubsetCollection structure S.
Require: Set of candidates C.

```
for all c ∈ C do
  T ← GenerateSubsets(c, |Concepts(c)| - 1)
  if T = ∅ then
    C' ← GetAllCandidates(S)
    for all p ∈ C' do
      if Descendant(Concepts(p), Concepts(c)) and Score(p) ≥ Score(c) then
        InvalidateCandidate(c)
      end if
    end for
    InsertCandidate(S, c)
  else
    for all t ∈ T do
      C' ← GetCandidatesFromSubgroup(S, t)
      for all p ∈ C' do
        if Descendant(Concepts(p), Concepts(c)) and Score(p) ≥ Score(c)
        then
          InvalidateCandidate(c)
        end if
      end for
    end for
    InsertCandidateInSubgroups(S, c, C')
  end if
end for
```

a hash table entry, the candidate is compared to other candidates of the same dimensionality for redundant knowledge, checking whether their differentiating concepts (the two concepts these candidates do not have in common) are related. If this is the case, and the score of the more specific candidate is higher than or equal to the score of the more general one, then the more general candidate is discarded as a rule (but is still kept for future candidate generation, and inserted in all the subgroups it belongs to). In case there are no subgroups yet ($m = 1$), the candidate is compared to all the candidates previously inserted into the Subgroup structure, and then inserted itself. Note that since it does not belong to any subgroup, it will be inserted into a general list of $m = 1$ candidates.

After pruning of the horizontal cohort has finished, pruning of the *longitudinal* cohort commences. The remainder of the still valid candidates of dimensionality m are compared to those of dimensionality $1...m-1$, since rules of dimensionality m

are more specific than rules of a lesser dimensionality, and can thus render those rules obsolete. To this end, all subsets of dimensionality $1...m-1$ are generated from each remaining candidate of dimensionality m, and these are subgroups are compared to in the subset collection by comparing them to related entries in the castable of their respective dimensionality. This algorithm is shown in Algorithm 6.

Algorithm 6 *PruneCandidates(S)*

Require: SubsetCollection structure S.

```
C ← GetMostRecentCandidates()
for all c ∈ C do
  for i ∈ {1, ..., |Concepts(c)| − 1} do
    T ← GenerateSubsets(c, i)
    if T = ∅ then
      continue
    else
      for all t ∈ T do
        if i = 1 then
          C′ ← GetCandidatesOfDimension(S, 1)
          for all p ∈ C′ do
            if Descendant(Concepts(t),Concepts(p)) and
            Score(c) ≥ Score(p) then
              InvalidateCandidate(p)
            end if
          end for
        else
          T′ ← GenerateSubsets(t, i − 1)
          for all t′ ∈ T′ do
            C′ ← GetCandidatesFromSubgroup(S, t′)
            for all p ∈ C′ do
              if Descendant(Concepts(t),Concepts(p)) and
              Score(c) ≥ Score(p) then
                InvalidateCandidate(p)
              end if
            end for
          end for
        end if
      end for
    end if
  end for
end for
```

Postprocessing

In the postprocessing phase, all invalidated options are discarded, leaving only those options that are now rules. These rules are then clustered together by participants, since the same subgroups can have different rules associated with them. Furthermore, the list of rules is sorted and formatted according to the output format discussed earlier.

When clustering, the user has to specify a parameter between 0 and 1, the similarity threshold, indicating how similar rules need to be in order to be clustered together. Similarity score S for rules R_1 and R_2 is calculated by dividing the conjunction of both their identifier lists (rule participants) between the minimum of the two, as can be seen below:

$$\mathrm{S} = \frac{|Participants(R_1) \cap Participants(R_2)|}{min(|Participants(R_1)|, |Participants(R_2)|)}$$

The full algorithm is shown in Algorithm 7.

Algorithm 7 *ClusterRules(R, q)*

Require: Set of Rules R.
Require: Similarity Threshold q.

FinalClusters $\leftarrow \emptyset$
AssociationsMap $\leftarrow$ *BuildAssociations(R, q)*
Clusters $\leftarrow$ *BuildClusters(AssociationsMap)*
FinalClusters $\leftarrow$ *ReduceRedundancy(Clusters)*
return *FinalClusters*

In Algorithm 7, the rules are first divided into associations in the function *BuildAssociations*. In this function, a hash table is built with rules as keys, and for each key in R, the entry is a list of all rules that have a similarity score higher than or equal to threshold q.

In the next step, clusters are built from these conjunctions using the function *BuildClusters*, which is a function that builds clusters recursively from the association maps. The function *BuildClusters* is displayed in Algorithm 8. It is a recursive function that builds all possible maximal cliques for each rule in the association map *AssocMap*.

Finally, after all clusters have been built, they are pruned for redundancy in the function *ReduceRedundancy*. This function makes sure that each rules only appears in the cluster with the most members, though some redundancy is allowed if a rule appears in multiple clusters with the same number of members.

Clustering in *BuildClusters* proceeds as follows. First, for each of the entries in the *AssocMap* hash table it is checked if it is a clique. A clique is found when all rules within the entry have a similarity score higher than the threshold t, not only

Algorithm 8 *BuildClusters(AssocMap)*

Require: Map of Associations *AssocMap*.

```
ClusterSet ← ∅
for all key ∈ Keys(AssocMap) do
    SafeSet[key] ← key
    StrifeSet ← ∅
    UnhandledSet ← AssocMap[key]
    for all u ∈ UnhandledSet do
        strife ← false
        for all r ∈ UnhandledSet do
            if u = r then
                continue
            else if r ∉ AssocMap[u] then
                strife ← true
                StrifeSet ← StrifeSet ∪ {u}
            end if
        end for
        if strife = false then
            SafeSet[key] ← SafeSet[key] ∪ {u}
        end if
    end for
    if StrifeSet = ∅ then
        ClusterSet ← ClusterSet ∪ SafeSet[key]
    else
        FriendSet ← BuildFriendSet(StrifeSet, AssocMap)
        SubSet ← BuildClusters(FriendSet)
        for all s ∈ Subset do
            ClusterSet ← ClusterSet ∪ SafeSet[key] ∪ s
        end for
    end if
end for
return ClusterSet
```

with respect to the key rule, but also with each other. If this is the case, the cluster is added to *ClusterSet*.

If the set is not a clique, it means that at least two elements in the set do not have a similarity score that is sufficiently high, and thus they are put into *StrifeSet* instead of *SafeSet*. What happens next is that for *StrifeSet* a hash table is constructed like *AssocMap*, called *FriendSet*, which contains the entries of *AssocMap* for each of the keys in *StrifeSet*. Next, *BuildClusters* is recursively invoked, and all the clusters resulting from that invocation are merged with the clique represented in *SafeSet*. Finally, these are added as clusters into *ClusterSet*.

5.4 Initial Performance Experiments

In this section we give an indication on overall performance of the Fantom service. We apply Fantom to one of the microarray data sets used in [GST+99]. We discuss the data set used, what transformations we applied to get a ranking of genes, and what parameters we supplied to Fantom. We also present some performance statistics that address both speed and pruning.

Dataset and Inputs

In this use case we used a well-known publicly available dataset that compares gene expression profiles of AML and ALL [ASS+02]. In this data set, gene expression profiles were taken from 47 patients suffering from ALL and 25 patients suffering from AML. We first normalized the raw data using Quantile normalization [BIAS03]. After that, we performed mapping of the probes to ENTREZ genes using the Hu6800 annotations supplied by Affymetrix. We discarded any entries that could not be mapped successfully to a single identifier, to reduce the uncertainty error. Finally, we performed a t-value calculation with the Student's t-test between those two groups, thus researching what all over-expressed genes have in common. If a gene had multiple t-values, the average of those values was taken.

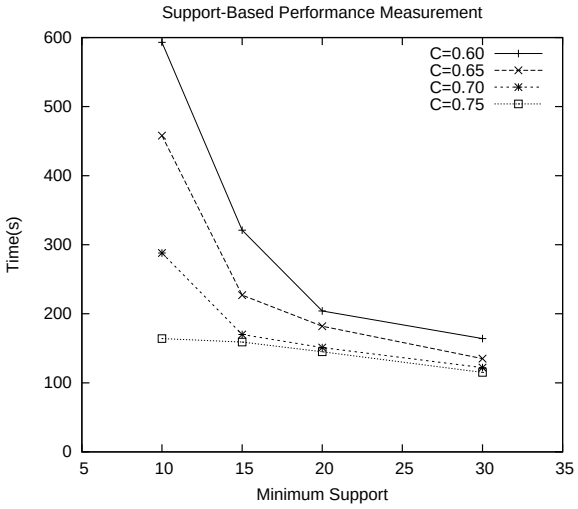
As ontology inputs the GO and KEGG ontologies were used, combined with the ES score metric. Context inputs were set to homo sapiens, and the identifier was kept default to ENTREZ.

Implementation

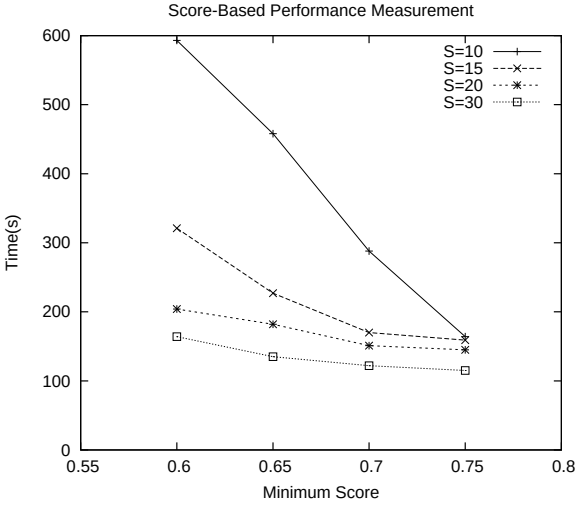
Implementation of the ontology and mapping generation as well as the interaction files was done in the Python language and run on Python 2.5.2, because of the efficient and easy string handling of the Python scripting language. The web service implementations were done in Microsoft C++ .Net 2008 and Microsoft C# 2008, both using the .Net Framework version 2.0 and 3.5. We embedded this web service in a workflow created in the Taverna [MyG08] workbench.

Performance

In Figure 5.3, performance measurements are shown for different minimal support sizes S and different minimum score thresholds C. In Figure 5.3(a), the horizontal axis indicates the minimum support in items, while the verical axis indicates the running time of an experiment. In Figure 5.3(b), the horizontal axis indicates the minimum score setting in an experiment. The vertical axis is once again the running time of an experiment.



(a)



(b)

Figure 5.3: The Fantom service performance measurements

As can be seen in Figure 5.3(a), the increase of the minimum participants has a profound effect on the performance of the algorithm. The same effect can be seen in Figure 5.3(b) where the minimum score was increased, though at a lesser extend with bigger subgroups. Still, if we extrapolate the lines in Figure 5.3(b), it is still obvious that pruning based on the ES measurement improves performance greatly (one can simulate the lack of ES pruning by taking a minimum score of 0).

Another interesting question is how these two thresholds affect pruning. Intuitively, smaller subgroups and lower minimum scores result in more rules being generated, and therefore more rules pruned, but if we examine the percentages of rules pruned we see that with both thresholds it is fairly stable around 99.8%. These results are shown in Figure 5.4.In Figure 5.4(a), the horizontal axis indicates the minimum support in items, while the verical axis indicates the percentage of rules pruned. In Figure 5.4(b), the horizontal axis indicates the minimum score setting in an experiment. The vertical axis is once again the percentage of rules pruned.

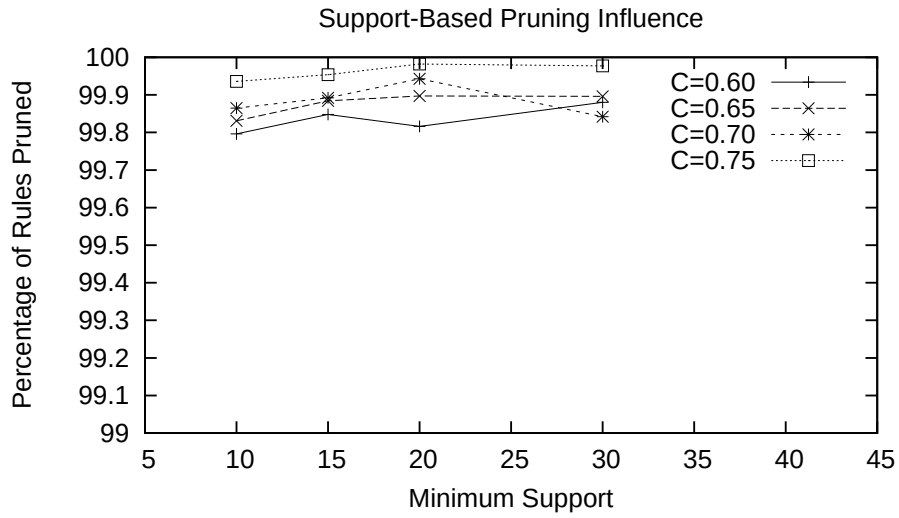
As can be seen, the pruning algorithm is slightly more erratic in the support threshold seen in Figure 5.4(a) than in the score threshold shown in Figure 5.4(b), but overall both are monotonically increasing.

5.5 Conclusions and Future Work

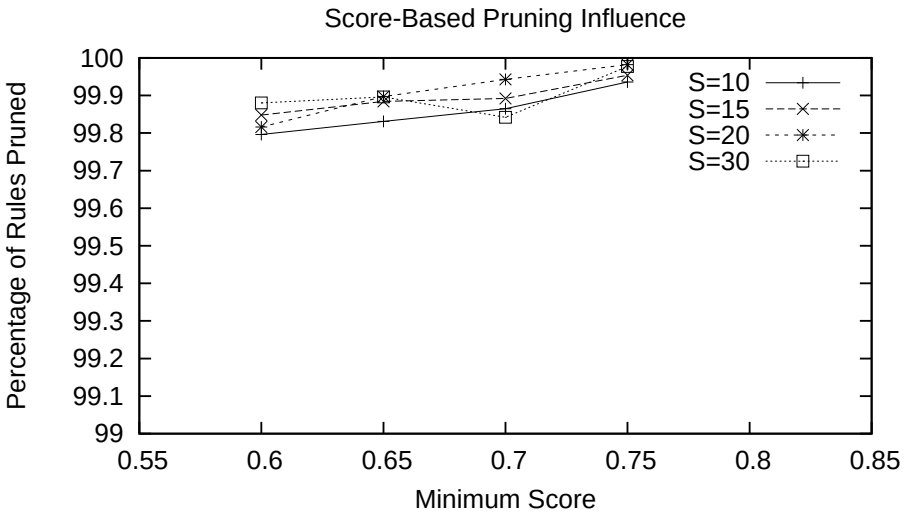
In this chapter we discussed a subgroup discovery service called Fantom that finds subgroups given a set of weighed elements. We explained the technologies behind the algorithm, its data sources, and its way of combining that data to generate comprehensive patterns that are tailored to the expert knowledge of the researcher.

In our experiments, we have shown several statistics on the Golub et al. data set, which we normalized and then extracted the participating genes and their scores from it. We have shown that pruning can be done with both a monotonic constraint such as support, but also by adapting a non-monotonic constraint such as the ES score measurement, thereby making use of the minimum support threshold. This resulted in the generation of less rules and increased pruning, which rendered at least 99.85% of all the rules generated useless, greatly diminishing redundant information.

For future work, efforts have to be made to increase rule statistics, not only with ES scores, but also p-values for confidence. Furthermore, some other score functions than just ES should be evaluated and supported. A wide overview is presented in [AS09]. Of course, a qualitative re-assessment of the rules with different score measures will have to be made, as well as research into the tradeoff between performance and quality.



(a)



(b)

Figure 5.4: The Fantom service pruning measurements

Chapter 6

The Fantom Service: Exact Testing

In this chapter we describe how we combine the Fantom service with the statistical principle of permutation testing. We demonstrate that by performing several iterations of the Fantom service on permutations of an identifier list, we can prune the rule set for the original list even further. We also demonstrate that by combining Fantom and permutation testing, we can determine an optimal support threshold for classes in a multi-class experiment with respect to interestingness of rules.

6.1 Introduction

When generating rules from a ranked list of identifiers, the rules usually reflect the ranking, and Fantom is no different in this respect. It is therefore good practise, where possible, to make sure that the ranking is correct, and to make sure that rules generated and reported are specific to that ranking, and not a product of randomness or chance, which can sometimes occur. By generating permutations of a ranked list, one can check if these permutations generate similar rules. If so, then the rules become less important and interesting, for they are not specific to the original ranking.

The method previously described is a variation on Fisher's Exact Test [Fis22, Fis67]. Fisher's Exact Test is a statistical significance test that uses permutation generation to determine the deviation from a null hypothesis. It is called an *exact* test because it does not rely on heuristics and approximations of the deviation, but can calculate it exactly through generation of all possible permutations. Usually, a *p-value* is calculated, which is the probability of obtaining a measurement that has at least (or at most) the same value as the value actually observed, assuming that the null hypothesis is true. The lower the p-value, the less likely the result is obtained by chance, assuming the null hypothesis is true, which makes the result more significant.

Consider a simple example: suppose we have a ranking of identifiers, and we create 99 permutations of those identifiers. After rule generation, we check for each rule if it also appears in the permutation experiments. Suppose a rule in the original ex-

periment appeared in n out of the residual 99 experiments, then a simplified p-value could be $\frac{n+1}{100}$. Usually an observation (or rule) is considered interesting if its p-value is below 0.05, or even 0.01. The rule is then said to be *statistically significant*

In this chapter we will apply this concept to the rules generated in Fantom. We generated permutations of the input, and then generated rules from each of those permutations using multiple dedicated instances of the Fantom service. Furthermore, we used the same principle to generate an automated score threshold for a multi-class problems in Fantom by using maximized rule count differentials.

This chapter is organized as follows. In Section 2, we discuss exact testing with Fantom on a single-class problem, generating a rule list with rules unique to the original permutation. We also discuss exact testing with Fantom for multi-class problems, whereby different groups of identifiers are compared to each other. We explain the difference with the normal experimental setup, and the algorithm behind automatic thresholding of different groups participating in the rules. In Section 3, we present experimental results on both variations using the AML versus ALL dataset again. Finally, in Section 4 we present some conclusions and future work.

6.2 Exact Testing for Pruning and Optimization

In this section we discuss two versions of exact testing with Fantom: one for single-class problems, which is used for pruning of the rules, and one for multi-class problems, which can be used for both rule pruning and accurate threshold determination.

6.2.1 Exact Testing: Single-Class Pruning

In this variant of Fantom, we use the Fantom algorithm repeatedly to generate rules from multiple identifier lists. One list is the original ranking, which reflects the original experiment data, and the other lists are derived from permutations of that data or ranking. It is up to the experimenter to create those permutations, although we created a permutation algorithm for microarray expression data.

Exact testing for single-class pruning is a three-stage process, as shown in Figure 6.1. In the first stage, input permutations are generated. The output of this phase, a set of ranked lists, will serve as input of the second stage, along with the original ranked list.

The second stage is the concurrent execution of the Fantom service on all ranked lists generated in phase one. Depending on how many permutations have been generated, all or a portion of the permutations are processed on the Fantom services available, until all permutations have been processed. The resulting rules are then forwarded to the final phase.

In the third and final phase, the output is gathered and combined. First, the rules generated on the original input, from here on called the *original rule set*, are loaded

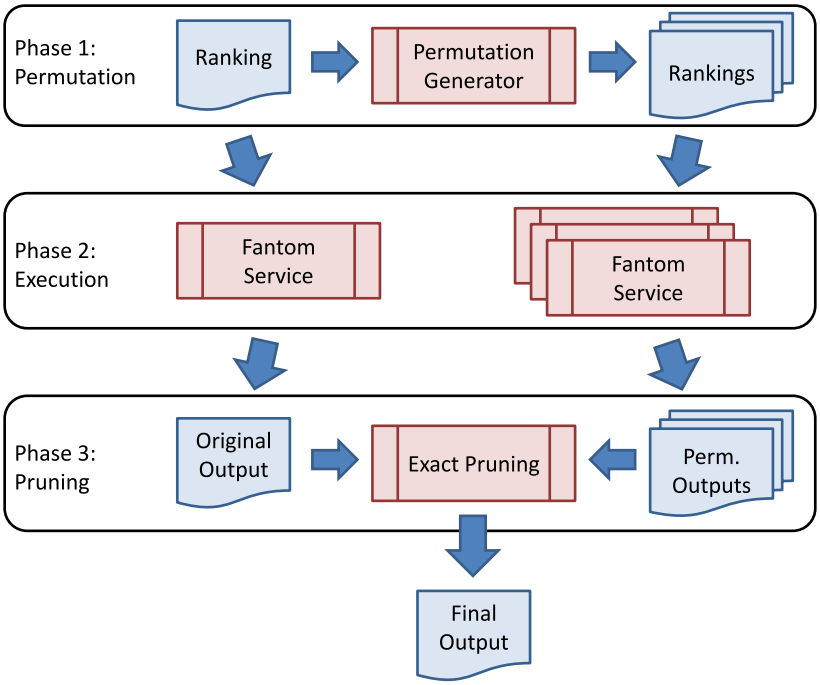


Figure 6.1: Workflow for exact pruning

into the system. After that, the rules generated on the permutations are loaded and compared to the original rule set. If there are any rules more specific than or equal to rules in the original rule set, and with a higher or equal score, then the related rules in the original rule set are pruned. Comparison and pruning of rules proceeds according to the algorithms described in Algorithm 5 and Algorithm 6 from Chapter 5.

6.2.2 Exact Testing: Multi-Class Threshold Optimization

Fantom can also be used to perform multi-class comparison experiments. The goal of these kinds of experiments is to investigate commonalities in subgroups of the identifiers belonging to the *interest class*. The rest of the identifiers belonging to the *control class* serve as contrast information, and their involvement in rules penalize the score of the rule. Consequentially, subgroup rules that contain many identifiers in the interest class and few in the control class gain a higher score.

As an example, let us consider a wine comparison problem. Suppose we have a list of wines, all with unique identifiers, and we want to investigate what the 10 most popular wines have in common according to a certain wine ontology, for example the one described in the OWL documentation [W3C04b]. We would label the popular wines with $ClassInterest$, and the rest with $ClassControl$. When running Fan-

tom, the service will strive to find all the rules that describe subgroups of the interest class, considering the control class purely as contrast information used in the score measurement of the rules. Experiments of these kind are usually shorter and faster in Fantom, since the support threshold only applies to the interest class.

Fantom exact testing for multi-class problems has a slightly different implementation than Fantom for single-class problems. Each ontology concept now has two lists associated to it, one list for the identifiers of the interest class, and one list for the control class. A similar principle holds for the representation of options, which now have two lists of participating identifiers, one for each class.

The scoring is also different, since the standard ES measurement is not suited for multi-class problems. We therefore adapted the ES measurement to be suited for multi-class problems. Let the interest list of an option be $L_{Interest}$, and the control list be $L_{Control}$. Furthermore, if we perceive the two classes as two different rankings, namely $InterestRank$ and $ControlRank$, then for each option $ES_{Interest}$ would be the ES of the identifiers of the interest class, calculated over $InterestRank$. However, we still have to penalize rules for their control set participation. As a penalty, the weighed percentage of control identifiers will be subtracted from $ES_{Interest}$. If we take score function s, the modified $ES_{MultiClass}$ is thus:

$$ES_{MultiClass} = ES_{Interest} - \frac{\sum_{id \in L_{Control}} s(id)}{\sum_{id \in ControlRank} s(id)}$$

As can be seen, maximization of $ES_{MultiClass}$ is a matter of maximizing $ES_{Interest}$, while (optimistically) assuming that there will be no identifiers in $L_{Control}$.

An interesting question here is: if we set specific thresholds on the interest group and control group, for what thresholds would we find the most interesting set of rules? To this end, a so-called *Permutation Counting Matrix* (PCM) was implemented.

A PCM is a structure that registers how many rules in the experiment apply to various threshold boundaries. On both axes, threshold boundaries are represented in percentages. How many thresholds are represented depends on the matrix resolution, which is by default set to 100, meaning that each axis has 100 thresholds, each threshold representing one percent. This number can be changed by the user, since experiments with a large number of identifiers require a higher resolution to still be able to discern smaller subgroups.

On the horizontal axis, also called the $max\_threshold$ axis, the control group is represented. Each threshold on this axes represents the percentage of identifiers in the entire control ranking that a rule can have at maximum. Note that if a rule satisfies a certain threshold x on a matrix of resolution $n \geq x$, then it satisfies all subsequent thresholds $x + 1, \ldots, n$.

On the vertical axis, which is called the $min\_threshold$ axis, the interest group is represented. In this case, each threshold represents the percentage of identifiers in the entire interest ranking that a rule can have at minimum. This implies that if a rule satisfies a certain threshold x on a matrix of resolution $n \geq x$, then it satisfies all

subsequent thresholds $1, \ldots, x$.

Consider PCM A below, which has a resolution of $m \cdot n$, whereby m is the $min\_threshold$ resolution, and n is the $max\_threshold$ resolution.

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

According to the properties discussed above, $a_{m,n} \geq a_{m,n-1} \geq \ldots \geq a_{m,1}$, and $a_{1,n} \geq a_{2,n} \geq \ldots \geq a_{m,n}$. This implies that the rules with potentially best scores have a high m index, while the n index is low, ultimately making rules belonging to $a_{m,1}$ the best rules, containing rules that describe many interest group identifiers, while none or very little of the control group.

We can use the structure above to determine optimal threshold settings for interest and control classes by using exact testing, whereby *optimal* indicates threshold settings where the PCM value of the original input differs most from the PCM value in the PCMs of the permutations. To calculate these settings, we subtract the permutation PCMs from the original one. The optimal thresholds are indicated by the matrix element with the highest value. We then select the rules of the original list and the permutations that adhere to these thresholds, compare them, and prune rules that are better in permutations. The result is then returned to the user.

Note that this type of experiment is more focussed on participations and threshold than on interestingness. Percentiles on the axes indicate the weighed percentiles of identifier scores, which is determined as follows: for each ranking R containing all identifiers in the experiment, and for a list of identifiers L associated to a specific rule, the weighed score percentile is calculated by the following formula:

$$\frac{\sum_{id \in L} s_{id}}{\sum_{id \in R} s_{id}}$$

Participation thresholds should also be set to a lower minimum in these kinds of experiment, since the purpose of these experiments is to find optimal thresholds by itself. However, some participation thresholds might be needed to avoid rules that have far more control identifiers than interesting ones, or rules that have merely one or two interesting identifiers and thus are not very informative or substantial for the group as a whole.

A workflow of multi-class exact testing is shown in Figure 6.2. As can be seen, phases one and two are the same, only phase three differs. Not only pruning takes place here, but also recombination of PCMs, and determination of the optimal class thresholds.

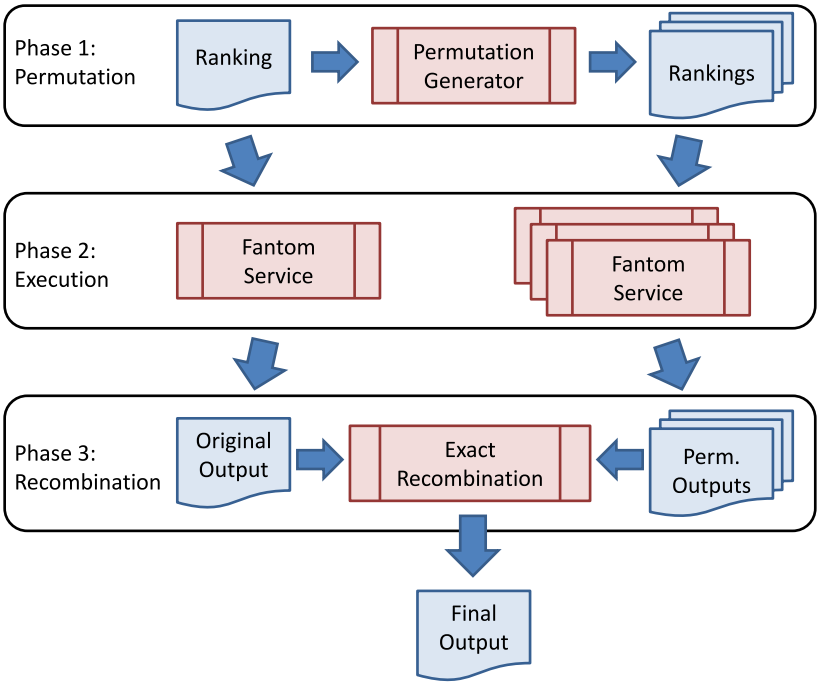


Figure 6.2: Workflow for multi-class exact optimization

6.3 Experimental results

In this section we discuss the experimental results of the two versions of exact testing with Fantom. We applied both versions on the AML vs. ALL publicly available microarray data set that compares gene expression profiles of AML and ALL [ASS+02], and present statistics on run times, pruning and parallelism.

Before we discuss the different experiments we will first provide a basic description of that data set. In the ALL vs. AML microarray data set, there are a total of $7,129$ probes, and 72 measurements per probe, 25 for AML patients, 47 for ALL. Mapping from probes to ENTREZ gene identifiers was performed with the Hu6800 annotations supplied by Affymetrix, and after elimination of probes that have no gene association or multiple ones, $5,445$ unique genes remained. To normalize the raw data, we used Quantile normalization again [BIAS03]. Rankings are obtained by performing t-value calculation with the Student's t-test between groups labelled with AML and ALL.

For the remainder of this section, all experiments were carried out on one or multiple machines all with the same configuration, namely an Intel Core Duo 2 times 2 GHz, with 4 GB of RAM.

6.3.1 Exact Testing: Single-Class Pruning

Single-class pruning is a process of three phases: permutation generation, rule discovery, and rule pruning. In the first phase of our experiments, permutations were generated from the microarray data set by swapping labels. By generating permutations in this way, instead of modifying values in the ranked list, we ensure that dependencies between genes are preserved. Since there are 72 labels to consider, the total number of unique permutations would be:

$$\frac{72!}{25!\cdot 47!} = 1.53 * 10^{19}$$

Since this number exceeded our computational resources, which were limited to 16 computers, we generated 15 random permutations for each experiment, and ran all 16 simultaneously. We performed two different experiments, one generating rules that associates ontological concepts directly with genes, and another that uses the interaction option, indirectly associating genes with ontological terms through interaction with other genes. For each of these types, we performed three different experiments with low, medium, and high thresholds. To generate as many experiments as possible, we modified Fantom so that it will only allow at most one concept for each predicate per rule. Both the GO and KEGG ontologies were used which resulted in rules containing a maximum of four predicates.

Since interaction experiments are far more intensive in terms of computation and data access, a normal, unbiased way of experimentation resulted in a rule explosion that was unfeasible to be evaluated by experts. This is due to the interaction data, which documents many interactions, and thus many genes were associated to each ontological term. As a result, we had to insert a bias, declaring a list of genes that had to be in the rules. In our case, we declared this list to be the top 200 and bottom 200 genes, which are most differentially expressed. The participation threshold then only applies to the biased part instead of the entire ranking. This way, we avoided rule explosion at the cost of exhaustiveness. However, since the genes in the list were already at the top of the ranking, rules that could be found without a bias were likely to have a low interestingness. Note that due to the identifier association explosion, thresholds in the interaction experiments are still high in terms of participation.

First let us consider speed results of the experiments, shown in Table 6.1. We outlined several statistics, such as participation threshold PT, minimum interestingness score IS, average completion time of the original input T_O in minutes (m) and seconds (s), lowest completion time of a permutation $min(T_P)$, highest completion time of a permutation $max(T_P)$, average completion time of permutations $av(T_P)$, and overhead of exact rule pruning T_{Pr}, which is the time of pruning the original rules by evaluating the outcome of the permutations.

Note that for each association type, we took three different configurations reflecting a low, medium and high threshold setting. These values differ between association types, since interaction association experiments involve more identifiers, but

their scores tend to be lower. For direct association experiments, all values are the averages of 20 runs, where in each run we generated different random permutations. Since interaction association experiments took much longer, all values of those experiments are the averages of 5 runs.

Direct Association Experiment						
PT	IS	T_O	$min(T_P)$	$max(T_P)$	$av(T_P)$	T_{Pr}
8	0.40	2m	1m58s	2m	2m	5m41s
10	0.50	1m47s	1m41s	1m48s	1m44s	3m47s
12	0.70	1m23s	1m11s	1m26s	1m18s	18s

Interaction Association Experiment						
PT	IS	T_O	$min(T_P)$	$max(T_P)$	$av(T_P)$	T_{Pr}
60	0.35	233m05s	203m31s	222m18s	210m14s	17m41s
75	0.40	149m51s	134m32s	148m22s	146m12s	10m11s
80	0.50	129m50s	119m11s	131m27s	124m34s	1m27s

Table 6.1: Exact test single-class pruning benchmarks

As can be seen in Table 6.1, interaction association experiments are much more data and processing intensive, yet exact pruning overhead increases much less dramatically. This is because options in interaction association experiments contain many genes associated to them, which makes the calculation of the maximum ES score very expensive. Exact pruning is only dependent on the number of rules in the original rule set and those of the permutations, which increase less dramatically.

Another observation is that execution times tend to increase exponentially with even a minor change in threshold setting, thus starting out with a high threshold and then moving to lower ones seems like the best strategy to apply.

A final observation is that rule generation times do not seem to differ much between the original input and the permutations, although experiments with permutations do seem to consistently take less time. This is because in some permutations very few rules could be generated with the given thresholds, sometimes even none, which reduces experiment times significantly.

Now let us consider pruning results of the experiments, shown in Table 6.2. We again outline several measurements for each configuration, such as the number of rules generated in the original input R_O, the minimum number of rules generated in the permutations $min(R_P)$, the maximum number of rules generated in the permutations $max(R_P)$, the average number of rules generated in the permutations $av(R_P)$, and the average number of rules pruned from the original set $av(R_{Pr})$.

We can see in Table 6.2 that permutations consistently yield less rules, which

Direct Association Experiment						
PT	IS	R_O	$min(R_P)$	$max(R_P)$	$av(R_P)$	$av(R_{Pr})$
8	0.40	2,964	2,625	2,815	2,762	224
10	0.50	1,252	539	775	681	95
12	0.70	27	0	6	4	3

Interaction Association Experiment						
PT	IS	R_O	$min(R_P)$	$max(R_P)$	$av(R_P)$	$av(R_{Prune})$
60	0.35	26,212	17,544	21,996	20,112	1,766
75	0.40	13,612	5,344	11,278	9,775	728
80	0.50	4	0	2	1	0

Table 6.2: Exact test single-class pruning results

explains the shorter experiment time. Pruning statistics do not differ much between direct and interaction association experiments, if we consider the last measurement an outlier. In both direct and interaction association experiments, higher settings result in relatively better pruning; at low settings around 6–7% of the original rules get pruned, while in higher settings this is as much as 11%.

6.3.2 Exact Testing: Multi-Class Threshold Optimization

In multi-class experiments we investigate what the interest class or classes have in common with each other, or how they differ with respect to the rest of the identifiers that do not fall in these classes. In our experiments we labelled all genes with a t-value greater than 0.5 as interesting, and compared this group with the rest of the genes in the input. As a consequence, instead of $5,445$ genes to be considered, the interest group now contained only 662 genes, while the rest was serving as contrast information for scoring purposes.

Participation thresholds in these experiments kept very low, since we want to uncover the optimal thresholds for which to return rules for to the user. Setting thresholds too high could interfere with this process, and yield a suboptimal result. Note that all participation thresholds now only hold for the interest group instead of the whole ranking. This has implications for the permutations generated, since there have to be at least that many genes in the interest group to generate any rules at all. Therefore, a bit of bias in the permutations cannot be avoided, and thus we cannot speak of true randomness.

Once again we consider speed results of the experiments, which are shown in Table 6.3. We measured the same statistics as in Table 6.1, again for three different configurations. As can be seen, compared to the single-class experiments, these experiments take less time for the direct associations, but more time for the interac-

Direct Association Experiment						
PT	IS	T_O	$min(T_P)$	$max(T_P)$	$av(T_P)$	T_{Pr}
4	0.40	1m19s	1m12s	1m19sm	1m4s	8s
6	0.50	1m14s	1m8s	1m12s	1m11s	7s
8	0.70	1m7s	1m2s	1m10s	1m6s	5s

Interaction Association Experiment						
PT	IS	T_O	$min(T_P)$	$max(T_P)$	$av(T_P)$	T_{Pr}
30	0.35	610m05s	418m12s	584m12s	556m59s	44s
40	0.50	605m41s	432m32s	600m17s	541m51s	32s
50	0.60	608m33s	444m21s	567m22s	554m22s	11s

Table 6.3: Exact test multi-class pruning benchmarks

tion associations. This is due to the fact that direct association experiments are not negatively influenced as much by a lower threshold as interaction association experiments are. In interaction associations, groups in rules tend to experience an explosive growth, and this gets worse when lower thresholds are chosen.

Another observation is that experiment times seem to be rather constant. This is not surprising, since the experimental participation threshold was kept constant, and Fantom's pruning effect can really be seen in later stages of rule generation. Since we modified Fantom to only allow one ontological concept per predicate, those later stages never appear, and thus pruning on the basis of ES is minimal.

Finally, it seems that pruning takes a very short time, indicating that there were few rules that could be used for pruning; few rules were generated, and the rules that were generated had a lower score than the original rules generated, thus not much pruning was done.

Now let us consider the pruning results, shown below in Table 6.4. We included the same statistics as in Table 6.2 with an added column R_{Opt}, which denotes the number of rules given the optimal PCM thresholds. As can be seen, optimized thresholds yield a pruning optimization in rules of 40 to 50 percent in direct association experiments, and between 60 to 90 percent in interaction association experiments (discarding the last threshold, which seems to be an outlier due to its strictness).

Once more, permutations do not yield many rules, and as a result additional exact pruning also has very little result. This is due to the design of the experiment. We set the threshold for over-expression to a t-value of 0.5. When generation permutations of the class labels, we influenced the t-values of the permutations, and thereby the ranking. As a result, there were less permutations that had sufficient genes with a t-value over 0.5, hence permutations structurally yield less rules.

Another reason why pruning yields so little result is because of the low thresh-

Direct Association Experiment							
PT	IS	R_O	R_{Opt}	$min(R_P)$	$max(R_P)$	$av(R_P)$	$av(R_{Pr})$
4	0.40	1,182	638	0	5	3	2
6	0.50	946	524	0	3	2	0
8	0.70	436	227	0	1	1	0

Interaction Association Experiment							
PT	IS	R_O	R_{Opt}	$min(R_P)$	$max(R_P)$	$av(R_P)$	$av(R_{Pr})$
30	0.35	171,071	21,349	1,325	1,833	1,411	21
40	0.50	11,113	4,453	0	121	32	8
50	0.60	1	1	0	1	0	0

Table 6.4: Exact test multi-class pruning results

olds. Rules with small subgroups often contain very specific rules with high scores. In permutations, other genes are bound to be in the interest sets, yielding rules that contain more specific concepts located in other parts of the ontology. As a result, these rules cannot be used to prune the original ones since they are not related to these rules.

6.4 Conclusions and Future Work

In this chapter we discussed two ways of exact testing with Fantom in order to prune Fantom outputs even further, and to optimize participation thresholds in case of a multi-class problem. We described how through a variation on Fisher's Exact Test we could prune more rules in addition to the pruning conducted by the Fantom pruning algorithms. We also showed that with permutation testing we could find optimal participation thresholds for multi-class problems, as well as use the exact pruning method on it, but with less effect.

In single-class problems, both direct association and interaction association experiments had benefit from exact pruning, ranging from 6% with low thresholds to about 10% when thresholds are set high. A reason for this difference is that at high settings, there are not many rules, so if one does get pruned, the impact is much higher than on lower settings, where the impact is less. An overall performance of this algorithm is to prune about 6–8% of the rules.

In terms of performance, exact pruning on single-class problems is especially worthwhile in interaction association problems. Where in direct-association problems the overhead is sometimes relatively grave, e.g., more than 200%, it is relatively low for interaction association experiments, where the experiments themselves can sometimes take hours. Overall, overhead on interaction association experiments was

between 1–7%, which was much better compared to the direct association experiments.

In multi-class problems, experiment time was more or less constant if the participation threshold was fixed, since pruning had minimal effect for rule generation for small collections of rule concepts. Since exact pruning had little effect, those pruning overheads were significantly lower for direct association experiments as well, between 7–10% of the experiment time. Overhead for the interaction association experiments was almost negligible.

Pruning due to optimized constraints of the rules yielded good results in both direct and interaction association experiments. For direct associations, pruning varied between 45–50% of the rules generated. For interaction associations, this number was higher, pruning from 65% to as much as 88% of the rules. Exact pruning had almost no effect on both data sets, due to the experimental design and the fact that low thresholds create highly specific rules with high scores, which are unlikely to be pruned by rules generated from a permutation on the class labels, since that can modify the ranking profoundly.

Finally, we would like to discuss future work. Apart from pruning with exact tests, p-values for resulting rules could also be established with some modifications to the algorithm, although that would require more permutations than the 15 we generated. These p-values could give an even better indication on how special a specific rule is.

For multi-class problems, we kept the number of classes to two for now, but there are many problems that deal with more than two classes, so one interesting question is how to deal with those. Furthermore, we created a matrix based on participation thresholds. It would be interesting to see if the optimized rules would differ much when we optimize on the basis of score thresholds, or a mixture between score and participation.

Part III

Applications

Chapter 7

Gene Experiments: Mouse Hearts

In this chapter we use the Fantom service to perform experiments on microarray expression concerning mice that show cardiac overexpression of the transcription factor TBX3. We perform both direct association experiments and interaction association experiments, and for each of these types we discuss performance, pruning and rules. Where possible, we also compare the results of the Fantom service with the results of the DAVID tool.

7.1 Introduction

A microarray study measures the activity of many genes in a biological entity at a certain time. The result of a microarray study is a matrix of numbers reflecting the expression of genes in different conditions. By comparing gene expressions for each gene for different classes of entities, genes can be assigned an activity score for each class. In our experiments this is the t-score, calculated by the Student's t-test. As a result, a ranking of genes can be produced that reflects the extent and significance of each gene's activity in the conditions under study.

The Fantom service takes as input this ranking of genes together with additional information, such as species, gene identifier type, ontologies and thresholds, to find subgroups in those genes that match a rule, which is a conjunction of ontological concepts that all the genes in the subgroup are associated with. This is called a *direct association experiment*, for Fantom uncovers rules that contain ontological concepts that can be directly associated to genes.

Fantom also provides the option to search for rules that can be indirectly associated with subgroups of genes within the gene ranking; this is called an *interaction association experiment*. By including protein-protein interaction data, Fantom can extract subgroups that interact with other groups that have certain ontological properties. These rules can give the experimenter more insight in the influence of genes beyond the scope of direct influence, providing a richer view on complex biological

processes.

In this chapter we will perform both types of experiments on data generated by a microarray study on two different groups of mice; one group consists of control mice and the other group contains mice that express the TBX3 transgene. We will measure experiment performance and pruning statistics for both types, present the most important rules and describe the predominant cluster themes. Furthermore, we will apply exact pruning in both experiment types, thereby providing statistics on experiment performance on permutations, as well as statistics on exact pruning. The goal is to give an indication of the performance of the Fantom service in diverse areas such as pruning and execution time, not to present a complete and thorough interpretation of the results; for each experiment we will provide the top rules, and discuss them briefly.

This chapter is organized as follows. In Section 2, we will present an overview of microarray studies and the biological backgrounds of the experiment. We will discuss what microarray studies are and what kind of processes are involved. In Section 3, we discuss the microarray study on the mouse heart, discussing the reasons for and goals of the study, and its goals. In Section 4, we will present the experiments we performed with the Fantom service. We discuss experiment design and present statistics on performance as well as pruning for both direct association and interaction association experiments. Finally, in Section 5, we will draw some conclusions from the results presented in Section 4.

7.2 Biological Backgrounds and Microarrays

In this section we discuss the biological backgrounds of microarray studies, and the microarray technology itself.

7.2.1 Biological Backgrounds

All information about the functioning and construction of a living organism is stored in Deoxyribo Nucleic Acid (DNA) molecules. These molecules are said to be the blueprint of an organism, for they contain almost all traits. These traits can be visible, such as eye color or hair color, but also invisible, such as the functions of an organism's immune system, or the presence of hereditary diseases.

DNA is present in each cell of an organism, where it is organized into long strings called *chromosomes*. These chromosomes can be seen as chains of DNA molecules bonded together, also called macromolecules, or *polymers*. The DNA molecules that together form the polymer are called *nucleotides*. There are four kinds of nucleotides in DNA: Adenine (A), Guanine (G), Thymine (T) and Cystosine (C). Chromosomes are organized in pairs that bond together, thereby forming a spiral, know as the double helix. Each nucleotide on a polymer bonds with the opposite nucleotide on the

other polymer, forming what is called a *base-pair*. However, not all bondings are possible; Adenine can only bond with Guanine, while Thymine can only bond with Cytosine. This is called the Watson-Crick complementary. An illustration is shown in Figure 7.1[1].

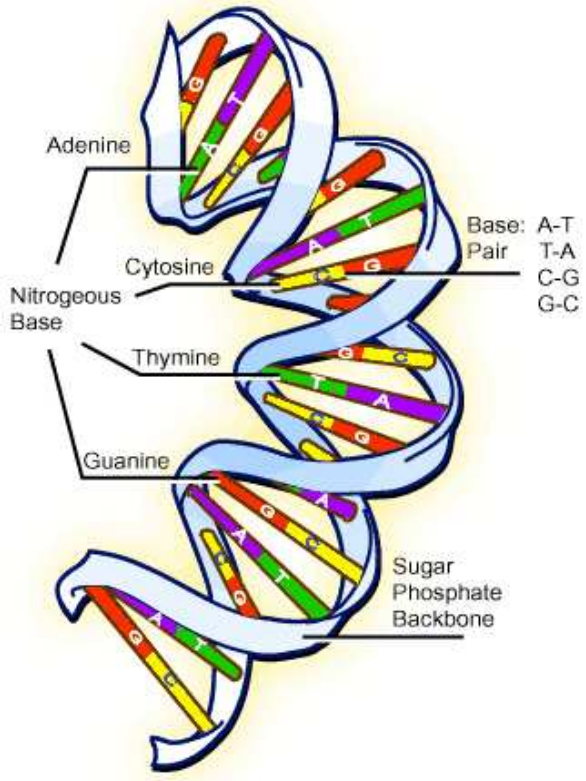


Figure 7.1: A chromosome molecule

DNA contains information thousands of different biological processes. Regions of a chromosome that contain information on the support or execution of these processes are called *genes*. Each gene can play a role in various processes of the body. When such a process needs to be carried out, the DNA double helix is unravelled, and the information contained in genes on a single strand is translated into messenger Ribo Nucleic Acid (mRNA). The main differences between DNA and RNA is that RNA is a single strand, and all the Thymine molecules are translated into Uracil (U) molecules. This mRNA is then translated into amino acids, which bond together to

[1]Image taken from the Science Creative Quarterly, http://www.scq.ubc.ca/a-monks-flourishing-garden-the-basics-of-molecular-biology-explained/

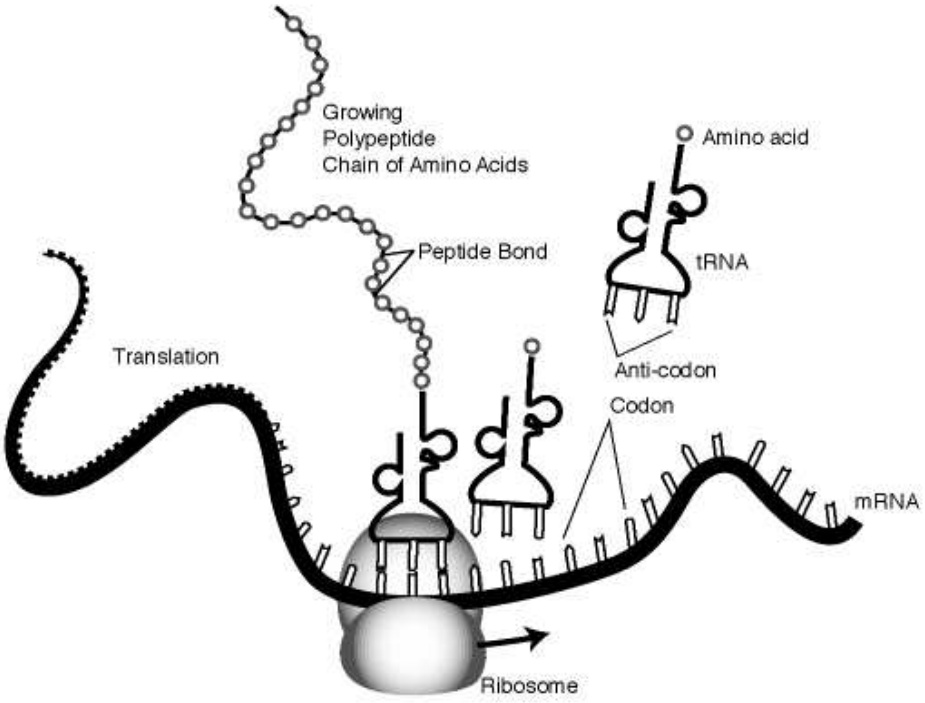


Figure 7.2: The mRNA translation process

form proteins. These proteins then perform the required processes. Figure 7.2[2] shows this process.

7.2.2 Microarrays

Whenever a biological process needs to be performed, the genes that are responsible for that process become more active, meaning that they are translated into many mRNA strands. Microarrays are designed to detect those mRNA strands, and to measure their intensity.

Microarrays are arrays of thousands of microscopic spots. Each of these spots contains thousands of short mRNA sequences that match the gene of interest, or at least part of the gene. These partial strands are called *probes*. The probes bind to the target genes by means of *hybridization*: The probe is a complementary nucleic acid

[2] Image taken from National Human Genome Research Institute, http://www.genome.gov/Pages/Hyperion/DIR/VIP/Glossary/Illustration/Images/peptide.gif

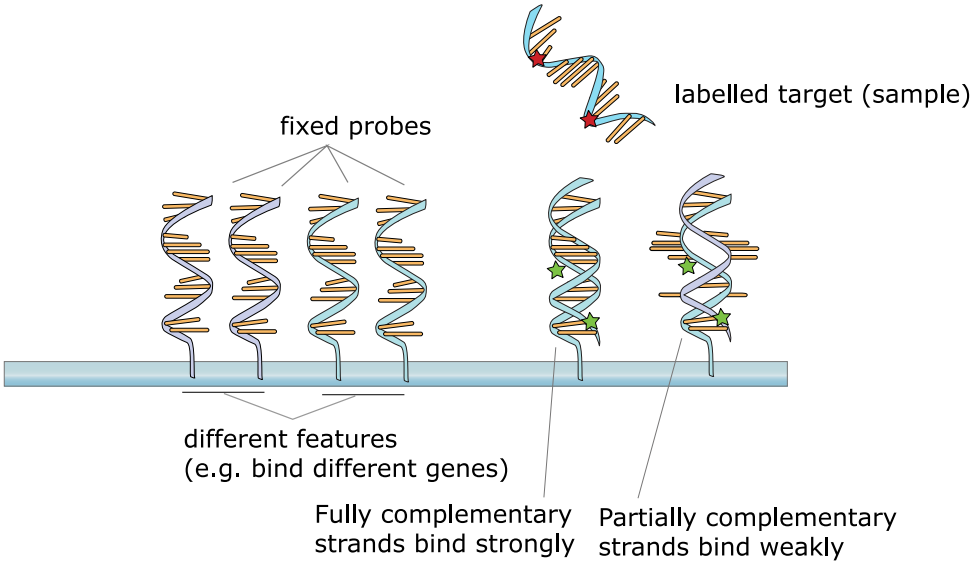


Figure 7.3: Microarray hybridization

sequence, designed to pair with the target mRNA sequence. Figure 7.3[3] shows this process.

As can be seen in Figure 7.3, probes can bind to multiple mRNAs that have partially the same sequence, but the target genes provide the strongest bonds. By washing the samples after hybridization, the weak bonds can be filtered out.

Each target is labelled with a fluorescent substance. When more targets bonded to a probe, more light is emitted. Microarrays use relative quantitation in which the intensity of a feature is compared to the intensity of the same feature under a different condition. By comparing the intensities of the two conditions, a number is produced that indicates the expression of a certain probe, and thus gene.

Microarrays are used to measure expression levels or changes in expression levels of genes. They can also be used to detect Single Nucleotide Polymorphisms, which we will discuss in the next chapter. Another use is the Comparative Genomic Hybridization (CGH), in which DNA sequences are compared for changes, such as gains or losses. This is often applied with tumor cells.

7.3 Microarray Study on Mouse Hearts

In the TBX3 microarray study, gene expression patterns in the heart of two groups of mice were compared. One group consists of the control group, containing mice that

[3] Image taken from Wikipedia,
http://en.wikipedia.org/wiki/DNA_microarray

did not express TBX3, the other group consisted of mice which expressed the TBX3 transgene. TBX3 is a transcription factor that is important for heart development; it is involved in the specification of myocardiumial tissue into working myocardium, as well as the myocardium of the conduction system.

There are important differences between these two types of myocardium. Working myocardium contracts fast, is metabolically highly active and has no autonomous contractile activity, whereas the conduction myocardium contracts slowly, is metabolically less active and has spontaneous pacemaker activity.

TBX3 is normally only active in the conduction myocardium where it represses the working myocardium activity and ensures its pacemaker activity. In the transgenic mice under study, TBX3 is also active in the working myocardium where it is not supposed to be expressed. This leads to a loss of synchronous contractions (arrhythmias).

The study of the transgenic mice helps to identify the genes and pathways important for the specification of the two different types of myocardium, which may help to find a way to suppress or avoid TBX3 expression in the working myocardium.

7.4 Experimental Results

In this section we will present and discuss the results of the various experiments we performed on the TBX3 microarray study. We performed both direct association and interaction association experiments. The microarray data contained $48,318$ probes that were used to measure gene expressions of 12 samples, 6 samples of the control group and 6 of the TBX3 group. After mapping probes to the SYMBOL identifier type, a ranking of $34,327$ genes remained. No losses occurred in mapping SYMBOL identifiers to the internal ENTREZ representation. Scores of those probes were obtained by using the Student's t-test on the samples, and taking the absolute values of the outcome.

As an extra pruning step we applied exact pruning on the outcome of the experiments by performing the same experiment on permutations of the original ranking. These permutations were generated in the same way as was done in Section 6.3.1, by swapping labels. For the TBX3 data set, the total number of permutations is:

$$\frac{12!}{6!\cdot 6!} - 1 = 923$$

We generated 16 random permutations for each experiment, and then performed the Student's t-test on each probe again to obtain the new scores.

All experiments were conducted with the Fantom service implementation discussed in Chapter 5, and performed on Windows Vista with an Intel Core 2 Duo T6400 2 GHz CPU and 4 GB RAM. The experiment setup also remained the same: for each experiment type we examined performance and pruning by varying support threshold P and keeping the score threshold S constant (support-dependent measurements). For the direct association experiments, we also studied the effect of the

score threshold by varying it, while keeping the support threshold constant (score-dependent measurements).

For each type of experiment we spread the measurements over two tables. The first table contains the performance measurements, which include the average time for the experiment to complete over 10 runs on the original ranking $T_{Orig_{avg}}$ in minutes m and seconds s, the minimum time for the experiment to complete over 10 runs on a permutation $T_{Perm_{min}}$, the maximum time for the experiment to complete over 10 runs on a permutation $T_{Perm_{max}}$ and the average time for the experiment to complete over 10 runs on all 16 permutations $T_{Perm_{avg}}$.

The second table contains pruning statistics, which include the number of rule options generated on the original ranking *#Options*, the number of options pruned on the original ranking *#Pruned*, the pruning ratio $Ratio_{Pruned}$, the number of rules pruned by exact pruning *#XPruned* and the ratio of rules pruned by exact pruning $Ratio_{XPruned}$.

Apart from performance and pruning statistics we also present the results for each experiment. We present the top rules, along with an interpretation. Apart from individual rules we also discuss themes and trends in the output by describing the clusters that were generated on the output.

7.4.1 Direct Association Experiments

We first discuss the direct association experiments. In this section we will discuss performance, pruning and the resulting rules, each in a separate subsection.

Performance

Table 7.1 shows support-dependent and score-dependent performance measurements for the direct association experiments. For the upper half of the table, which measure support-dependent measurements, the score threshold was fixed to $S = 0.60$, while the support was varied. For the lower half of the table, the score-dependent measurements, the support threshold was kept constant at $P = 13$ while the score thresholds were varied.

For the original ranking, execution time seems to grow exponentially as the support threshold gets lower, while this is not always the case for the permutations. This indicates such behaviour is ranking-dependent. Some rankings contain very little rules with sufficient support, as the average execution time of the permutations does not seem to incline as much as the average execution time of the original ranking.

For the score-dependent measurements, a similar statement could be made, although inclines are not as dramatic as with support-dependent measurements. Average completion times for permutations are lower, which is to be expected since scores

Support-dependent measurements				
Threshold	$T_{Orig_{avg}}$	$T_{Perm_{min}}$	$T_{Perm_{max}}$	$T_{Perm_{avg}}$
$P = 11$	70m41s	8m20s	74m13s	26m02s
$P = 13$	22m50s	7m59s	26m06s	12m22s
$P = 15$	14m07s	6m10s	15m06s	7m31s
$P = 20$	7m44s	5m58s	7m30s	6m09s

Score-dependent measurements				
Threshold	$T_{Orig_{avg}}$	$T_{Perm_{min}}$	$T_{Perm_{max}}$	$T_{Perm_{avg}}$
$S = 0.55$	44m20s	8m25s	58m48s	18m13s
$S = 0.60$	22m50s	7m59s	26m06s	12m22s
$S = 0.65$	15m24s	7m16s	10m25s	8m30s
$S = 0.70$	10m51s	5m59s	7m41s	6m35s

Table 7.1: Performance measurements for the direct association experiments

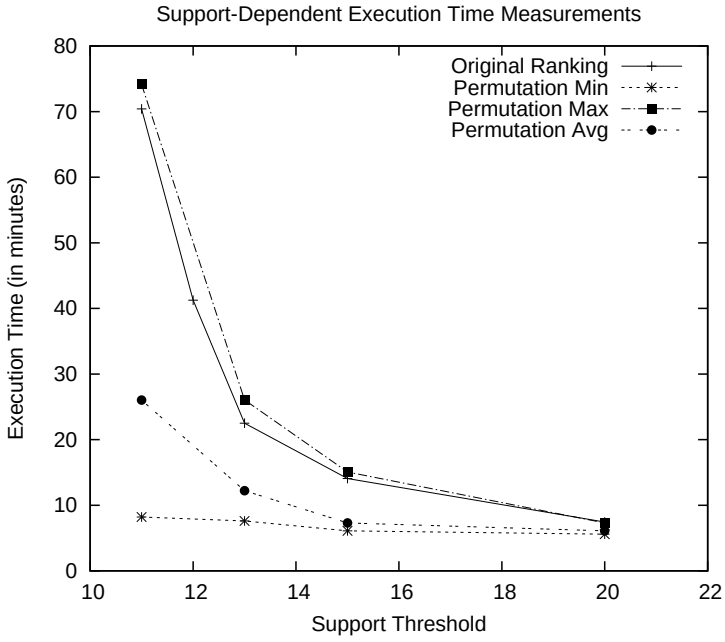
in the permutation lists are lower on the whole, thus the resulting rules generally have lower scores as well.

Figure 7.4 illustrates the performance behaviour of the direct association experiments. Figure 7.4(a) shows support-dependent behaviour for the original ranking and the permutations, while Figure 7.4(b) shows score-dependent behaviour. As can be clearly seen, behaviour is the same across all the measurements, although in case of Figure 7.4(b), execution time for the original ranking decreases at a slower pace, for it keeps generating more rules than in the permutations. Intuitively this is correct; the permutation rankings are randomized rankings and therefore should contain identifiers with lower scores, resulting in rules with lower scores. Therefore, if the score threshold approaches a boundary that is considered high, such as $S = 0.70$, the permutations should yield little rules, in any case less than a ranking that was obtained from a directed experiment and not random generation.

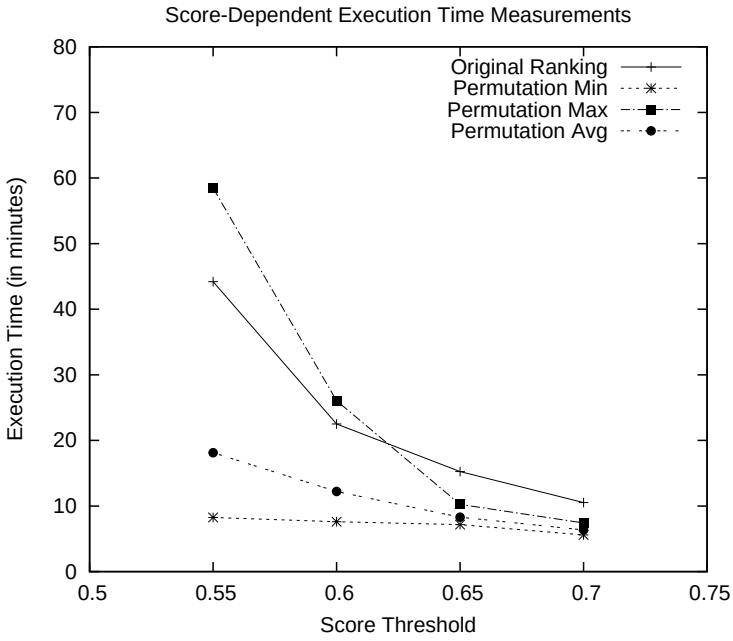
Pruning

Table 7.2 shows support-dependent and score-dependent pruning measurements for the direct association experiments. Once again, the score threshold in the upper table was fixed to $S = 0.60$, while the support threshold in the lower table was kept constant at $P = 13$.

For support-dependent measurements, pruning ratio goes down as the threshold grows. This is due to the design of the Fantom service; rules with high support usually contain more general terms and ontological concepts, while rules with lower support are more specific refinements of the rules with higher support. By design, those rules are kept, and thus less pruning takes place.



(a)



(b)

Figure 7.4: Execution time measurements for direct association experiments

Support-dependent measurements					
Threshold	*#Options*	*#Pruned*	$Ratio_{Pruned}$	*#XPruned*	$Ratio_{XPruned}$
$P = 11$	949,179	947,454	0.9982	159	0.0921
$P = 13$	470,270	469,097	0.9975	78	0.0665
$P = 15$	289,462	288,618	0.9971	36	0.0426
$P = 20$	109,209	108,847	0.9967	5	0.0148

Score-dependent measurements					
Threshold	*#Options*	*#Pruned*	$Ratio_{Pruned}$	*#XPruned*	$Ratio_{XPruned}$
$S = 0.55$	628,586	626,543	0.9967	364	0.1782
$S = 0.60$	470,270	469,097	0.9975	78	0.0665
$S = 0.65$	375,332	374,721	0.9984	10	0.0163
$S = 0.70$	251,084	250,847	0.9991	0	0

Table 7.2: Pruning measurements for the direct association experiments

Exact pruning lowers as the support threshold increases. On average, permutations need less time and generate much less rules. As support thresholds increase, even less rules will be generated. Given that rules generated by permutations generally yield lower scores, pruning decreases as support threshold increases.

For the score-dependent threshold, pruning monotonically increases, which is to be expected from a threshold that has pruning as primary purpose, Furthermore, lowering the score threshold seems to have more influence on exact pruning than lowering support thresholds. This indicates that this technique is especially useful when trying to discover rules with lower score thresholds but still specific to the ranking.

Results

In this part we discuss the rule output of the experiments. Since the TBX3 data set is not publicly available yet, we will omit the genes involved in the rules, and just give the support, the score and the rule itself.

When looking at the rules generated with $P = 11$ and $S = 0.60$, we find rules that are reasonably diverse. The top 4 rules are displayed on the next page. As can be seen, returning concepts are mitochondrion and membrane, which is expected since those are the areas that were studied. Interesting concepts unrelated to cardiac problems are seen in the fourth rule, which links strongly to different types of cancer, and in the second rule, which links to neurodegenerative diseases.

```
Rule 1
Support: [15]
Score: 0,855
All genes in the subgroup
  have the following properties:
    cellular_component(mitochondrion),
    biological_process(aerobic respiration),
    biological_process(acetyl-CoA catabolic process),
    KEGG_pathway(Citrate cycle (TCA cycle))

Rule 2
Support: [11]
Score: 0,849
All genes in the subgroup
  have the following properties:
    molecular_function(iron ion binding),
    cellular_component(mitochondrial inner membrane),
    KEGG_pathway(Alzheimer’s disease),
    KEGG_pathway(Parkinson’s disease),
    KEGG_pathway(Huntington’s disease)

Rule 3
Support: [11]
Score: 0,848
All genes in the subgroup
  have the following properties:
    biological_process(neuron differentiation),
    cell_comp(intracellular membrane-bounded organelle),
    biological_process(regulation of transcription),
    KEGG_pathway(Signal Transduction)

Rule 4
Support: [11]
Score: 0,845
All genes in the subgroup
  have the following properties:
    cell_comp(intracellular membrane-bounded organelle),
    KEGG_pathway(Signal Transduction),
    KEGG_pathway(Endometrial cancer),
    KEGG_pathway(Glioma),
    KEGG_pathway(Prostate cancer),
    KEGG_pathway(Melanoma)
```

Because we do not want to present only individual rules, we also looked at global trends by performing clustering on the rules. When clustered with a similarity score of 0.60, thus an overlap of terms of 60%, we see three distinct themes, which agree with the top 4 rules: the first theme is centered around the mitochondrial part and transmembrane transporter activity, the second theme considers signal transduction, and the third theme deals primarily with neurodegenerative diseases.

7.4.2 Interaction Association Experiments

For the interaction association experiments we faced a problem. Because of the interactions, so many concepts were annotated that mining for rules with small enough thresholds to be interesting took so much time and memory that it was beyond the capabilities of the Fantom service. However, we could partially circumvent this problem by using a biased approach, which we discussed in Section 6.3.1

In the biased approach, we confine the support threshold to a fixed class, indicating that each rule should have as least that amount of identifiers of the fixed class. In case of our experiments, we defined the fixed class as consisting of those genes that have a score higher than 2. Any higher would limit the search space too much, and any lower resulted in an explosive time increase on the experiments, while not producing many significant changes in the rules.

After relabelling al the genes according to this process, of the $34,327$ involved in the experiment, $1,909$ were labelled as the fixed class. Even then, if we took a support threshold of 25 or less, the experiment time would take well over 5 hours, without any significant result in the rules.

For the permutations in the exact pruning mechanism we labelled the same genes as the fixed class as those in the original ranking, since we need to prune rules that are partially made up of that fixed group. However, since these genes mostly have lower scores in the permutation rankings, rules with higher or equal scores would have never been reached. Therefore we allowed for a bit of score leniency, stating that if a rule appears with a high enough score (in the experiments we took score threshold 0.60) that is also in the original rule output, then we pruned it.

Performance

For the interaction association experiments we solely took support-dependent performance measurements; since the results of those experiments all had scores of 0.90 or higher, doing score-dependent measurements had little added value in such a small spectrum of scores.

For the support-dependent measurements, the score threshold was fixed to $S = 0.60$ for the original list, while for the permutations it was set to $S = 0.40$. Results are shown in Table 7.3.

Since the score threshold for the original ranking and the permutations differ, we

Support-dependent measurements				
Threshold	$T_{Orig_{avg}}$	$T_{Perm_{min}}$	$T_{Perm_{max}}$	$T_{Perm_{avg}}$
$P = 26$	44m33s	37s	45m30s	9m58s
$P = 27$	7m57s	32s	7m41s	2m23s
$P = 28$	3m27s	29s	3m39s	1m20s
$P = 29$	3m15s	28s	3m20s	1m09s
$P = 30$	2m20s	26s	2m24s	52s

Table 7.3: Performance measurements for interaction association experiments

cannot perform a direct comparisons. However, we can conclude that they follow a similar pattern as in Table 7.1. The use of interactions did have a small effect on the explosive nature of the curve, which si somewhat steeper now, as can be seen in Figure 7.5. As we will explain in the next chapter, the shape and position of this curve could be important for automatic threshold selection.

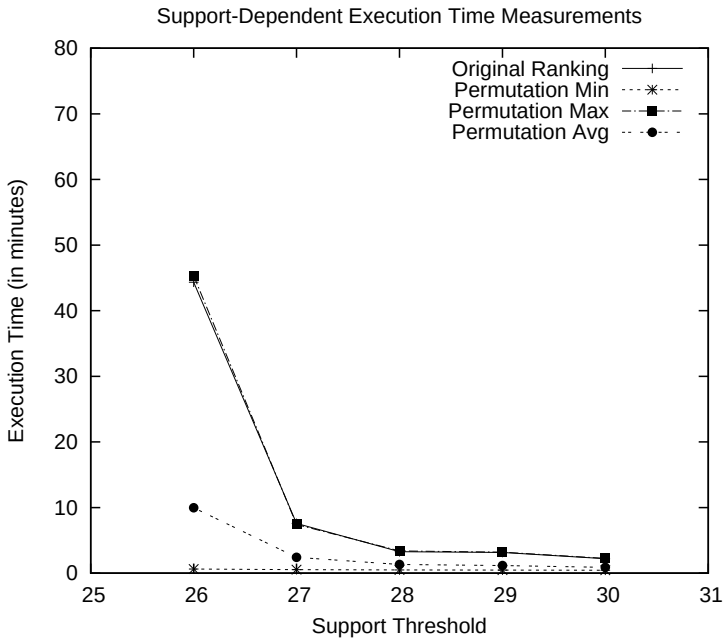


Figure 7.5: Execution time measurements for interaction association experiments

As can be seen in Figure 7.5, the permutations can still require a significant amount of processing in interaction association experiments, as was the case in direct association experiments. Overall, however, permutations in interaction association experiments yield much less rules when combined with the bias. Without bias, however, the experiments would take a very long time.

Pruning

When we look at the pruning data shown in Table 7.4, we notice that not a lot of rules are generated for the amount of time it took to complete the experiment. There are a few explanations for this. Since rules now have more participants, calculation of the ES score and the maximum ES score takes longer. Furthermore, since genes are now associated to many more concepts, rules in interaction experiments have more conjunctions, which take longer to prune in the pruning stages. Therefore, experiment time increases for the same number of rules.

Compared to direct association experiments, the pruning effect is noticeably lower, especially when little rules are generated. This is partially caused by the introduced bias, that restricts the number of rules that can be generated. If an unbiased experiment was feasible, pruning would probably have had a similar efficiency as that of direct association experiments, or even slightly higher since we expect more rules to be generated.

Support-dependent measurements					
Threshold	*#Options*	*#Pruned*	$Ratio_{Pruned}$	*#XPruned*	$Ratio_{XPruned}$
$P = 26$	362,928	361,144	0.995	81	0.0454
$P = 27$	126,888	125,378	0.988	52	0.0344
$P = 28$	56,902	55,565	0.977	45	0.0337
$P = 29$	45,490	44,281	0.973	31	0.0256
$P = 30$	38,089	37,038	0.972	23	0.0219

Table 7.4: Pruning measurements for the interaction association experiments

When evaluating exact pruning, we see that it is monotonically decreasing, as was the case in direct association experiments. Exact pruning with leniency with threshold 0.60 results in about 2% to 4.5% of the rules being pruned, but this is very dependent on the permutations and the leniency threshold. For example, if for $P = 26$ we take leniency threshold 0.50, the pruning ratio would already be 62%, and for leniency threshold 0.40 it was 97%, so depending on how strict the experimenter wants the rules to be ranking dependent, exact pruning can have a greater impact.

Results

When analyzing the rules generated with $P = 26$ and $S = 0.60$, they appear to accurately reflect the themes clustered in the direct association experiments, indicating that there is a lot of activity going on in those parts and processes. Because of high similarity between the rules, we present the best rules of the top three clusters:

```
Rule 1
Support: [69]
Score: 0,951
All genes in the subgroup have interaction with
  a gene that has the following properties:
    molecular_function(iron ion binding),
    bio_proc(regulation of protein modification process),
    cel_comp(mitochondrial membrane),
    bio_proc(regulation of phosphorylation),
    KEGG_pathway(Alzheimer's disease),
    KEGG_pathway(Parkinson's disease),
    KEGG_pathway(Huntington's disease)

Rule 2
Support: [209]
Score: 0,938
All genes in the subgroup have interaction with
  a gene that has the following properties:
    cel_comp(microtubule cytoskeleton),
    cel_comp(cytoplasmic vesicle),
    cel_comp(intracellular membrane-bounded organelle),
    cel_comp(cytoskeletal part),
    KEGG_pathway(Signal Transduction)

Rule 3
Support: [139]
Score: 0,937
All genes in the subgroup have interaction with
  a gene that has the following properties:
    bio_proc(cell projection assembly),
    bio_proc(neurite development),
    cel_comp(intracel. membrane-bounded organelle),
    bio_proc(cell morph. in neuron differentiation),
    KEGG_pathway(Signal Transduction),
    KEGG_pathway(Immune System),
    KEGG_pathway(Focal adhesion),
    KEGG_pathway(Regulation of actin cytoskeleton),
```

As can be seen, scores, rule support and the number of conjunctions are all larger for interaction rules. Themes are very similar to the themes found in the direct association rules, but with more specific terms, such as neurite development, and specifications of mitochondrion and membrane terms.

7.4.3 Comparison

To get an idea of how accurate and powerful the Fantom service is, we compared its output with the output generated by the Database for Annotation, Visualization and Integrated Discovery (DAVID) [HST+07], which is an online analytical tool that can perform a similar function as the Fantom service for direct association experiments. In order to use DAVID, we had to map all identifiers to their ENTREZ identifier, since DAVID seemed to experience problems with the SYMBOL identifiers. Furthermore, DAVID did not allow us to enter more than 3,000 genes in the list, thus we had to restrict ourselves to the top 3,000.

DAVID presents the results in a list of ontological terms, and provides for each term the number of genes in the list associated and not associated to it, and the p-value of the association. Judging from the outcomes of the clusters in DAVID, there is still quite some overlap between them, since mitochondrion appeared in all clusters in the list. Overall, DAVID found terms that were quite straightforward such as mitochondrion, cytoplasmic part, and intracellular membrane-bound organelle. More specific terms that were found by Fantom were also found by DAVID, though lower down the list. What is striking is that the term manganese ion binding, which was in the best rule in Fantom, was not found in DAVID. We suspect this is due to the fact that DAVID solely takes genes itself into account, and not scores to direct the importance of concepts.

7.5 Conclusions and Future Work

In this chapter we discussed Fantom experiments that were performed on microarray expression data obtained from samples taken from two different groups of mice, one group that has cardiac over-expression of the transcription factor TBX3, and a control group that lacks this over-expression. To determine the difference in gene expression between the groups, we calculated absolute t-values to create a ranking, and then performed both direct association experiments and interaction association experiments. For each of these association types we also generated 16 permutations per ranking in order to apply exact pruning. Finally, we presented performance measurements on both experiments, as well as pruning and exact pruning statistics and a reflection on the results.

For direct association experiments, support-dependent execution time behaviour and score-dependent execution time behaviour were much alike. As thresholds would go down, an explosion in execution time would become apparent, which is the result of the scoring function and the score distribution. There are many more genes with low scores, which form rules with lower scores. If we allow for those rules in the rule generation process, the execution time of that process will grow exponentially.

Support-dependent pruning behaviour and score-dependent pruning behaviour were not alike, however. While in both cases the pruning ratio was about 99.5% or

higher, the pruning ratio went down as the support threshold grew higher, but up as the score threshold went higher. Support-dependent pruning was monotonically decreasing because most rules with the top scores were all related, for all the different support thresholds, whereby the rules in the experiments with a lower score threshold are refinements of rules with a higher threshold. Due to the higher support threshold, less rules are generated, yet relatively more rules survive due to the ones with the high score, hence the pruning ratio is lowered a bit. Score-dependent pruning was monotonically increasing because the threshold is in nature a pruning threshold. Increasing it decreases a rule's survival chances in both generation and postprocessing stages.

For both support-dependent measurements and score-dependent measurements, exact pruning seemed very useful, especially at the lower thresholds, where 9–17% of the original output rules could be pruned. Given that this was only the result of 16 permutations, the principle should have a greater impact if more permutations could be generated on a grid of many more machines.

Rules of the direct association experiment seemed to focus on a few topics, discovered by clustering the remainder of the rules. These themes include mitochondrial parts and transmembrane transporter activity, signal transduction, and neurodegenerative diseases. This is more specific and diverse than the themes that DAVID found, since the clusters found there primarily focussed on the mitochondrial part and membrane activity.

For interaction association experiments, we needed to introduce a bias in order to be able to generate experiments with any kind of interesting results, since an unbiased search over all the possible groups proved beyond the capabilities of the Fantom service. In the biased approach, we confined the support threshold to a fixed class, and stated that each rule should have at least a minimum amount of support of the fixed class. In case of our experiments, the fixed class was defined as those genes that have a t-score higher than 2.

Even with the bias in place, rule generation took considerably longer for interaction association experiments, since one third of the number of rules of the direct association approach was generated in roughly 60% of the time. However, the resulting rules all had a score of 0.90 or higher, and thus very characteristic for the TBX3 class.

Given these high numbers of rules with a high score, only support-dependent measurements were taken. These measurements showed that with a little steeper incline the support-dependent execution time behaviour was similar to direct association experiments, though the permutations would cost much less time on average than in the direct association approach, which is the result of the bias.

Conventional pruning behaved in a monotonically decreasing fashion, yet this time with more impact since there were many more rules with higher scores, thus the decrease numbers were in the range of percentage points, instead of tenths of a percentage.

Exact pruning had to be done with some leniency. This leniency stated that if a rule surpassed a leniency score threshold, the related rule would be pruned in the original output. This was a change from the direct association approach, where a related or similar rule would only be pruned if a rule in the permutation output was found with a higher score. Depending on this leniency, more or less rules get pruned. If a leniency of 0.6 was taken, about 5% of the rules was pruned at lower thresholds, while only about 2% would be pruned at a higher support. Decreasing the leniency had a drastic effect; when 0.5 was chosen, 62% was pruned at the lowest threshold, and at 0.4 it was 97%.

For the results, rule support and the number of conjunctions were larger for interaction rules than direct association rules. Themes of both experiment types were very similar, but the results in the interaction association rules experiments were more specific. Since DAVID does not support interaction association in the way the Fantom service does, a direct comparison was not possible.

While the results presented above give new insights in the performance of the Fantom service, it also raised questions. For example, what is an optimal or minimum amount of permutations to be considered for exact pruning? And how can we improve Fantom efficiency for lower thresholds? Finally, how can we make conventional pruning even more rigorous, since many surviving rules are related to each other, and must be clustered to give an overview? These are all topics for future research.

Chapter 8

SNP Experiments: Human Depression

In this chapter we perform experiments on data that was obtained from a Single Nucleotide Polymorphisms (SNP) study done on human depression. We conduct two different experiments; in one experiment we let Fantom mine the SNP rankings directly, and in another data set we let Fantom mine on gene rankings that were extracted from the SNP ranking. We present performance measurements of the Fantom service for both sets as well as pruning and clustering statistics, and a discussion on the best rules generated for each experiment type for each of the different conditions in human depression.

8.1 Introduction

When applying the Fantom service to a ranked list of genes, the result will be a list of specific groups of genes with an interestingness score, and a descriptive rule consisting of conjunctions of GO and KEGG concepts. A different but related experiment would be to mine rules on SNP data instead of genes.

A SNP is a modification within a DNA sequence on a single spot, which might have an effect on one or more genes located near that spot. Depending on the location of the SNP in the DNA string, it can be associated to genes near that location, thus creating a many-to-many mapping between SNPs and genes. This mapping can then be used to establish a mapping between ontology concepts and SNPs.

There are two approaches that can be taken when using the Fantom service on a SNP ranking. The first approach is to use the SNP identifiers as a naming convention, thus mapping SNP identifiers to associated genes, and mediating the SNP scores for each gene to obtain an aggregated gene score. The Fantom service is then executed with the gene ranking obtained from this process. The second option is to create a mapping from SNPs to ontological concepts by combining the SNP to gene mapping

and the gene to ontology mapping. The Fantom service would then be executed on the SNP ranking, using the SNP to ontology mapping.

In this chapter we will perform and analyze both experiment approaches. We will discuss the advantages and disadvantages of each approach and present statistics on the experiments. For each approach, we will perform experiments on different rankings associated with different conditions in human depression. We will then compare the results of those rankings by comparing experiment performance and effectiveness in terms of pruning and clustering, as well as give a brief interpretation of the best rules for each condition.

This chapter is organized as follows. In Section 2, we will present a biological background on SNPs and related terms. We will discuss what SNPs are, what kinds of SNP types there are and why SNP research can be important. In Section 3, we will discuss the SNP study on human depression. We will give an introduction to the subject of human depression, present the studies that were done, and how the SNP data from this study was obtained. In Section 4 we will present the experiments we performed with the Fantom service. We discuss experiment design, and for various rankings we present statistics on performance as well as pruning and clustering for diverse participation and score thresholds. We also compare the rankings with each other to see how they influence aforementioned statistics. Finally, in Section 5, we will draw some conclusions from the experiment results. We make comparisons between both experiment approaches, as well as some overall conclusions on experiment results.

8.2 Single Nucleotide Polymorphisms

A Single Nucleotide Polymorphishm (SNP), pronounced as ”snip”, is a modification of a nucleotide on a DNA strand. These modifications include removal, insertion, or substitution, of which Figure 8.1[1] is an example. The effect of a SNP can be profound. If the information in a gene changes, this might result in the creation of different amino acids, resulting in the formation of different proteins, leading to a change in bodily functions. Furthermore, even if a mutation does not affect a gene directly by modifying a nucleotide that is part of it, it can still influence the activity and translation of genes that are located near the SNP.

SNPs that do not result in a different gene translation are called *synonymous* SNPs, or silent mutations. A SNP that does result in a translational change is called a *nonsynonymous* SNP. There are two kinds of translational changes: a *nonsense* change, which results in an abrupt stop of the gene translation prior to its completion, or a *missense* change, which results in the translation of a gene into a different amino acid.

Human SNP studies are often used to compare one group of participants with

[1] Picture by David Hall, http://en.wikipedia.org/wiki/File:Dna-SNP.svg

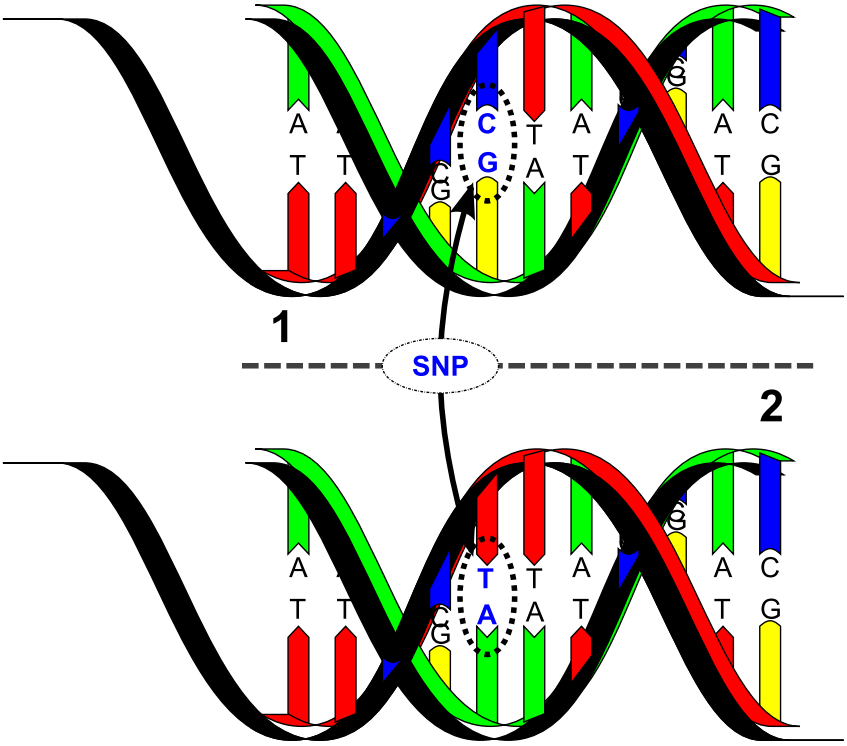


Figure 8.1: Single-Nucleotide Polymorphism in DNA molecules

a certain condition to a matched group of participants that lack that condition. By comparing SNP regions, genes and gene variations can be identified that facilitate or counter certain diseases or conditions. Similarly, SNP can be used in drug research to investigate how certain treatments affect patients. In this case, SNP studies compare patients that received treatment with patients that were untreated, or that were given a placebo.

8.3 SNP Study on Human Depression

Depression is a common mental disorder that presents itself with a depressed mood, loss of interest or pleasure, feelings of guilt or low self-worth, disturbed sleep or appetite, low energy, and poor concentration. These problems can become chronic or recurrent and lead to substantial impairments in an individual's ability to take care of his or her everyday responsibilities[2].

According to the The Tripartite Model of Depression and Anxiety, symptoms of depression and anxiety can be assigned to one of three dimensions, negative affect

[2] http://www.who.int/mental_health/management/depression/definition/en/

(NA), (lack of) positive affect (PA), and somatic arousal (SA) [WvVG+09]. These dimensions are influenced by fewer genes than the diagnosis of major depressive disorders (MDD). Therefore, the effect of genetic factors will be easier to identify. To unravel the genetic component of MDD, we analysed the effects of sets of related genes on NA, PA and SA, as measured by the Mood en Anxiety Symptom Questionnaire (MASQ).

For this, we used data from the Netherlands Study of Depression and Anxiety (NESDA). This is a longitudinal cohort-study meant to identify risk factors for depression and anxiety. For 2,951 individuals, extensive data on symptoms of depression and anxiety, psychology, demographics, lifestyle, and biological measures are collected at baseline, and 1, 2, 4 and 8 years thereafter [PSZ+07]. A genome-wide association study was carried out in a subgroup of NESDA, for 1,860 patients with a Diagnostic and Statistical Manual of Mental Disorders (DSM-IV) diagnosis of MDD [OWS+08]. Genotypes were determined using Perlegen 600k SNP-chips. After quality control of the data, we had 435,276 SNPs for 1,738 patients. MASQ scores for NA, PA and SA were available for 1,543 of these patients.

8.4 Experimental Results

In this section we will present and discuss the results of the various experiments we performed on the human depression SNP study. There are two approaches that we considered in our experiments. In the first approach, we mapped the SNP identifiers to genes, which resulted in a ranking of genes that we experimented on. In the second approach, we experimented on the original ranking of SNP identifiers and their scores, and created a mapping from SNP identifiers to ontological concepts in GO and KEGG.

For each of the approaches, we performed experiments on rankings that were extracted for all three conditions discussed in the previous section: Positive Affect (PA), Negative Affect (NA) and Somatic Arousal (SA). Note that results in this section include primarily performance indications and statistics on pruning and clustering, for interpretation and quality assessment of the rules is beyond the scope of this thesis.

All experiments were conducted with the Fantom service implementation discussed in Chapter 5, and performed on Windows Vista with an Intel Core 2 Duo T6400 2 GHz CPU and 4 GB RAM. The experiment setup also is similar to that of Chapter 7: for each condition, we examined performance, pruning and clustering for support-dependent measurements and score-dependent measurements. After the results of all three conditions have been presented, a comparison will be made to analyze whether there are any significant differences between the three conditions.

The measurements done in each experiment are: the average time for the experiment to complete over 10 runs T_{avg} in minutes m and seconds s, the number of rule options generated *#Options*, the number of options pruned *#Pruned*, the pruning ra-

tio $Ratio_{Pruned}$, the number of clusters derived from the rules with a setting of 60% overlap[3] *#C*, and the rules-per-cluster ratio $Ratio_C$.

We next discuss the experiments on gene translations and the experiments with SNP ontology mappings in two subsections.

8.4.1 Experiments on Gene Translations

In this approach we obtain gene identifier rankings by mapping SNP identifiers to ENTREZ [MOPT05] gene identifiers. This mapping provides a translation from each SNP identifier involved in the SNP study to one or more ENTREZ gene identifiers. For each gene, all score values of associated SNP identifiers are stored, and in the end the median is taken as its score. The workflow is shown in Figure 8.2.

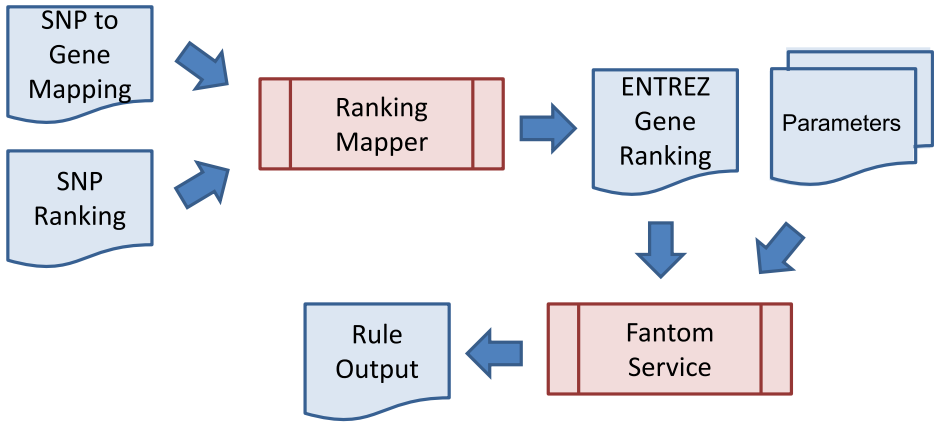


Figure 8.2: The gene translations experiment workflow

The weakness of this method is that gene scores are not accurately measured values, but instead are representatives of a set of SNP scores; aggregating the SNP scores into a single gene score obfuscates the influence of the individual SNPs. The strength of this method is that the experiments can be carried out faster, and at lower thresholds, since the number of SNPs are usually far greater than the number of genes that they are associated with.

After mapping the SNPs to ENTREZ gene identifiers, the ranking size was reduced from $435,290$ SNP identifiers to $15,176$ ENTREZ gene identifiers, whereby 325 SNP identifiers could not be mapped. Compared to the total ranking, this loss was not significant, nor did they have much impact; the 325 identifiers had an average score of 0.80, and only 13 were considered statistically significant, since they had a p-value equal to or less than 0.05.

[3]This threshold was determined on the basis of expert feedback, after several results were presented with various cluster outcomes from different thresholds.

When measuring the impact of support thresholds, the score threshold was set to a constant value, and vice-versa. For the gene translation experiments, the fixed score threshold was set to 0.60, while the fixed support threshold was kept constant at 15 genes. In each of the following sections, we present the results in a combined table, accompanied by explanations on their behaviour. A comparison between the results in these tables will take place in Section 8.4.1.

Negative Affect

Table 8.1 shows the support-dependent measurements and score-dependent measurements for the NA condition. What becomes immediately apparent is difference in

Support-dependent measurements						
Threshold	T_{avg}	*#Options*	*#Pruned*	$Ratio_{Pruned}$	*#C*	$Ratio_C$
$P = 10$	64m51s	862,728	862,073	0.999241	229	2.8602
$P = 11$	17m42s	491,533	491,126	0.999172	165	2.4667
$P = 15$	9m33s	149,890	149,852	0.999566	40	1.625
$P = 20$	7m36s	69,983	69,977	0.999914	5	1.2
$P = 25$	7m09s	47,314	47,312	0.999958	2	1

Score-dependent measurements						
Threshold	T_{avg}	*#Options*	*#Pruned*	$Ratio_{Pruned}$	*#C*	$Ratio_C$
$S = 0.50$	18m26s	263,816	262,768	0.996028	339	3.0914
$S = 0.55$	13m06s	221,786	221,414	0.998323	135	2.7555
$S = 0.60$	9m33s	149,890	149,852	0.999566	40	1.625
$S = 0.65$	7m43s	104,954	104,941	0.999876	12	1.0833

Table 8.1: Performance measurements for the NA condition

running times between $P = 10$ and $P = 11$, which seems to be a breakpoint. Apparently there are many more subgroups that are shared by 10 genes or less than by 11 or more. Also, pruning experiences a local minimum around that breakpoint, indicating an inverse relationship between those two properties, whereby the point close after the breakpoint forms an optimal support threshold for maximized interestingness of the rules.

For score-dependent measurements, running times degrade much more gradual and linear than in the support-dependent measurements within the given score-boundaries. This is expected, since there are less rules with these high scores, and thus a sudden change like in support-dependent measurements is expected only at lower scores. As expected of a pruning threshold, the pruning ratio is monotonically increasing if the threshold goes up. For both score-dependent and support dependent measurements, the rules-per-cluster ratio is monotonically decreasing.

When analyzing the rules generated for the NA condition, there were some surprises in terms mixed with terms known to be associated with depression. We present the best three rules below. Once again, since the data set is not public, the genes involved in the rules are removed from the output:

```
Rule 1
Support: [11]
Score: 0,747
All genes in the subgroup
  have the following properties:
    bio_proc(cholesterol biosynthetic process),
    cel_comp(intracellular membrane-bounded organelle),
    KEGG_pathway(Lipid Metabolism)

Rule 2
Support: [12]
Score: 0,744
All genes in the subgroup
  have the following properties:
    bio_proc(pos. reg. of protein kinase cascade),
    bio_proc(reg. of I-kappaB kinase/NF-kappaB cascade),
    cell_comp(intracel. non-membrane-bounded organelle)

Rule 3
Support: [13]
Score: 0,722
All genes in the subgroup
  have the following properties:
    bio_proc(regulation of transcription),
    KEGG_pathway(Pancreatic cancer),
    KEGG_pathway(Chronic myeloid leukemia)
```

The rules shown above are concerned primarily with metabolism, which is known to be affected by stress. The second rule contains concepts that are all related to infections, since depression influences responses to infection. Finally, the last rule concerns signalling constructs within the body, which are also affected by depression. Leukemia is among these terms, since leukemia also is caused by signalling deficiencies.

Positive Affect

The results for support-dependent measurements and score-dependent measurements for the PA condition are shown in Table 8.2. As with the NA condition, the PA con-

Support-dependent measurements						
Threshold	T_{avg}	*#Options*	*#Pruned*	$Ratio_{Pruned}$	*#C*	$Ratio_C$
$P = 09$	83m38s	1,977,442	1,975,733	0,99914	628	2.7229
$P = 10$	35m13s	911,444	910,375	0.99883	377	2,8355
$P = 15$	9m34s	157,889	157,715	0.99890	72	2.4167
$P = 20$	9m00s	84,841	84,816	0.99971	16	1.5625
$P = 25$	6m50s	40,191	40,186	0.99988	4	1.25

Score-dependent measurements						
Threshold	T_{avg}	*#Options*	*#Pruned*	$Ratio_{Pruned}$	*#C*	$Ratio_C$
$S = 0.50$	13m58s	252,727	251,251	0.99416	464	3.1810
$S = 0.55$	12m39	184,283	183,672	0,99668	203	3,0099
$S = 0.60$	9m34s	157,889	157,715	0.99890	72	2.4167
$S = 0.65$	9m20s	130,540	130,505	0.99973	16	2.1875

Table 8.2: Performance measurements for the PA condition

dition also shows a breakpoint, at $P = 9$. Furthermore, there seems to be a sort of processing plateau between $P = 15$ and $P = 20$. The reason for this is because the rules do not differ a lot between these two thresholds, both in dimensionality and content they are closely related or even the same. That means that almost the same amount of time is spent on pruning, while calculation of the maximum ES score takes more time, since now for each rule option 20 maxima have to be calculated instead of 15.

Pruning seems to converge to a local minimum again, in this instance near $P = 10$, after the breakpoint. Clustering seems to be erratic, but still overall decreasing. We will explain the reason for this erratic behaviour in Section 8.4.1.

Score-dependent measurements behave as expected. Similar to the NA condition, we see a more linear correlation between running times and scores for the given score thresholds, for the same reasons. Pruning and clustering also behave in a similar way.

The best 3 rules in the output of the Fantom service for the PA condition is shown below.

```
Rule 1
Support: [10]
Score: 0,82835619871135
All genes in the subgroup
have the following properties:
bio_proc(protein amino acid phosphorylation),
bio_proc(peptidyl-amino acid modification),
bio_proc(pos. regulation of protein kinase activity)
```

```
Rule 2
Support: [10]
Score: 0,820677828269983
All genes in the subgroup
  have the following properties:
    bio_proc(negative regulation of transcription),
    bio_proc(post-translational protein modification),
    cellular_component(nucleoplasm part)

Rule 3
Support: [10]
Score: 0,820
All genes in the subgroup
  have the following properties:
    bio_proc(neg. reg. of macromol. biosynth. process),
    bio_proc(neg. reg. of gene expression),
    bio_proc(neg. reg. of cellular biosynth. process),
    bio_proc(post-translational protein modification),
    cellular_component(nucleoplasm part),
    bio_proc(regulation of transcription),
    bio_proc(neg. reg. of nucl. acid metabolic process)
```

For the PA condition scores are very high, and the differential part largely lies in the biological process domain. This is confirmed by the clustering; the three biggest clusters deal with positive and negative regulation of diverse processes, specifically in the nucleoplasm part.

Somatic Arousal

The results for support-dependent measurements and score-dependent measurements for the PA condition are shown in Table 8.3. The first observation in Table 8.3 is that experiments on the SA condition do not take nearly as much time as the other conditions, given the same thresholds. Even when choosing a low support threshold, it still results in a relatively low number of rules. As a result, rules-per-cluster ratios are also rather low. Pruning is on a similar level as the other conditions, indicating that pruning is not directly dependent on the ranking.

Apart from shorter experiment times, the score-dependent measurements for the SA condition show the same behaviour as the score-dependent measurements in other conditions; pruning ratio, the number of cluster and cluster ratio all show the same decline, although the cluster ratio is lower due to the smaller number of rules generated.

The best 3 rules in the output of the Fantom service for the PA condition is shown

Support-dependent measurements						
Threshold	T_{avg}	*#Options*	*#Pruned*	$Ratio_{Pruned}$	*#C*	$Ratio_C$
$P = 07$	45m25s	1,255,501	1,253,358	0.99829	857	2.5006
$P = 10$	16m20s	443,470	442,992	0.99892	209	2.2871
$P = 15$	7m59s	106,517	106,468	0.99954	28	1.75
$P = 20$	6m48s	62,277	62,264	0.99979	8	1.625
$P = 25$	5m42	30,497	30,496	0,99997	1	1

Score-dependent measurements						
Threshold	T_{avg}	*#Options*	*#Pruned*	$Ratio_{Pruned}$	*#C*	$Ratio_C$
$S = 0.50$	9m01s	167,250	166,603	0.99613	301	2.1495
$S = 0.55$	8m21s	134,242	134,056	0,99861	91	2.044
$S = 0.60$	7m59s	106,517	106,468	0.99954	28	1.75
$S = 0.65$	7m08s	95,029	95,024	0.99995	5	1

Table 8.3: Performance measurements for the SA condition

below. When looking at the rules of the SA condition, we see that they are closely related to the PA condition in terms of concepts.

```
Rule 1
Support: [11]
Score: 0,830
Participants: [
All genes in the subgroup
  have the following properties:
    bio_proc(reg. of trans. from RNA polym.II promoter),
    bio_proc(pos. reg. of cellular biosynth. process),
    cellular_component(nuclear part),
    bio_proc(pos. reg. of RNA metabolic process),
    cellular_component(intracellular organelle lumen)

Rule 2
Support: [10]
Score: 0,818
All genes in the subgroup
  have the following properties:
    bio_proc(pos. reg. of gene-specific transcription),
    cellular_component(nuclear part),
    bio_proc(reg. of trans. from RNA polym.II promoter),
    cellular_component(intracellular organelle lumen)
```

```
Rule 3
Support: [12]
Score: 0,788
All genes in the subgroup
  have the following properties:
    bio_proc(reg. of spec. from RNA polym.II promoter),
    cellular_component(nuclear part)
```

Where in the PA condition negative regulation of processes was predominant, in the SA condition those same processes seem to be positively regulated, yet within both conditions they do not differ much in score. The parts where these processes occur are confined to the nuclear part, as well as the intracellular organelle lumen.

Discussion

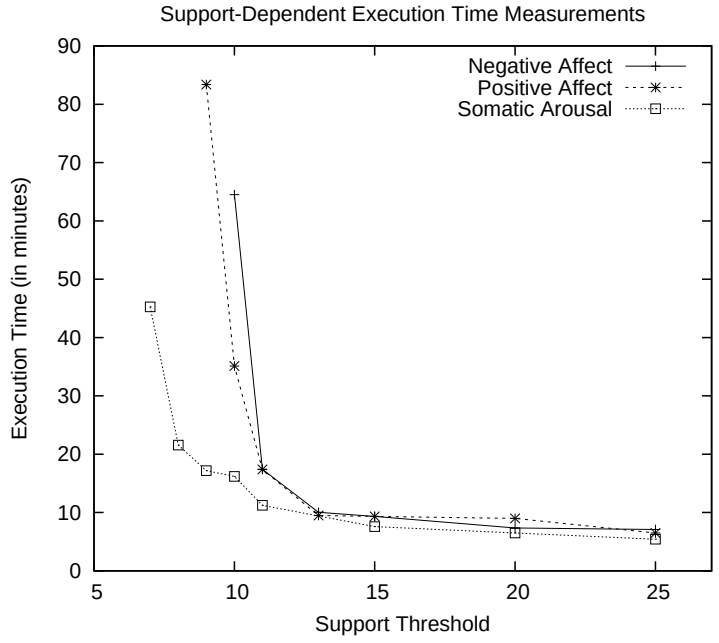
In this part we will discuss the measurements presented above. We will discuss three dimensions: execution time, pruning ratio and rule-per-cluster ratios. We discuss execution time to establish a global indication of the performance behaviour. Pruning and rule-per-cluster ratios are discussed to get an indication of what behaviour these algorithms show within the Fantom algorithm.

Execution Time As can be seen in Figure 8.3(a), all conditions have roughly the same curve, growing at an explosive rate as the support threshold becomes lower. The breakpoint varies for each condition. For the NA condition, it is at $P = 10$, for PA at point $P = 9$, and for SA at point $P = 7$, though in this case the incline is less abrupt.

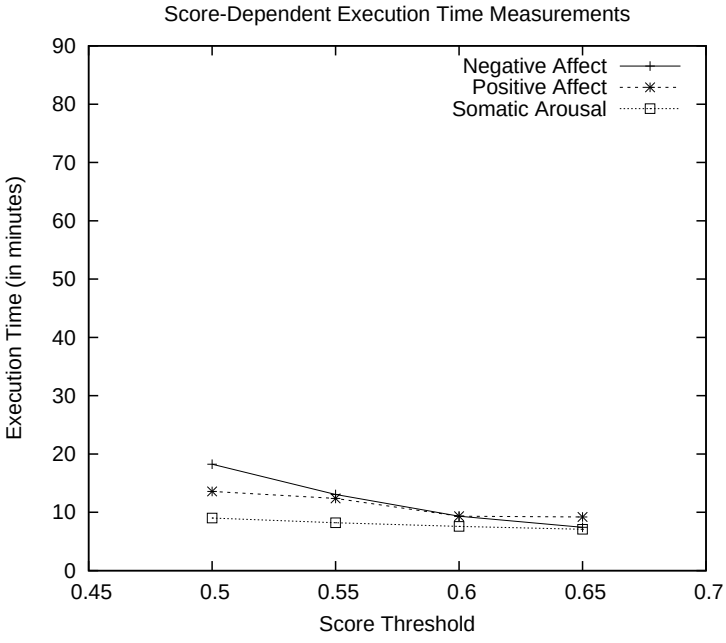
The breakpoint seems to be ranking dependent: different rankings produce rules with different scores. For SA, most subgroups that have more than 7 participants do not seem to get a score or even maximum score higher than the fixed $S = 0.6$, while for NA and PA there are. Since more rules and rule options adhere to these boundaries, the experiment takes longer, for more combination, calculation and pruning operations have to be performed.

When analyzing the relationship between score thresholds and experiment time, as depicted in Figure 8.3(b), we see that the influence is far less profound, though still noticeable. The correlation seems to be somewhat linear, since points can be connected with a fairly straight line. This linear correlation thus indicates that for scores between 0.50 and 0.65, subgroups with these scores and maximum scores are evenly distributed. Differences in influences between the conditions can partially be related to the number of rules generated; when more rules are generated, more will be pruned whenever the threshold goes up, hence the impact in execution time will be longer.

The fact that the score threshold influences the experimental time to a lesser extend is because it requires more effort to calculate. When two rule options are com-



(a)



(b)

Figure 8.3: Execution time measurements for the experiments on gene translations

bined, and the rule option support is below the threshold, it can be discarded immediately, without the need for calculation of its ES and maximum ES. However, to determine if the maximum ES of a rule option is below par, it has to be calculated first, taking a substantial amount of extra time. Therefore, the score threshold only saves on the pruning part of the rule generation, and not the score calculation.

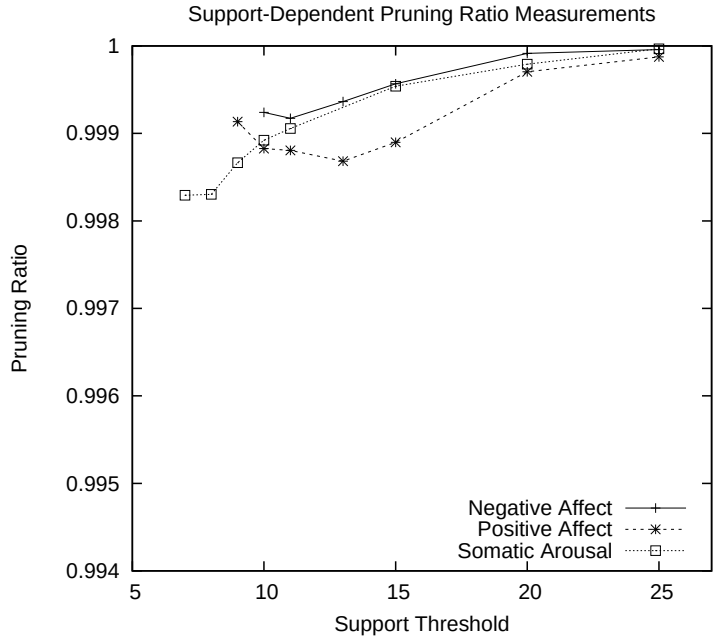
From a practical point of view, analyzing lower score thresholds was not necessary. The lowest score usually already generates between 500–1,000 rules or more, which is unacceptable to review from a bio-informatician's point of view. We do see a sudden rise in execution time for lower score thresholds, since most rules have a lower score, as we will see in the SNP ranking experiments.

Pruning The pruning ratio is the number of pruned rule options divided by the total number of options generated. As can be seen in Figure 8.4(a), there is a connection between the pruning ratio and the breakpoints of the support thresholds; the breakpoints appear to be local minima, or close to local minima, in the pruning ratio. This implies that the breakpoints form some sort of optimal trade-off between specificity of the rules and support.

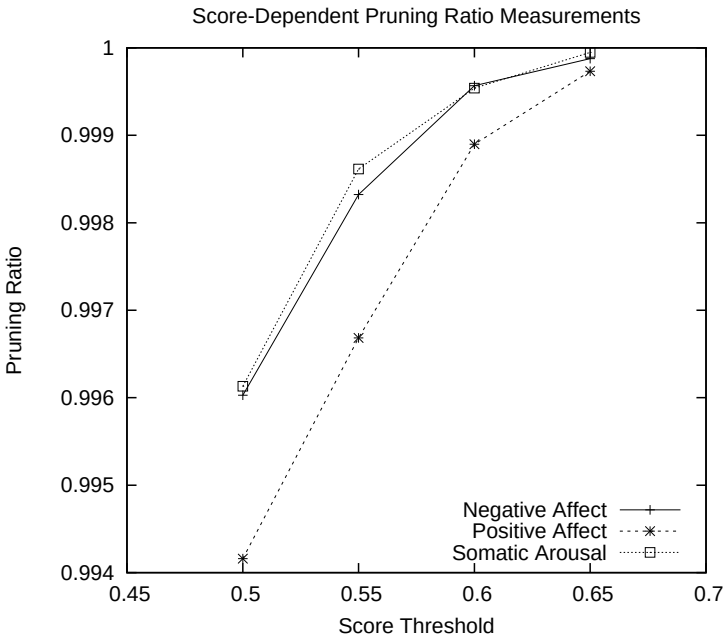
Before the breakpoint, relatively more rule options are pruned because one rule with very specific concepts can invalidate all related rule options with less specific terms, which, depending on the depth of the term in the ontology as well as the upward connectivity of terms (how many parents they have), can be many. After the breakpoint, relatively more rule options are pruned because they can no longer meet the support threshold. By finding the lowest pruning threshold, a fairly good balance between rule specificity and rule support can be obtained.

In contrast, the effect of the score threshold is much more profound on the pruning ratio, as can be seen in Figure 8.4(b). This is because the score threshold works both as an intermediate option pruner *and* as a postprocessing rule pruner. If the score threshold is increased, more options are pruned while generating rule options, since now the maximum ES score of more rule options does not suffice anymore. Moreover, even if a rule option did have the required maximum ES score, it does not guarantee its actual ES score suffices. Since the threshold has become stricter, this increases the likelihood of the rule to be pruned in the post-processing stage.

The effect of increasing the pruning threshold is more significant for lower scores, which is the result of the gene ranking score distribution, shown in Figure 8.5.
As can be seen, all three rankings have almost identical distributions, where most genes have very low scores, and thus a lower impact on the ES score compared to the higher genes. This means that compared to the total rule collection, a higher percentage of the rule options resides in the lower section of the ranking. Consequently, any change made to the pruning ratio in the lower spectrum has more impact than changes made in a higher spectrum.



(a)



(b)

Figure 8.4: Pruning ratio measurements for the experiments on gene translations

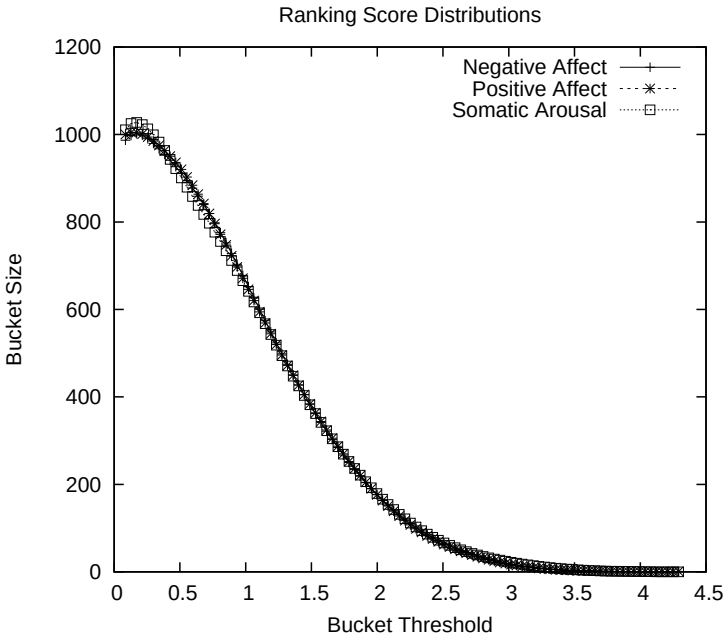


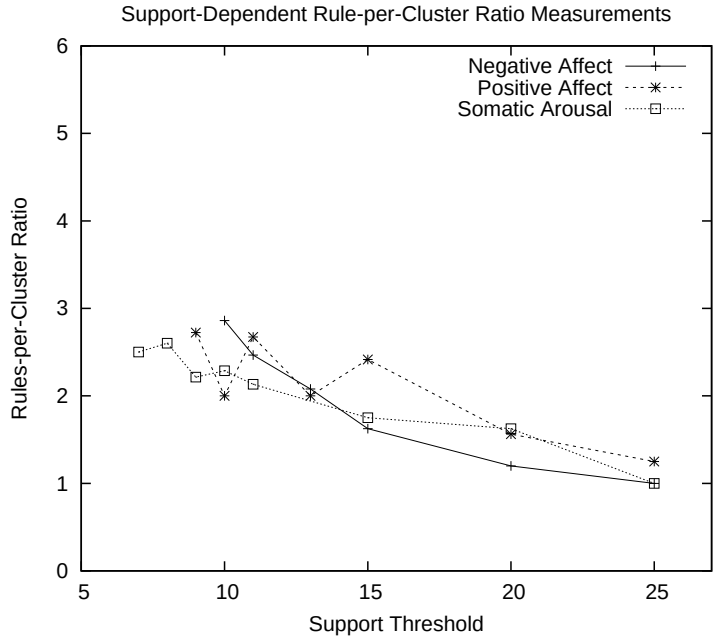
Figure 8.5: Ranking score distributions of all conditions

Clustering When analyzing the relationship between support thresholds and rules-per-cluster ratios, shown in Figure 8.6(a), it seems almost linear. However, measurements seem to get more erratic as the group size gets smaller. This is a direct effect of a small group size and the cluster threshold, since it can cause a large discrepancy error.

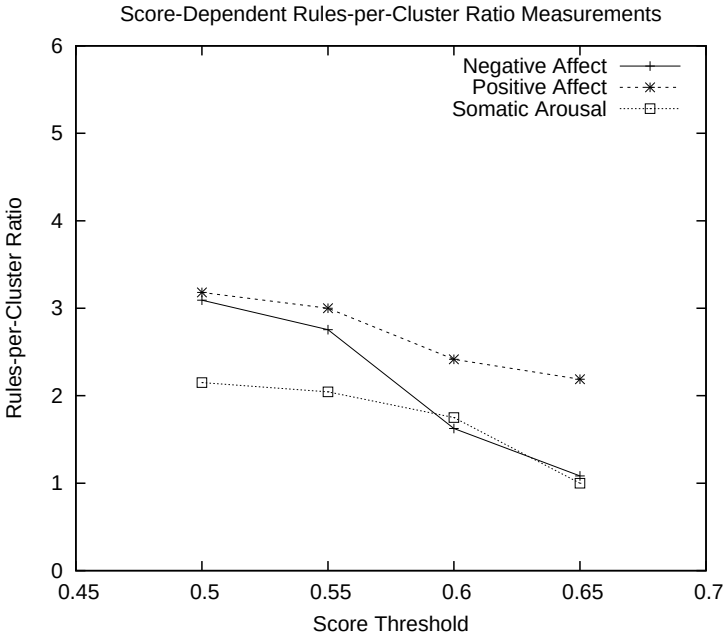
Consider the following example: When comparing two rules that both have 6 genes while clustering at a threshold of 0.6, these two rules need to have 4 genes in common, or 66.67%, a discrepancy of 6.67 percentage points with respect to the original threshold of 0.6. In contrast, when comparing two rules that have 16 genes, they need to have 10 genes in common, or 62.5%, which only has a discrepancy error of 2.5 percentage points. As a rule, the lower the discrepancy error becomes, the more stable and accurate clusters are.

The relationship between score threshold and rules-per-cluster ratio shows sudden drops. These drops are the properties of the rules that come with the specific ranking. Since clustering is based on similarity of genes, most rules clusters will have roughly the same ES score. When the score threshold moves past a certain threshold, most of the rules in a cluster will vanish, except for the few with a higher score. The sudden removal of may rules while retaining a cluster causes the rule-per-cluster ratio to drop suddenly.

Another explanation for the drops is that we observed that the rules with the higher score seem to appear in small clusters, usually only one or two rules per clus-



(a)



(b)

Figure 8.6: Cluster ratio measurements for the experiments on gene translations

ter, while rules with a lower score tend to form bigger clusters, since there are more rules in that range. That means that increasing the score threshold in this range is likely to reduce the rule-to-cluster ratio more.

Rules When comparing the rules of the three conditions, both PA and SA are closely related in concepts, yet the opposite seems to be happening; in PA, processes are negatively regulated, where in SA those same processes are positively regulated. The NA condition is involved in regulation too, yet terms there are very directed and specific, and even include associations with certain cancer types.

8.4.2 Experiments with SNP Ontology Mappings

Instead of mapping SNPs to genes and aggregating their scores, a second approach would be to leave the ranking as is, using SNP identifiers and their individual scores, and let the Fantom service use a mapping between SNP identifiers and the GO and KEGG ontologies. The mapping is a combination of the SNP to ENTREZ mapping and the ENTREZ to GO and ENTREZ to KEGG mappings discussed earlier, resulting in a list of SNP identifiers and the GO and KEGG concepts they are associated with. The workflow is shown in Figure 8.7.

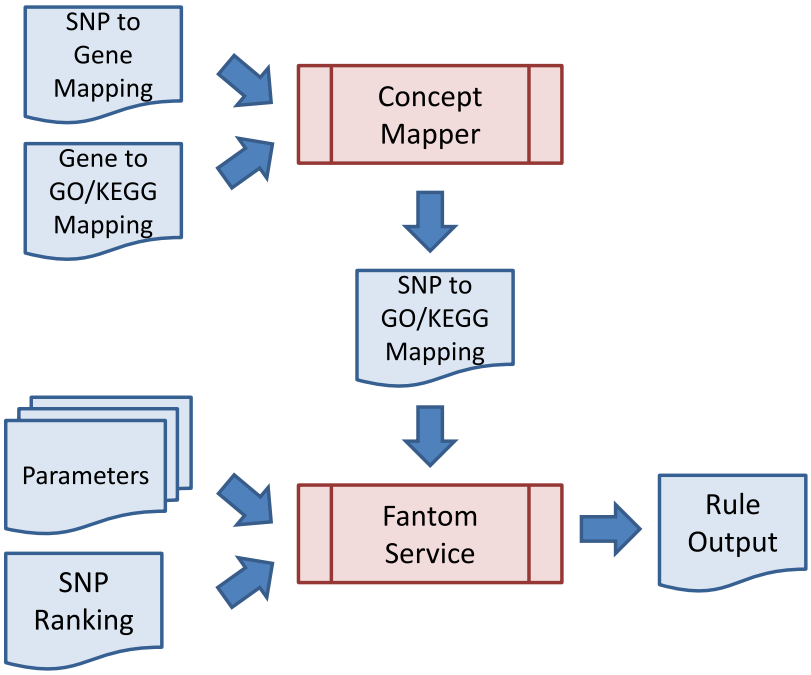


Figure 8.7: The ontology mappings experiment workflow

This method has its strengths and weaknesses. When keeping the individual SNPs and their scores, rules become more accurate, as do rule scores, since individual SNPs are now associated instead of their related gene aggregates. Consequently, rules are tailored to SNP subsets that are differentially expressive, not gene subsets, giving an accurate description of the SNP properties, and a more accurate score based on the SNP scores. When mapping to genes, and then mapping back in the output, this cannot be done, due to the many-to-many relationship between genes and SNPs.

A weakness of this method is the larger number of identifiers. When using gene translation, the set of $435,290$ SNP identifiers was translated to $15,176$ ENTREZ gene identifiers, which allowed Fantom to search a larger rule space. When using $435,290$ SNP identifiers, loading takes a long time. Furthermore, running the Fantom service would require both support and score thresholds to be very high, making it very unlikely to obtain subgroups with a high interestingness score using the ES measurement. When running the Fantom service on the SA ranking, for example, the experiment took over seven hours, and produced only two rules, both very low in score and deemed uninteresting by the expert analysis.

Despite the large amount of identifiers, we can still use the SNP ranking to do a multi-class study. In this particular experiment, for each of the three conditions we made a ranking of all SNPs with a p-value of less than 0.05, which were all SNPs with a t-value of higher than 1.962. Next, for each of the three conditions we created a ranking that contained only SNP identifiers that had a higher score than in the other two rankings and labeled these as the interest class, while labeling the residu as the control class. We performed the multi-class experiment on the resulting ranking.

To determine thresholds for comparison, we first analyzed the data set with a quick run of the Fantom algorithm, restricting the rules in the maximum amount of conjunctions as we did in Chapter 6. Finally, we found $P = 15$ and $S = 0.48$ suitable fixed values to test performance, clustering and pruning, since all three rankings showed interesting results on tests with these fixed values.

Negative Affect

In this condition there were $223,554$ SNPs statistically significant. Filtering with the other two condition lists resulted in $11,746$ SNP identifiers in the interest class. The residual of the SNP identifiers from the rankings of the other two conditions were merged, whereby the highest score of a SNP was taken as its score in the merged lists. This resulted in $39,793$ SNPs in the control class. Results of the support-dependent and score-dependent measurements are shown in Table 8.4.

Scores of rules are significantly lower in this kind of experimental setup. The ranking scores are much closer together, forcing rule and rule option scores to have a lower average. Furthermore, the control group penalty adjustment lowers scores even more, thus the boundary had to be set from $S = 0.60$ to $S = 0.48$. The trend is the same as in the other experiments, though the rule space is much more compacted: the drop in

Support-dependent measurements						
Threshold	T_{avg}	*#Options*	*#Pruned*	$Ratio_{Pruned}$	*#C*	$Ratio_C$
$P = 10$	48m01s	1,119,855	1,119,823	0.99997	14	2.29
$P = 11$	8m48s	65,822	65,801	0.99968	10	2.10
$P = 12$	8m23s	34,054	34,039	0.99956	8	1.88
$P = 13$	8m19s	26,909	26,895	0.99948	7	2.00
$P = 15$	8m09s	21,345	21,340	0.99977	2	2.50

Score-dependent measurements						
Threshold	T_{avg}	*#Options*	*#Pruned*	$Ratio_{Pruned}$	*#C*	$Ratio_C$
$S = 0.24$	51m26s	1,116,328	1,116,109	0.99980	100	2.19
$S = 0.25$	8m58s	64,756	64,563	0.99702	91	2.12
$S = 0.35$	8m32s	40,457	40,425	0.99921	16	2.00
$S = 0.40$	8m21s	26,301	26,289	0.99954	7	1.71
$S = 0.48$	8m09s	21,345	21,340	0.99977	2	2.50

Table 8.4: Contrast experiment measurements on the NA condition

execution time and amount of generated rules is very large and sudden, meaning that for $S = 0.48$, $P = 11$ seems to be a barrier.

We also see the same trend in pruning ratio, where it starts high, moves to a local minimum, and then moves upward again. The trend is more crude this time, since rule generation was lower, thus creating a less accurate picture. Rule-per-cluster statistics seem to be inconclusive. At some point, the statistic rises again. This could partially be because some cluster themes contain rules with high scores, but also because there are very few rules and clusters, thus a larger discrepancy error.

Since scores are now less varied, the score-dependent measurements show a strong resemblance to support-dependent data. Both execution times and pruning ratio show the same kind of behaviour, indicating the breakpoint for $P = 15$ is set somewhere between $S = 0.24$ and $S = 0.25$. The pruning ratio also shows the local minimum after the breakpoint, which is, as expected, lower than the one that shows in the support-dependent measurement. Rule-per-cluster statistics are overall decreasing, but for the last entry. The 3 best rules for the settings $P = 10$ and $S = 0.48$ are shown below.

```
Rule 1
Support: 65
Score: 0,569
All genes in the subgroup
have the following properties:
mol_func(phosphatidylinositol phosph. activ.)
```

```
Rule 2
Support: 44
Score: 0,568
All genes in the subgroup
have the following properties:
cellular_component(nonmotile primary cilium),
biological_process(photoreceptor cell maintenance)

Rule 3
Support: 53
Score: 0,560
All genes in the subgroup
have the following properties:
molecular_function(ATP binding),
cellular_component(membrane fraction),
mol_func(P-P-bond-hydrolysis-driven  transp. activ.),
cellular_component(integral to membrane),
mol_func(ATPase activ., coupled to movem. of substances)
```

As can be seen in these rules, the scores are noticeably lower than the scores in the NA experiments of the gene translation method. The conjunctions, on the other hand, are larger, and the concepts used in the rules are much more specific. Rules of the NA condition shown here are related to the rules in the NA condition of the gene translation method in terms of kinase, phosphatase and energy bindings. Within the top 10 rules, the earlier reported association with cancer was also still present with a reasonable score.

Positive Affect

In the PA condition, $21,544$ SNPs were statistically significant which, after filtering, resulted in a ranking of $13,287$ SNP identifiers in the interest class. The control class consisted of $38,252$ SNPs after filtering. Results of the support-dependent and score-dependent measurements are shown in Table 8.5 Although the execution time shows a familiar pattern, both the pruning ratio and the rules-per-cluster ratio deviate from support-dependent behaviour we have seen so far. When inspecting the rules of the diverse support thresholds, a moderate-sized group of related rules consistently scored very high. As the total number of rules gets lower, this influences the pruning statistics as well as the rules-per-cluster statistics to a great extend, increasing the pruning ratio as the support threshold increases.

Score-dependent measurements are also showing different behaviour. Even for very low score thresholds, a sudden increase in generated rules and execution time does not occur. This has to do with the participation threshold. As the participation

Support-dependent measurements						
Threshold	T_{avg}	*#Options*	*#Pruned*	$Ratio_{Pruned}$	*#C*	$Ratio_{C}$
$P = 07$	69m12s	1,644,519	1,643,917	0.99963	212	2.8396
$P = 08$	12m24s	279,366	279,212	0.99945	59	2.6012
$P = 10$	8m24s	67,453	67,362	0.99865	43	2.1163
$P = 13$	8m03s	38,396	38,350	0.99880	17	2.7059
$P = 15$	7m48s	23,906	23,867	0.99837	14	2.7857

Score-dependent measurements						
Threshold	T_{avg}	*#Options*	*#Pruned*	$Ratio_{Pruned}$	*#C*	$Ratio_{C}$
$S = 0.10$	10m23s	89,159	87,933	0.98625	451	2.7184
$S = 0.25$	9m15s	60,810	60,404	0.99332	178	2.2809
$S = 0.35$	8m47s	53,756	53,628	0.99762	61	2.0984
$S = 0.40$	8m32s	39,596	39,529	0.99831	33	2.0303
$S = 0.48$	7m48s	23,906	23,867	0.99837	14	2.7857

Table 8.5: Contrast experiment measurements on the PA condition

thresholds get higher, the sudden increase is pushed back to a lower score threshold and eventually flattens out, since the number of rule options that support the thresholds becomes smaller.

As expected, pruning ratio is monotonically increasing as the score threshold increases. Rules-per-cluster ratio shows the same behaviour as in the support-dependent measurements. The resulting best 3 rules for the settings $P = 07$ and $S = 0.48$ are shown below.

```
Rule 1
Support: [13]
Score: 0,853
All genes in the subgroup
  have the following properties:
    bio_proc(MAPKKK cascade),
    bio_proc(protein amino acid phosphorylation),
    molecular_function(zinc ion binding),
    bio_proc(pos. reg. of cel. protein metabolic proc.),
    bio_proc(regulation of MAP kinase activity),
    bio_proc(pos. reg. of protein kinase activity)
```

```
Rule 2
Support: [8]
Score: 0,851
All genes in the subgroup
  have the following properties:
    bio_proc(activation of MAPK activity),
    molecular_function(zinc ion binding),
    bio_proc(pos. reg. of cell. protein metabolic proc.)

Rule 3
Support: [8]
Score: 0,851
All genes in the subgroup
  have the following properties:
    bio_proc(transf. growth fact. beta rec. sig. path),
    molecular_function(zinc ion binding),
    bio_proc(regulation of cytoskeleton organization)
```

Similar to the gene translation method, positive regulation of protein kinase activity has a high score again, even in these condensed contrast studies. Zinc ion binding also seems to be very important to the PA condition, as opposed to other conditions. Cytoskeleton organization is an unexpected concept, but even in the clustering it stands out, together with cytosol as cellular component. Clustering also promotes the protein kinase and zinc ion binding as the most important concepts, together with the regulation of metabolic processes.

Somatic Arousal

The SA condition contained $22,071$ statistically significant SNP identifiers. Through filtering these were reduced to a ranking of $14,498$ SNPs in the interest class. The control class consisted of $37,041$ SNPs. Results of the support-dependent and score-dependent measurements are shown in Table 8.6.

For support-dependent measurements, execution time and pruning ratio show familiar patterns, although the drop in execution time is more extreme than in the other conditions. The pattern in the rules-per-cluster ratio is also familiar, but what is surprising is the large number. After inspecting the rules and clusters, the top 10 rules always appear in the top 3 largest clusters, which explains why they do not get pruned, and the rules-per-cluster number stays high.

As a result of the $P = 15$ fixed support threshold, the SA condition lacks a breakpoint in the score-dependent measurements. Similar to the PA condition, the pruning ratio is monotonically increasing, but so is clustering; due to the fact that the rules with the highest score are also in the largest clusters, these clusters remain large as

Support-dependent measurements						
Threshold	T_{avg}	*#Options*	*#Pruned*	$Ratio_{Pruned}$	*#C*	$Ratio_C$
$P = 14$	83m12s	1,113,230	1,111,733	0.99866	326	4.59
$P = 15$	8m35s	43,277	43,190	0.99799	9	9.67
$P = 16$	8m25s	37,203	37,153	0.99866	8	6.25
$P = 17$	8m03s	23,443	23,430	0.99945	5	2.60
$P = 18$	7m48s	22,648	22,639	0.99960	3	3.00

Score-dependent measurements						
Threshold	T_{avg}	*#Options*	*#Pruned*	$Ratio_{Pruned}$	*#C*	$Ratio_C$
$S = 0.10$	11m29s	91,950	90,768	0.98715	403	2.93
$S = 0.20$	9m48s	76,200	75,509	0.99093	247	2.80
$S = 0.30$	9m18s	69,124	68,801	0.99533	86	3.76
$S = 0.40$	8m51s	48,640	48,487	0.99685	30	5.10
$S = 0.48$	8m35s	43,277	43,190	0.99799	9	9.67

Table 8.6: Contrast experiment measurements on the SA condition

the score threshold increases, thus increasing the rules-per-cluster ratio. The three best rules for the settings $P = 15$ and $S = 0.48$ are shown below.

```
Rule 1
Support: [51]
Score: 0,600
All genes in the subgroup
  have the following properties:
    bio_proc(sensory perception of sound),
    cellular_component(axon),
    bio_proc(mechanoreceptor differentiation),
    bio_proc(inner ear development)

Rule 2
Support: [48]
Score: 0,599
All genes in the subgroup
  have the following properties:
    bio_proc(cellular metal ion homeostasis),
    bio_proc(sensory perception of sound),
    bio_proc(regulation of action potential in neuron),
    bio_proc(regulation of membrane potential)
```

```
Rule 3
Support: [48]
Score: 0,599
All genes in the subgroup
  have the following properties:
    bio_proc(sensory perception of sound),
    bio_proc(regulation of action potential in neuron),
    bio_proc(cellular cation homeostasis),
    bio_proc(regulation of membrane potential),
    bio_proc(metal ion homeostasis)
```

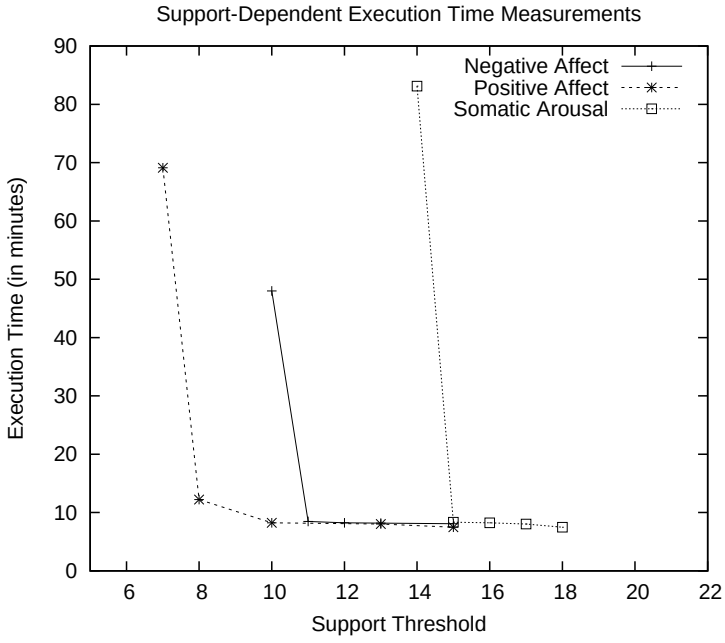
What is surprising about these rules is that they are not very similar to the rules in the gene translation process. Part of that can be explained by the experiment design; many SNPs were removed from the list because the SNPs in the PA condition had a higher expression level. This is the reason that all concepts shared with the PA condition are now less noticeable. What remains are rules that are specific to the SA condition. The scores are therefore not as high, but the rules are very specific. They all still contain involvement of the membrane, but also contain sensory perception of sound and inner ear development as a primary concept. When clustered, these concepts appear in the top cluster as well, together with neurotransport activity.

Discussion

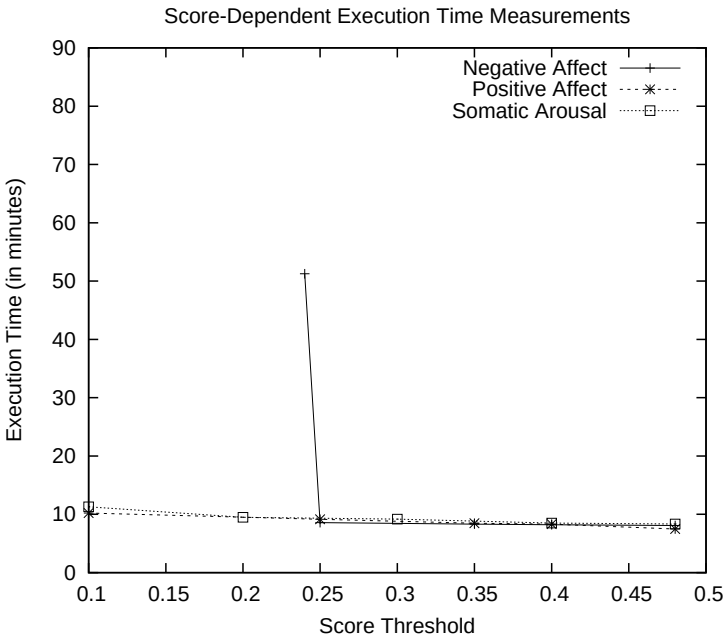
Similar to Section 8.4.1, we will again discuss the measurements by comparing them in three dimensions: Execution time, pruning ratio and rule-per-cluster ratio.

Execution time As was the case in the single-class experiments, multi-class experiments also show a breakpoint in support-dependent measurements. Its location is again dependent on the ranking, as can be seen in Figure 8.8(a). However, there is a difference between the behaviours. The decline is much steeper in the multi-class, and the variation in the positions of the decline is also larger. This is caused by the many-to-many relationship between SNPs and genes. SNP identifiers can be associated to many genes, thus many ENTREZ and KEGG concepts. Furthermore, one gene can be associated to multiple SNP identifiers, associating all those identifiers with the same ontological concepts. As a result, under a specific support threshold, all these identifiers will share a group, thus causing an explosion in the rule space.

What is also striking is that the score-dependent measurements of the NA condition, displayed in Figure 8.8(b), also show such a breakpoint. The explanation is similar to the one of the support-dependent measurements: using SNP identifiers instead of genes resulted in an explosion in the rule space. Furthermore, the multi-class adaptation of the ES function folds the rule space instead of spreading rules more equally, concentrating the number of rules even more to the lower boundaries. As a result,



(a)



(b)

Figure 8.8: Performance measurements of the ontology mappings experiment

many rule options at support threshold $P = 15$ have a maximum score $S = 0.24$. The other two conditions lack this breakpoint because the $P = 15$ support threshold is too strict; most combinations will never have that support, and therefore execution times will only increase slightly, even for score thresholds as low as $S = 0.10$.

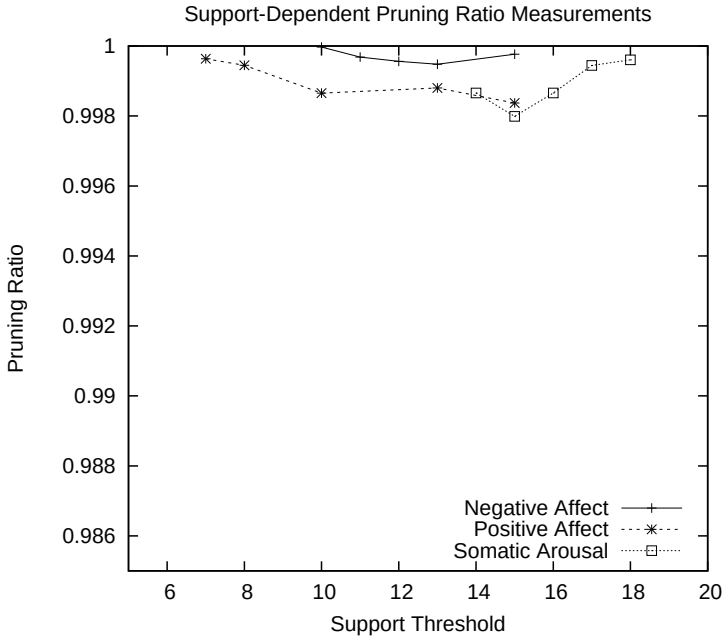
Pruning For each condition, the pruning behaviour of the support-dependent measurements is similar to some extend, as can be seen in Figure 8.9(a). Before the breakpoint, pruning is high, due to the fact that only the most specific rules with the highest scores are kept, and all combinations of more general conjunctions are pruned. On, or short after the breakpoint, a local minimum is reached, where relatively the least rules are pruned. After that, less rules will be generated due to stricter thresholds, and thus increasingly more rules will be pruned.

Due to the fact that less rules are generated as well, the resolution of the measurements becomes lower, and thus the measurement becomes less accurate. Differences in the position are the result of the different rankings: some generated a lot of rules at a higher support, while others needed a lower threshold.

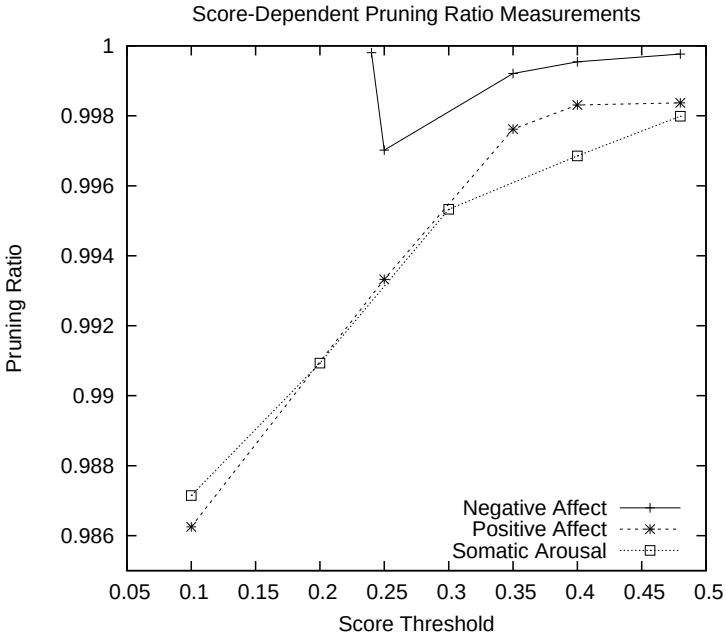
Depending on whether there is a breakpoint, different behaviour is shown in the score-dependent measurements in Figure 8.9(b). If there is a breakpoint, such as in the NA condition, then the line has a similar shape as the ones in Figure 8.9(a), for a similar reason: before a breakpoint there is an explosion of rules that have a maximum score higher than the breakpoint score value, for the fixed support threshold of $P = 15$. However, if the support threshold is too high, a rule explosion will never occur, and pruning shows behaviour similar to that of Figure 8.4(b), whereby the pruning ratio is monotonically decreasing as the score threshold decreases. Again, the score threshold shows to have a much greater impact on the pruning ration than the support threshold.

Clustering When comparing the behaviours in Figure 8.10(a) and Figure 8.6(a) in Section 8.4.1, they look nothing alike. In fact, they are each other's opposites. A clear pattern can be seen. After the breakpoint, the rules-per-cluster ratio decreases to a local minimum, and then rises again. This local minimum coincides with the local minima in Figure 8.9(a). That is because at that point, the pruning effect is relatively lower, and only the best rules for each cluster are kept, and are thus spread thin over all clusters. After that, only the very best rules survive, and these are grouped in one or a few clusters.

An explanation for this phenomenon lies in the experiment setup. In the new ranking, the interest class SNPs are only those SNPs that do not occur in the control group, which is likely to result in less overlap in ontology association. Combined with a modified ES score that punishes rules that have associations with many SNPs in the control group, this promotes high scores for rules that contain SNPs only in the interest class. These rules usually contain a small set of SNP identifiers that occur



(a)



(b)

Figure 8.9: Pruning measurements of the ontology mappings experiment

over and over. Since clustering is based on a percentage of the support shared by rules, these rules will all be in the same cluster(s), thus keeping the rules-per-cluster ratio high. Since these rules have the highest score, they are more likely to have a high score even if the support threshold increases, thus increasing the rules-per-cluster ratio.

The NA condition and PA condition in Figure 8.10(b) show similar behaviours as in Figure 8.6(b) in Section 8.4.1, but for the rise in rules-per-cluster ratio, which has a similar explanation as the one in the support-dependent measurements. However, the rules-per-cluster ratio in the SA condition is monotonically increasing. This is due to the ranking; the SA class was rich in rules that were similar or even the same in participants, and thus concentrated in a few clusters. As the score threshold grew higher, all other clusters were pruned, while the biggest clusters containing the rules with the highest scores remained.

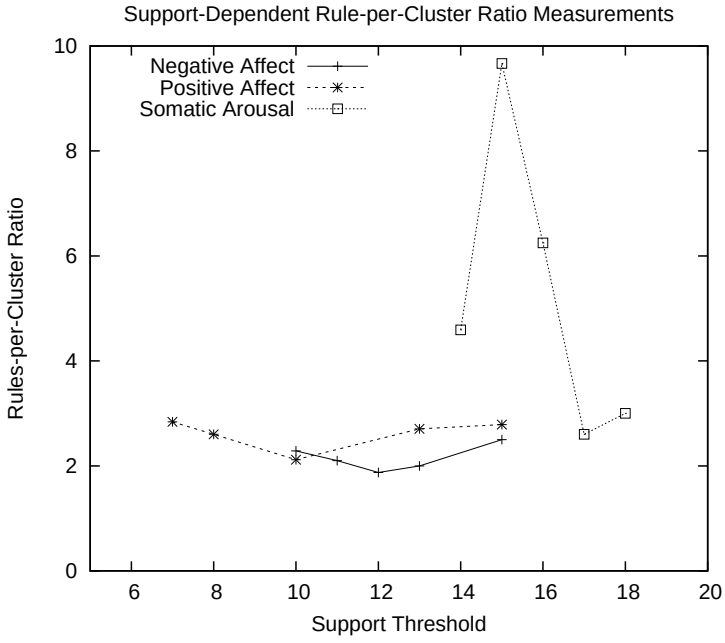
Rules The first observation is that rules in the current experiment setup have lower scores, and that the scores are compacted more into a middle range, which is largely caused by of the experimental setup as well as the modified ES score measurement. A second observation is that rules do tend to be more specific, and yield more conjunctions due to the increased specificity of each identifier. For the NA and PA conditions, rules were still noticeably related to their gene translation counterparts, albeit in a representation of more specific ontological terms. Rules in the PA condition even had very high scores, though that was the result of a group of SNPs that scored very high t-scores in that specific condition. It is likely that this group also influenced the scores of the other two conditions.

The SA condition yielded the most surprising rules of the three conditions, involving concepts such as sensory perception of sound and inner ear development, as well as neurotransport activity. Although the score of these rules was fairly low, they are specific to the SNPs that scored highest only for the SA condition, and were confirmed by clustering.

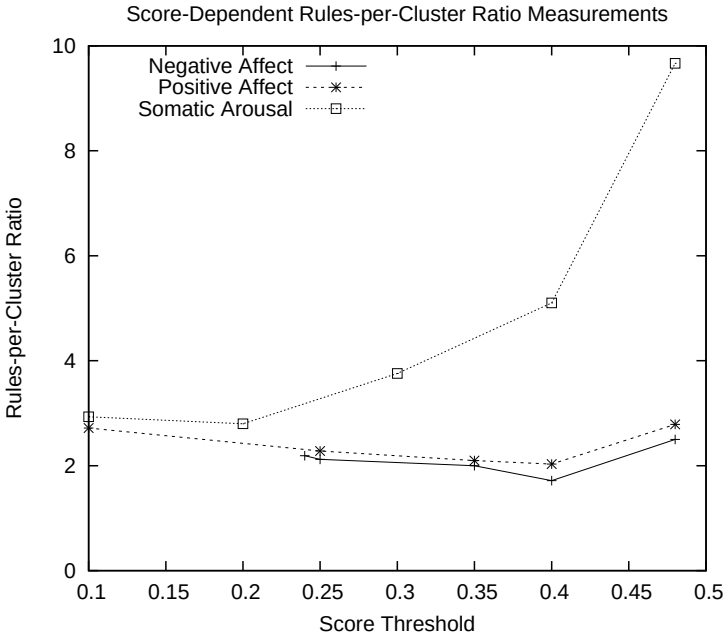
8.5 Conclusions and Future Work

In this chapter we discussed experiments that were performed on a SNP study done on human depression. We provided an introduction to SNPs and related concepts, and explained their importance. We also presented an overview of the SNP depression study and how it led to the various rankings that we used in our experiments.

In our experiments, we took two approaches to experimentation on SNP rankings. In the first approach, we mapped the SNP identifiers to genes, which resulted in a ranking of genes that we experimented on. In the second approach, we experimented on the original ranking of SNP identifiers and their scores, and created a mapping from SNP identifiers to ontological concepts in GO and KEGG.



(a)



(b)

Figure 8.10: Cluster ratio measurements of the ontology mappings experiment

When comparing the two methods, both have their merits and difficulties. Gene translation results in a smaller ranking, and can thus be mined faster with lower thresholds. The downside is that gene scores are aggregated, and that the experiment is only as accurate as the mapping of SNPs to gene identifiers. Using SNPs in the ranking increases the accuracy of the experiment, identifying rules that over-expressed SNPs have in common, and not their associated genes. However, associating these SNPs to ontological concepts still requires the SNP to gene mapping. Another problem is the sheer size of the identifier space, which usually makes it challenging to perform single-class mining on the whole data-space. Therefore, this method is better suited for more aimed studies, to take a closer look at specific subgroups, as we did in the multi-class experiments.

When analyzing the behaviours of both methods, they often show similarities across different rankings and dimensions. Of course there are differences, either caused by the experimental setup of the method, the score function, or just by the ranking and inherently the ontological properties of the identifiers; different concept associations lead to a difference in the quality and amount of rules that can be mined.

When comparing execution times and the amount of rules that have been generated, the behaviours of the two methods are in principle the same. Given that the score threshold and the support threshold is small enough, a rule explosion will take place at some point, a point that we referred to as the breakpoint. However, if one of the thresholds is high enough, no explosion will take place and the execution time will only grow at a somewhat linear pace. When an explosion does occur, it seems more sudden in the multi-class experiments, though that is also dependent on factors such as ranking and experiment design.

Pruning behaviour is similar between the two conditions as well, for both scoring and support constraints. Given that there is a breakpoint, pruning ratio will be high before that, because only the most specific and most interesting rules survive, while all other combinations will be pruned away. After the breakpoint, when the number of generated rules has dropped considerably, there is also a drop in pruning ratio, since the number of rules shrank disproportionately to the number of rule options generated. After that it will slowly rise again, due to the increasing strictness of the score or the support threshold.

In cases where a breakpoint does not appear, behaviour is also the same for both methods, as shown in the score-dependent measurements. Both show a logarithmic shape since lower score thresholds have a higher effect on the pruning ratio than higher score thresholds. Since the shape after the breakpoint is the same for both score and support-dependent thresholds, the same statement is likely to hold for behaviour in the support-dependent measurements.

The rules-per-cluster ratio is a bit off for both methods. In the gene translation method, the ratio is monotonically decreasing as support or score thresholds get bigger. The reverse is true in the case of the multi-class experiments, where in the end the ratio was steadily growing bigger. This has nothing to do with the difference between

the two approaches, but more with the experiment setup and the Fantom implementation.

When comparing the outputs between the two experiments, they show similarities for the rankings that have the highest t-scores. This is due to the experiment setup of the multi-class experiment. The rule output shows both familiar and unexpected concepts, which can of course be mere artifacts of ontologies, but they can also serve as new hypotheses for further research.

As future work, there are a few questions that still need answering. More research has to be done on the occurrence of the breakpoint and the exact relationship with pruning ratio. There is a strong connection between the two, and combining research on both could lead to an algorithm that automatically determines an optimal threshold for a certain ranking, to optimize a trade-off between running time and rule specificity.

Another point of interest is the score function in the multi-class experiments. Although it performed well in most instances, it also compacted the rule space, leaving potentially useful and interesting rules buried between many others. This might be a combined artifact from both the experiment setup and the score function, so a proper study on that phenomenon would help to refine the experiment setup as well as the score function.

Finally, other forms of clustering other than one with a hard cutoff may result in a better and more constant result. Although behaviour could be explained and rationalized, perhaps clustering on the basis of individual or combined ontological concepts within the rules itself would result in better and more similar clusters.

Bibliography

[ABB+00] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: Tool for the unification of biology. The Gene Ontology Consortium. *Nat Genet*, 25(1):25–29, 2000.

[ABD+04] Alain Abran, Pierre Bourque, Robert Dupuis, James W. Moore, and Leonard L. Tripp. *Guide to the Software Engineering Body of Knowledge — SWEBOK*. IEEE Press, 2004.

[Abd07] H. Abdi. *Bonferroni and sidak corrections for multiple comparisons*. Sage, 2007.

[AC06] Juan Jos Garca Adeva and Rafael A. Calvo. Mining text with Pimiento. *IEEE Internet Computing*, 10(4):27–35, 2006.

[AIS93] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216. ACM Press, 1993.

[AN07] A. Asuncion and D.J. Newman. UCI machine learning repository. http://www.ics.uci.edu/mlearn/MLRepository.html, 2007.

[AS09] M. Ackermann and K. Strimmer. A general modular framework for gene set enrichment analysis. *BMC Bioinformatics*, 10:47+, 2009.

[ASS+02] S. A. Armstrong, J. E. Staunton, L. B. Silverman, R. Pieters, M. L. den Boer, M. D. Minden, S. E. Sallan, E. S. Lander, T. R. Golub, and S. J. Korsmeyer. MLL translocations specify a distinct gene expression profile that distinguishes a unique leukemia. *Nat Genet*, 30:41–47, 2002.

[AY98] Charu C. Aggarwal and Philip S. Yu. A new framework for itemset generation. In *PODS '98: Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 18–24. ACM, 1998.

[Bal95] Mira Balaban. The F-logic Approach for Description Languages. *Annals of Mathematics and Artificial Intelligence*, 15:19–60, 1995.

[BBCD+98] Len Bass, Charles Buhman, Santiago Comella-Dorda, Fred Long, and John Robert. Volume 1: Market assessment of component-based software engineering. http://handle.dtic.mil/100.2/ADA395250, 1998.

[BBMM04] Marco Botta, Jean-François Boulicaut, Cyrille Masson, and Rosa Meo. Query languages supporting descriptive rule mining: A comparative study. In *Database Support for Data Mining Applications*, volume 2682 of *Lecture Notes in Computer Science*, pages 24–51. Springer, 2004.

[BCF+08] Hendrik Blockeel, Toon Calders, Elisa Fromont, Bart Goethals, Adriana Prado, and Céline Robardet. An inductive database prototype based on virtual mining views. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1061–1064. ACM, 2008.

[BCK03] L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice*. Addison-Wesley, 2 edition, 2003.

[BCKL02] Daniele Braga, Alessandro Campi, Mika Klemettinen, and Pier Luca Lanzi. Mining Association Rules from XML Data. In *DaWaK 2000: Proceedings of the 4th International Conference on Data Warehousing and Knowledge Discovery*, pages 21–30. Springer-Verlag, 2002.

[BCM04] Elisa Bertino, Barbara Catania, and Anna Maddalena. Towards a language for pattern manipulation and querying. In *PaRMa*, volume 96 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2004.

[BIAS03] B. M. Bolstad, R. A. Irizarry, M. Astrand, and T. P. Speed. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, 19(2):185–193, 2003.

[BKM98] Jean-François Boulicaut, Mika Klemettinen, and Heikki Mannila. Querying inductive databases: A case study on the mine rule operator. In *PKDD*, volume 1510 of *Lecture Notes in Computer Science*, pages 194–202. Springer, 1998.

[BMUT97] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. Dynamic itemset counting and implication rules for market basket data. In *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pages 255–264. ACM, 1997.

[Bod03] Ferenc Bodon. A fast apriori implementation. In *FIMI '03, Frequent Itemset Mining Implementations, Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations*, 2003.

[Boo93] Grady Booch. *Object-Oriented Analysis and Design with Applications*. Addison-Wesley, 2nd edition, 1993.

[Bor03] Christian Borgelt. Efficient implementations of apriori and eclat. In *FIMI '03, Frequent Itemset Mining Implementations, Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations*, 2003.

[bou05] *Constraint-Based Mining and Inductive Databases, European Workshop on Inductive Databases and Constraint Based Mining, Hinterzarten, Germany, 2004, Revised Selected Papers*, volume 3848 of *Lecture Notes in Computer Science*. Springer, 2005.

[BPE07] Business process execution language for web services (BPEL4WS) 1.1. http://www.ibm.com/developerworks/library/specification/ws-bpel/, 2007.

[Bro87] Frederick P. Brooks. No Silver Bullet: Essence and Accidents of Software Engineering. *IEEE Computer*, 20(4):10–19, 1987.

[CGP06] Toon Calders, Bart Goethals, and Adriana Prado. Integrating Pattern Mining in Relational Databases. In *PKDD*, volume 4213 of *Lecture Notes in Computer Science*, pages 454–461. Springer, 2006.

[CMM+04] Barbara Catania, Anna Maddalena, Maurizio Mazza, Elisa Bertino, and Stefano Rizzi. A framework for data mining pattern management. In *PKDD*, volume 3202 of *Lecture Notes in Computer Science*, pages 87–98. Springer, 2004.

[Con09] The Gene Ontology Consortium. Mappings of External Classification Systems to GO. http://www.geneontology.org/GO.indices.shtml, 2009.

[CZW+06] William K. Cheung, Xiao-Feng Zhang, Ho-Fai Wong, Jiming Liu, Zong-Wei Luo, and Frank C.H. Tong. Service-oriented distributed data mining. *IEEE Internet Computing*, 10(4):44–54, 2006.

[Dav06] J. Davies. *Semantic Web Technologies: Trends and Research in Ontology-based Systems*. John Wiley & Sons, 2006.

[DBG+06] Souptik Datta, Kanishka Bhaduri, Chris Giannella, Ran Wolff, and Hillol Kargupta. Distributed data mining in peer-to-peer networks. *IEEE Internet Computing*, 10(4):18–26, 2006.

[DBL02] *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan*. IEEE Computer Society, 2002.

[DBL06] *Knowledge Discovery in Inductive Databases, 4th International Workshop, KDID 2005, Porto, Portugal, October 3, 2005, Revised Selected and Invited Papers*, volume 3933 of *Lecture Notes in Computer Science*. Springer, 2006.

[DDH72] O. J. Dahl, E. W. Dijkstra, and C. A. R. Hoare, editors. *Structured programming*. Academic Press Ltd., 1972.

[DGST09] Ewa Deelman, Dennis Gannon, Matthew Shields, and Ian Taylor. Workflows and e-science: An overview of workflow system features and capabilities. *Future Generation Computer Systems*, 25(5):528–540, 2009.

[Dij72] Edsger W. Dijkstra. The humble programmer. *Commun. ACM*, 15(10):859–866, 1972.

[Fis22] R. A. Fisher. On the interpretation of χ^2 from contingency tables, and the calculation of p. *Journal of the Royal Statistical Society*, 85(1):87–94, 1922.

[Fis67] Ronald A. Fisher. *Statistical methods for research workers*. Number 5 in Biological monographs and manuals. Oliver and Boyd, 13 edition, 1967.

[FPSS96] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17:37–54, 1996.

[Gar95] S. Garner. Weka: The waikato environment for knowledge analysis. In *Proceedings of the New Zealand Computer Science Research Students Conference*, pages 57–64, 1995.

[Gen09] GeneRIF – Gene Reference Into Functions. http://www.ncbi.nlm.nih.gov/projects/GeneRIF/, 2009.

[GGNZ06] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lotfi A. Zadeh, editors. *Feature Extraction: Foundations and Applications (Studies in Fuzziness and Soft Computing)*. Springer, 2006.

[GJF06] Dorgival Guedes, Wagner Meira Jr., and Renato Ferreira. Anteater: A service-oriented architecture for high-performance data mining. *IEEE Internet Computing*, 10(4):36–43, 2006.

[Gro07] The Open Group. Definition of SOA, version 1.1. http://opengroup.org/projects/soa/doc.tpl?gdid=10632, 2007.

[GST+99] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537, 1999.

[HD06] Jeffrey Hasan and Mauricio Duran. *Expert Service-Oriented Architecture in C# 2005, Second Edition*. Apress, 2006.

[HFW+96] Jiawei Hah, Yongjian Fu, Wei Wang, Krzysztof Koperski, and Osmar Zaiane. DMQL: A data mining query language for relational databases. 1996.

[HPY00] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *SIGMOD Conference*, pages 1–12. ACM, 2000.

[HST+07] Da Huang, Brad Sherman, Qina Tan, Jack Collins, W. Gregory Alvord, Jean Roayaei, Robert Stephens, Michael Baseler, H. Clifford Lane, and Richard Lempicki. The DAVID gene functional classification tool: A novel biological module-centric algorithm to functionally analyze large gene lists. *Genome Biology*, 8(9):R183+, 2007.

[IM96] Tomasz Imielinski and Heikki Mannila. A database perspective on knowledge discovery. *Commun. ACM*, 39(11):58–64, 1996.

[IV99] Tomasz Imielinski and Aashu Virmani. Msql: A query language for database mining. *Data Min. Knowl. Discov.*, 3(4):373–408, 1999.

[Jac92] I. Jacobson. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley, 1992.

[JMD+05] Aleks Jakulin, Martin Možina, Janez Demšar, Ivan Bratko, and Blaž Zupan. Nomograms for visualizing support vector machines. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 108–117. ACM, 2005.

[KAH+05] Stefan Kramer, Volker Aufschild, Andreas Hapfelmeier, Alexander Jarasch, Kristina Kessler, Stefan Reckow, Jörg Wicker, and Lothar Richter. Inductive databases in the relational model: The data as the bridge. In *KDID* [DBL06], pages 124–138.

[KFH+06] G. Kandaswamy, L. Fang, Y. Huang, S. Shirasuna, S. Marru, and D. Gannon. Building web services for scientific grid applications. *IBM J. Res. Dev.*, 50(2/3):249–260, 2006.

[KLJ03] Branko Kavšek, Nada Lavrac, and Viktor Jovanoski. *APRIORI-SD: Adapting Association Rule Learning to Subgroup Discovery*, volume 2810. 2003.

[KS05] Mehmed Kantardzie and Ashok N. Srivastava. Data mining: Concepts, models, methods, and algorithms. *Journal of Computing and Information Science in Engineering*, 5(4):394–395, 2005.

[Lab09] Kanehisa Laboratories. Kegg pathway mappings. ftp://ftp.genome.jp/pub/kegg/pathway/map, 2009.

[LB06] Elo Leung and Pierre R. Bushel. Page: phase-shifted analysis of gene expression. *Bioinformatics*, 22(3):367–368, 2006.

[LKFT04] Nada Lavrač, Branko Kavšek, Peter Flach, and Ljupčo Todorovski. Subgroup discovery with CN2-SD. *J. Mach. Learn. Res.*, 5:153–188, 2004.

[LLF+09] Cui Lin, Shiyong Lu, Xubo Fei, Artem Chebotko, Darshan Pai, Zhaoqiang Lai, Farshad Fotouhi, and Jing Hua. A reference architecture for scientific workflow management systems and the view soa solution. *IEEE Transactions on Services Computing*, 2:79–92, 2009.

[LM98] Huan Liu and Hiroshi Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, 1998.

[LRS+08] Y. Lu, R. Rosenfeld, I. Simon, G. J. Nau, and Z. Bar-Joseph. A probabilistic generative model for go enrichment analysis. *Nucl. Acids Res.*, pages 434+, 2008.

[LV03] P. Lyman and H. R. Varian. How much information (2003). http://www.sims.berkeley.edu/how-much-info-2003, 2003.

[LZF02] Nada Lavrac, Filip Zelezný, and Peter A. Flach. RSD: Relational subgroup discovery through first-order feature construction. In *ILP*, volume 2583 of *Lecture Notes in Computer Science*, pages 149–165. Springer, 2002.

[MCOW05] X. Mao, T. Cai, J. G. G. Olyarchuk, and L. Wei. Automated genome annotation and pathway identification using the kegg orthology (ko) as a controlled vocabulary. *Bioinformatics*, 21(19):3787–3793, 2005.

[Meo05] Rosa Meo. Inductive databases: Towards a new generation of databases for knowledge discovery. In *DEXA Workshops*, pages 1003–1007. IEEE Computer Society, 2005.

[MER01] Barbara McKee, Dave Ehnebuske, and Dan Rogers. UDDI version 2.0 api specification. http://xml.coverpages.org/ProgrammersAPI-V200-Open-20010608.pdf, 2001.

[Mey92] Bertrand Meyer. Applying "design by contract". *IEEE Computer*, 25(10):40–51, 1992.

[MOPT05] D. Maglott, J. Ostell, K. D. Pruitt, and T. Tatusova. Entrez gene: gene-centered information at ncbi. *Nucleic Acids Res*, 33(Database issue), 2005.

[MP02] Rosa Meo and Giuseppe Psaila. Toward XML-based knowledge discovery systems. In *ICDM* [DBL02], pages 665–668.

[MPC98] Rosa Meo, Giuseppe Psaila, and Stefano Ceri. An extension to SQL for mining association rules. *Data Min. Knowl. Discov.*, 2(2):195–224, 1998.

[MRB04] Cyrille Masson, Céline Robardet, and Jean-François Boulicaut. Optimizing subset queries: A step towards sql-based inductive databases for itemsets. In *SAC*, pages 535–539. ACM, 2004.

[MyG08] MyGrid. Taverna workbench 2.0. http://taverna.sourceforge.net/, 2008.

[NK05] Siegfried Nijssen and Joost N. Kok. Multi-class correlated pattern mining. In *KDID* [DBL06], pages 165–187.

[OAM+03] Angele Moench Oppermann, J. Angele, E. Moench, S. Staab, and D. Wenke. Ontonova @ project halo. In *Proceedings of the Second International Semantic Web Conference (ISWC2003). 2003*, pages 913–928. Springer Verlag, 2003.

[OGS+99] H. Ogata, S. Goto, K. Sato, W. Fujibuchi, H. Bono, and M. Kanehisa. KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res*, 27(1):29–34, 1999.

[Omi03] Edward R. Omiecinski. Alternative interest measures for mining associations in databases. *IEEE Transactions on Knowledge and Data Engineering*, 15(1):57–69, 2003.

[Onl07] Computer Dictionary Online. Ontology definition in information science. http://www.computer-dictionary-online.org/ontology.htm?q=ontology, 2007.

[OWS+08] D.I. Oomsma, G. Willemsen, P.F. Sullivan, P. Heutink, P. Meijer, and D. et al. Sondervan. Genome-wide association of major depression: description of samples for the gain major depressive disorder study: Ntr and nesda biobank projects. *Eur.J.Hum.Genet*, 16:335–42, 2008.

[Par72] D. L. Parnas. A technique for software module specification with examples. *Commun. ACM*, 15(5):330–336, 1972.

[PSZ+07] B.W. Penninx, J.H. Smit, F.G. Zitman, W. Nolen, A.T. Beekman, and R. van Dyck. Netherlands study of depression and anxiety (NESDA): examining the long-term course of affective disorders. *European Psychiatry*, 22:S330, 2007.

[PTM07] K. D. Pruitt, T. Tatusova, and D. R. Maglott. Ncbi reference sequences (refseq): A curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic Acids Res*, 35(Database issue):D61–D65, 2007.

[Rae02] Luc De Raedt. A perspective on inductive databases. *SIGKDD Explorations*, 4(2):69–77, 2002.

[RBL+90] James R. Rumbaugh, Michael R. Blaha, William Lorensen, Frederick Eddy, and William Premerlani. *Object-Oriented Modeling and Design*. Prentice Hall, 1990.

[RJB04] James Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 2nd edition, 2004.

[RJLM02] Luc De Raedt, Manfred Jaeger, Sau Dan Lee, and Heikki Mannila. A theory of inductive query answering. In *ICDM* [DBL02], pages 123–130.

[Roy70] W. Royce. Managing the development of large software systems. In *Proc. IEEE Wescon*, pages 1–9, 1970.

[SGM02] Clemens Szyperski, Domiinik Gruntz, and Stephan Murer. *Component Software: Beyond Object-Oriented Programming*. Addison Wesley, 2002.

[STM+05] Aravind Subramanian, Pablo Tamayo, Vamsi K. Mootha, Sayan Mukherjee, Benjamin L. Ebert, Michael A. Gillette, Amanda Paulovich, Scott L. Pomeroy, Todd R. Golub, Eric S. Lander, and Jill P. Mesirov. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences of the United States of America*, 102(43):15545–15550, 2005.

[TLT07] I. Trajkovski, N. Lavrač, and J. Tolar. Segs: Search for enriched gene sets in microarray data. *J Biomed Inform*, pages 588–601, 2007.

[TS06] C. Todorova and K. Stefanov. Selection and use of domain ontologies in learning networks for lifelong competence development. In *Proceedings of the 2006 International Workshop on Learning Networks for Lifelong Competence Development*, pages 11–17. Springer Verlag, 2006.

[TZTL06] Igor Trajkovski, Filip Zelezný, Jakub Tolar, and Nada Lavrac. Relational subgroup discovery for descriptive analysis of microarray data. In *Lecture Notes in Computer Science, Computational Life Sciences II*, volume 4216 of *Lecture Notes in Computer Science*, pages 86–96. Springer, 2006.

[VDS+07] I. Vastrik, P. D'Eustachio, E. Schmidt, G. Joshi-Tope, G. Gopinath, D. Croft, B. de Bono, M. Gillespie, B. Jassal, S. Lewis, L. Matthews, G. Wu, E. Birney, and L. Stein. Reactome: A knowledgebase of biological pathways and processes. *Genome Biology*, 8:39+, 2007.

[W3C01] The World Wide Web Consortium W3C. Web services description language (WSDL) 1.1. http://www.w3.org/TR/wsdl, 2001.

[W3C04a] The World Wide Web Consortium W3C. Resource description framework (rdf). http://www.w3.org/RDF/, 2004.

[W3C04b] The World Wide Web Consortium W3C. Web ontology language (OWL). http://www.w3.org/2004/OWL/, 2004.

[W3C07] The World Wide Web Consortium W3C. SOAP version 1.2 part 0: Primer (second edition). http://www.w3.org/TR/soap12-part0/, 2007.

[WF99] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*. Morgan Kaufmann, San Francisco, 1999.

[WL02] Mark D. Wilkinson and Matthew Links. Biomoby: An open source biological web services proposal. *Briefings in Bioinformatics*, 3(4):331–341, 2002.

[WLDP02] H. M. Wain, M. Lush, F. Ducluzeau, and S. Povey. Genew: The human gene nomenclature database. *Nucleic Acids Research*, 30:169–171(3), 2002.

[Wro97] Stefan Wrobel. An algorithm for multi-relational discovery of subgroups. In *PKDD '97: Proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 78–87. Springer-Verlag, 1997.

[WvVG+09] K Wardenaar, T. van Veen, E. Giltay, E. De Beurs, B. Penninx, and F. Zitman. Development and validation of a 30-item short form of the dutch adaptation of the mood and anxiety symptoms questionnaire: The SF-DAMASQ. *Psychiatry Res*, 2009.

[YC75] E. Yourdon and L. L. Constantine. *Structured Design*. Yourdon Press, 1975.

[ZW08] Qi Zheng and Xiu-Jie J. Wang. GOEAST: A web-based software toolkit for gene ontology enrichment analysis. *Nucleic acids research*, pages 358–63, 2008.

Part IV

Appendices

Appendix A

Fantom Formats

In this appendix we discuss the formats of the different inputs for the Fantom service. The inputs discussed in this appendix are the ranked list, mappings, ontologies, and the interface for score functions. For each input we discuss its potential contents as well as its format. We also supply an example for each of the formats.

File Formats

In this part we will discuss the file formats that Fantom takes as inputs. Included in this section are the ranked list, mappings and ontology file formats.

Ranked List

The ranked list is the ranking of identifiers, together with their scores. It is a table with three columns, separated by tab indents. The first column denotes the class label of the identifier, which are ignored if a single-class experiment is conducted. The second column contains the (unique) identifiers. If multiple identical identifiers are found, their scores will be averaged. Finally, the third column contains the scores. The Fantom service makes little assumptions about the scores, although the score functions might. For example, the Enrichment Score function assumes that a score of 0 is the least interesting, while very high positive or negative scores indicate a high interestingness of that identifier.

An example ranking is shown below. This ranking is part of the complete gene ranking obtained from the AML vs. ALL microarray experiment.

```
INTEREST     945    1.93768463633657
INTEREST    6929    1.91673657142379
INTEREST    1675    1.86252699812831
INTEREST    1509    1.72544529626413
INTEREST     896    1.58730893402055
```

Mappings

Fantom uses three kinds of mappings: Identifier mappings, interaction mappings and ontology mappings. Regardless of the mapping type, they all have the same structure; each file contains two columns, again separated by a tab indent. The first column contains an identifier, which must correspond with the identifier in the ranking, and the second column contains the identifier that is mapped to. If these are multiple identifiers, which is likely in ontology and interaction mappings, then the identifiers are separated by comma's.

The mapping shown below maps ENTREZ identifiers to HUGO Symbol identifiers. This mapping, and its inverse mapping, were taken from the KEGG public ftp site[1].

```
100009601    TRNAY1
100009602    TRNAY2
100009603    TRNAA2
100009604    TRNAA3
100009605    TRNAF1
```

The next mapping it a part of an ontology mapping, which maps ENTREZ identifiers to their most specific GO concepts. This partial mapping was obtained from the NCBI public ftp site[2] and then reformatted to the format shown below:

```
100128553    GO:0005575,GO:0008150
100128582    GO:0003676,GO:0005622,GO:0005634,GO:0006350,GO:0006355
100129271    GO:0003674,GO:0005575,GO:0008544
100129441    GO:0016020,GO:0016021
100129669    GO:0005886,GO:0016021
```

Finally, the last mapping is an interaction mapping, which maps identifiers to other identifiers that they interact with. Shown below is a partial mapping that maps interactions between ENTREZ identifiers. It was obtained from the NCBI GeneRIF project ftp site[3], and again reformatted.

```
10112    51560,5870
10114    355,7157,7161,7341,8772
10116    10116,317,355
10121    10121,1639,23299,4926,51164,6711,6712
10125    5495,7428,8525
```

[1] ftp://ftp.genome.jp/pub/kegg/pathway/organisms/hsa/hsa_synonym
[2] ftp://ftp.ncbi.nlm.nih.gov/gene/DATA/gene2go.gz
[3] ftp://ftp.ncbi.nlm.nih.gov/gene/GeneRIF/

Ontology

The ontology files represent the ontology concepts and the relationships among those concepts, as well as the predicates that these concepts belong to. An ontology file can contain multiple ontological predicates, which we perceive as the actual ontologies. For example, GO actually consists of three separate ontologies: cellular component, biological process and molecular function.

To be compatible with a range of ontologies, the ontology file format for the Fantom service contains six columns, which are separated by tabs. The first column contains the concept identifier, which must be unique for each concept, even across ontologies. The second column contains the ontology predicate, identifying to which ontology the concept belongs to. The third column contains the concept description, which is used in Fantom rules. The fourth column contains the list of alias keys, if any are available. These keys are identifiers that the concept is also known as. When the ontology tree is filled, these alias concepts are also annotated with an associated identifier whenever this concept is. The fifth column contains the parent or parents of this concept. Through these relationships, we can build the internal DAG representation of each ontology, which is used for pruning. Finally, the sixth column is a binary indication whether the current concept is obsolete. If the concept is obsolete, there is a 1, and the alias column should provide the new concept or concepts. If the concept is not obsolete, is a 0 in the last column.

The partial ontology shown below is a small part of the GO ontology, which was obtained from the GO website[4]. Since the ontology was originally formatted in the Open Biomedical Ontologies (OBO) format[5], we had to reformat it first.

GO:0008343	bio_proc	adult feeding behavior	[]	[GO:0007631,GO:0030534]	0
GO:0008344	bio_proc	adult locomotory behavior	[]	[GO:0007626,GO:0030534]	0
GO:0008354	bio_proc	germ cell migration	[]	[GO:0016477,GO:0007276]	0
GO:0008355	bio_proc	olfactory learning	[]	[GO:0007612,GO:0042048]	0
GO:0008356	bio_proc	asymmetric cell division	[]	[GO:0051301]	0
GO:0008409	mol_func	5–3- exonuclease activity	[]	[GO:0004527]	0
GO:0008410	mol_func	CoA-transferase activity	[]	[GO:0016782]	0

Score Function Interface

The Fantom service supports the usage of custom score measurements by way of uploading dll libraries that were programmed in the .Net framework. These score functions do have to adhere to the following interface:

[4] http://www.geneontology.org/ontology/obo_format_1_2/gene_ontology.1_2.obo

[5] http://www.obofoundry.org/

```
public interface class IFantomScorer
{
    String GetScoreFunctionName();
    void ScoreRule(FANTOM_Rule rule);
    void MaxScoreRule(FANTOM_Rule rule);
};
```

The first function, *GetScoreFunctionName*, is used to identify the score function with. This name must match the name supplied as a parameter in the Fantom service interface. If no match is found, the default Enrichment Score function is chosen. The second function, *ScoreRule*, is used to determine a score of a rule, whereby the rule is supplied as a parameter of type *FANTOM_Rule*, which is a publicly available class. Finally, the function *MaxScoreRule* is used to provide the supplied rule with its maximum potential score.

Appendix B

Enrichment Score Maximization

In this appendix we will provide a detailed description of the Enrichment Set score function. We first discuss the mathematical properties of the function, and then define the maximization problem, along with the polynomial solution.

The Enrichment Score

Consider a set X of items of size $|X| = n$. Let them be ordered, $X = (x_1, \ldots, x_n)$, such that they can be identified with the set $[N] = \{1, 2, \ldots, n\}$. Furthermore, let there be an *individual score* R_i assigned to each item i, $1 \leq i \leq n$. Let the order of X be compatible with the scores, such that the R_i form a decreasing sequence. Consider a sequence $S = \{n_1, \ldots, n_m\} \subset N$ of size $|S| = m$ with $m \leq n$, and let $r_i = R_{n_i}$ be the score of the i-th item from S. The number n_i is the position of this item in the set X (i.e., the rank of r_i in the set of all R_j's).

Definition 1. *The i-th* partial score *associated with the set S is*

$$c_i = \frac{\sum_{j=1}^{i} r_j}{\sum_{l=1}^{m} r_l} - \frac{n_i - i}{n - m}.$$

The partial score consists of two parts. The first term describes the positive growth of the score due to the relative individual scores of the elements from S encountered up to and including position i. The second term implements a penalty for each position encountered which does *not* correspond to an element from S. Writing $P_{\text{hit}}(S, i)$ for the first term, and $P_{\text{miss}}(S, i)$ for the second, the partial score satisfies $c_i = |P_{\text{hit}}(S, i) - P_{\text{miss}}(S, i)|$, as defined by [STM+05].

Define $c_0 = 0$. From the definition, clearly also $c_m = 0$ when $m = n$, and furthermore $-1 \leq c_i \leq 1$ for all $1 \leq i < m$.

Definition 2. *The* Enrichment Score *associated with the set S is*

$$E_S = \max_i |c_i|.$$

The enrichment score can obviously be calculated in linear time (with respect to $|S| = m$). Table B.1 shows an example for a dataset of $n = 34,327$ genes and a subset of size $m = 26$.

Maximal Subset Score

We now consider the following problem. Let $T \subseteq S$ be a subset of S of size $|T| = m' < m = |S|$.

Question 1 (k-removal subset enrichment score problem)**.** *Consider all possible m'-element subsets T of a given $S \subseteq X$, where $m' = m - k$ for a given k. What is the maximum of the possible enrichment scores E_T?*

Although theoretically possible, it is clearly infeasible to calculate the enrichment score for all $\binom{m}{m'}$ such subsets. We therefore seek a more efficient algorithm, i.e., at most polynomial in m. Intuitively, one would think that one can solve the problem efficiently by forward induction, i.e., by finding the optimal $m-1$ subset S_{m-1} of S, then its optimal $m-2$ subset $S_{m-2} \subset S_{m-1}$, etc., until arriving at the optimal $S_{m'} \subset S_{m'+1}$. This is not the case, however. Table B.2 shows an example where the optimal 20-element subset of the 26-element set S given in Table B.1 is not obtained by restriction of the 21-element subset. The *subset enrichment problem* is therefore a nonlocal problem. Nevertheless, let us consider what happens in the simple case where $m' = m - 1$, i.e., where $T = S \setminus \{s_k\}$ for some $k \leq m$. The new partial scores are then

$$c_i^{(k)} = \begin{cases} \frac{\sum_{j=1}^{i} r_j}{\sum_{l=1}^{m} r_l - r_k} - \frac{n_i - i}{n-m+1} & \text{for } i < k \\ \frac{\sum_{j=1}^{i} r_j - r_k}{\sum_{l=1}^{m} r_l - r_k} - \frac{n_i - i + 1}{n-m+1} & \text{for } i > k. \end{cases} \tag{B.1}$$

where the superscript (k) denotes removal of the k-th element from the set S. More generally, let $c_i^{(A)}$, where $A \subseteq S$ such that $s_i \notin A$, denote the i-th partial score after removal of the elements of A from S.

Lemma 1. *The difference between the scores before and after removal of s_k is*

$$c_i^{(k)} - c_i = \begin{cases} \frac{\sum_{j=1}^{i} r_j}{\sum_{l=1}^{m} r_l} \cdot \frac{r_k / \sum_l r_l}{1 - r_k / \sum_l r_l} + \frac{1}{n-m} \cdot \frac{n_i - i}{n-m+1} & \text{for } i < k \\ \left(\frac{\sum_{j=1}^{i} r_i}{\sum_{l=1}^{m} r_l} - 1 \right) \cdot \frac{r_k / \sum_l r_l}{1 - r_k / \sum_l r_l} - \frac{1}{n-m} \cdot \frac{(n-n_i) - (m-i)}{n-m+1} & \text{for } i > k. \end{cases} \tag{B.2}$$

i	Position n_i	Score r_i	Gene name	Partial score c_i	$n_i - i$
1	20	5.71685	Cs	0.1063	19
2	65	4.56335	Aco2	0.1904	63
3	129	4.03963	Idh2	0.2641	126
4	193	3.72751	Idh3a	0.3320	189
5	197	3.71779	Mdh1	0.4014	192
6	347	3.26785	Sdhb	0.4581	341
7	384	3.18568	Suclg1	0.5167	377
8	515	2.98120	Pygb	0.5686	507
9	645	2.80126	Mdh2	0.6172	636
10	879	2.52752	Fh1	0.6577	869
11	924	2.47618	Idh3g	0.7027	913
12	1,385	2.14931	Sdhc	0.7295	1,373
13	1,988	1.84017	Sdhd	0.7464	1,975
14	2,248	1.74533	Dlst	0.7715	2,234
15	3,274	1.43489	Pygm	0.7684	3,259
16	3,412	1.40278	Agl	0.7906	3,396
17	3,479	1.38664	Pygl	0.8146	3,462
18	5,296	1.09616	Gaa	0.7822	5,278
19	5,516	1.06895	Sucla2	0.7958	5,497
20	7,029	0.91375	Suclg2	0.7688	7,009
21	14,574	0.50044	Gys1	0.5582	14,553
22	18,262	0.38126	Idh1	0.4579	18,240
23	25,104	0.20227	G6pc	0.2622	25,081
24	27,062	0.15848	Aco1	0.2081	27,038
25	28,668	0.12378	Sdha	0.1636	28,643
26	31,025	0.06981	Pck1	0.0962	30,999

Table B.1: Example gene set S of length $m = 26$, subset of a gene set X of length $n = 34,327$. The enrichment score $E_S \approx 0.8146$ is attained for the 17th gene ("Pygl").

No. removals	Removed elements (index)	Max. element	Enrichment score
1	18	#17 (Pygl)	0.8338
2	18,19	#17 (Pygl)	0.8533
3	18,19,20	#17 (Pygl)	0.8706
4	18,19,20,21	#17 (Pygl)	0.8803
5	18,19,20,21,22	#17 (Pygl)	0.8879
6	15,16,17,18,19,20	#14 (Dlst)	0.9038

Table B.2: Optimal subsets for the example of Table B.1. Note the non-local behavior in the last line, when six elements are removed.

From here on, we will always make the following assumption about the relative sizes of the elements X and S:

Assumption 1. *The number of elements in S is assumed to be smaller than the number of elements from X after the last element S_m, i.e.,*

$$n - n_m > m. \tag{B.3}$$

The following observation is the key to the solution of the problem:

Lemma 2 (Unimodality property)**.** *It holds that*

$$c_i^{(1)} < \ldots < c_i^{(i-1)} < c_i < c_i^{(i+1)} > c_i^{(i+2)} > \ldots > c_i^{(m)} > c_i \tag{B.4}$$

In particular,

(i) $c_i^{(k)} < c_i$ *if* $k < i$*. Moreover, in this case* $c_i^{(k)}$ *is maximal for* $k = i - 1$*.*

(ii) $c_i^{(k)} > c_i$ *if* $k > i$*. Moreover, in this case* $c_i^{(k)}$ *is maximal for* $k = i + 1$*.*

Proof. The inequalities in (*i*) and (*ii*) are clear by inspection of (B.2) and Assumption (B.3), since the latter implies that $(n - n_i) - (m - i) > 0$ for all i.

If $k > i$, then the second term in (B.2) does not depend on k. The first term is maximal for $k = i + 1$, since $f(x) = x/(1 - x)$ is an increasing function of its argument, and $x_k = r_k / \sum_l r_l$ is decreasing in k by assumption, i.e., $f(r_k)$ is maximal for the smallest $k > i$. The monotonicity of $f(x_k)$ also implies the inequalities to the left of c_i in (B.4).

Analogously, if $k < i$, note the negative sign and that $f(r_k)$ is minimal for the largest $k < i$. The monotonicity of $f(r_k)$ also implies the inequalities to the right of c_i in (B.4). □

The following result immediately leads to an efficient algorithm by backward induction:

Proposition 1. *Assume $c_i^{(A)}$ is the optimal k-removal subset of S, i.e., $A \subseteq S$ with $|A| = k$, $s_i \notin A$ and such that*

$$c_i^{(A)} = \max_{|K|=k, s_i \notin K} c_i^{(K)}.$$

Denote the indices of the elements from A in S by $a_1, a_2, \ldots, a_k$ and let them be ordered such that $a_1 < a_2 < \cdots < a_k$. Then:

(i) If $i + k \leq m$ then $a_j = s_{i+j}$ for all j.

(ii) Otherwise, let $j = m - i$. Then

$$a_1 = m - k, a_2 = m - k + 1, \ldots, a_{k-j} = i - 1$$

and

$$a_{k-j+1} = i + 1, a_{k-j+2} = i + 2, \ldots, a_k = m.$$

Proof. This follows immediately from Lemma 2 by induction on k. □

The optimal k-removal subset enrichment score can then be found in time $O(m)$ by calculating the optimal k-removal subset scores $c_i^{(K)}$, $|K| = k$, for all $i \leq m$. The algorithm is given as Algorithm 10.

Algorithm 10 k-removal subset enrichment score

$n, m, k, \{r_1, \ldots, r_m\}, \{n_1, \ldots, n_m\}$

for $i \in \{1, \ldots, m\}$ **do**
  $T \leftarrow \{1, \ldots, m\} \setminus \{i\}$
  for $j \in \{1, \ldots, k\}$ **do**
    $T_+ = \{t \in T \mid t > i\}$
    if $T_+ \neq \emptyset$ **then**
      $t_0 \leftarrow \min_{t \in T_+} t$
    else
      $T_- = \{t \in T \mid t < i\}$
      $t_0 \leftarrow \max_{t \in T_-} t$
    end if
    $T \leftarrow T \setminus \{t_0\}$
  end for
  $R \leftarrow \{1, \ldots, m\} \setminus (\{i\} \cup T)$
  $R_i \leftarrow \{1, \ldots, i-1\} \setminus T$
  $c_i \leftarrow \frac{\sum_{j=1}^{i} r_j - \sum_{j \in R_i} r_j}{\sum_{l=1}^{m} r_l - \sum_{j \in R} r_j} - \frac{n_i - i + |R_i|}{n - m + k}$
end for
return $\max_{1 \leq i \leq m} c_i$

Appendix C

Fantom User Manual

Name	Frequent pAtterN Tree-based Ontology Miner (FANTOM)
Description	Fantom mines gene sets and combines them with ontologies, e.g., GO and KEGG, as well as interaction data, to find interesting combinations of ontological concepts
Inputs	Ranked gene list Score function Minimum participation threshold Minimum score threshold
Outputs	Clustered list of rules, whereby a rule is a set of identifiers described by a conjuction of ontological concepts accompanied by a score that indicates its interestingness. Example: Score: 0,552 Participants: [set of identifiers here] All genes in the subgroup have the following properties: biological_process(positive regulation of apoptosis), KEGG_pathway(Signaling Molecules and Interaction)
Features	Interaction rule generation Rule pruning and clustering Control over ontology specificity Available as web service

Manual

In this appendix we present a user manual for the Fantom application, which is a graphical user interface (GUI) for the Fantom web service. Note that all inputs in the GUI are the same as for the web service, and are merely collected and forwarded to the web service. The GUI is presented in Figure C.1, and consists of seven parts, which will be discussed below.

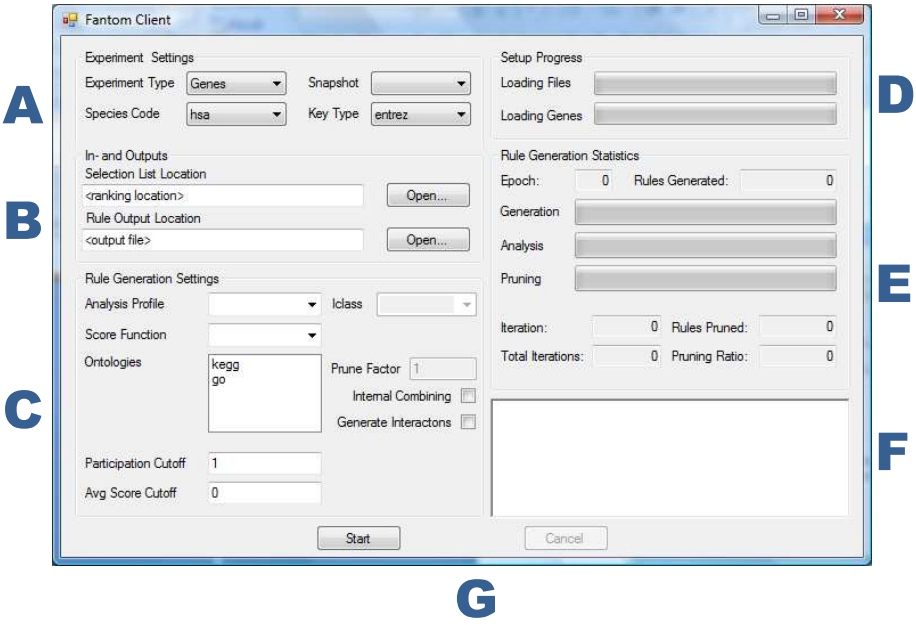


Figure C.1: Screenshot of the Fantom GUI for Windows

Section A

Section A contains all inputs related to global experiment settings, which consist of the following inputs:

- **Experiment Type**
  The experiment type indicates what type of experiment is being conducted. In case of Figure C.1 it would be a Gene mining experiment. The experiment type determines which ontologies can be selected, which key types can be used, and which species can be selected.

- **Species Code**
  The species code indicates what type of species or organism is being studied in

the experiment. The name species is very specific for the life-sciences domain, but its function remains the same across all domains. For different species there can be different instances of the same ontology, or different mappings to ontologies. The species code, or entity code to be more general, helps to select the correct mapping or ontology instance.

- **Snapshot**
  Mappings, interaction data and ontologies are subjective to change and updates. To support automatic updating, a service was created to update all ontologies and mappings daily, while saving old versions in backup folders. If the user needs to recreate an experiment, the date of the experiment can be selected in the snapshot field, so that the correct mapping and ontology versions are selected.

- **Key Type**
  There can be multiple names associated with an identifier in a ranked list. For example, genes can be identified by an ENTREZ identifier, a HUGO Symbol designation or various other names. Since the Fantom service prefers to keep one single identifier type for each domain, other identifier types can be mapped to this domain identifier. By selecting the correct key type that is used in the ranked list, the correct mapping to the domain identifier can be selected.

Section B

Section B contains the user-defined inputs for the ranked list location and the target location of the output file. Upon starting the experiment, the ranked list will be loaded and transferred to the web service.

Section C

Section C contains all experiment-specific parameters. These include:

- **Analysis Profile and Iclass**
  The analysis profile indicates if the experiment is a single-class experiment or a multi-class experiment. In case of a multi-class experiment, all class labels are listed in the Iclass list, where the user can select the class of interest.

- **Score Function**
  The score function parameter allows the user to select the score function for scoring rules. By default the Enrichment Score function is selected. Note that the analysis profile influences the score functions that can be selected.

- **Ontologies and Prune Factor**
  In the Ontologies list, all ontologies compatible with the experiment type are

listed, and the user can select one or more that need to be involved in the experiment. After that, the user can insert the prune factor, which determines the relative minimum depth that the ontological concepts within an ontology must have in order to be viable to participate in the experiment. This number ranges between 0 and 1, whereby 0 allows for all concepts to be considered, while 1 allows only the leaf concepts to be considered.

- **Internal Combining**
  The internal combining option allows the user to generate rules that contain multiple entries of the same ontological predicate. If this is option is unchecked, each rule may only contain at most one concept of each predicate. Leaving this option unchecked is a good way to explore the rule space first, and to tweak thresholds before performing a more detailed experiment.

- **Generate Interactions**
  This option allows the user to generate interaction association rules instead of direct association rules. Note that checking this option considerably slows down an experiment, and requires more memory.

- **Participation and Score Cutoff**
  These thresholds allow the user to put constraints on the generated rules. The participation threshold allows the user to set the minimum number of identifiers per rule, while the score threshold gives the user some control over the quality of the rules that are included in the output.

Section D

Section D provides feedback on the loading phase of the Fantom service. The first progress bar indicates how the loading of all files that Fantom depends on (ranking, identifier mappings, interactions, ontologies) is progressing. The second progress bar indicates the progress of associating identifiers with ontological terms. Since in Figure C.1 the experiment type is set to ”Genes”, the label indicates the progress of associating genes to ontological concepts.

Section E

Section E contains most feedback statistics for the experiment part of the Fantom service. A number of statistics are presented to the user:

- **Epoch**
  The epoch denotes the number of iterations that have been done in the main function of the Fantom service. It also indicates the number of conjunctions of terms in the rule candidates that are generated at that time.

- **Rules Generated**
  The total amount of rule candidates generated so far.

- **Generation**
  A progress bar indicating the progress of rule candidate generation in the current epoch.

- **Analysis**
  A progress bar indicating the progress of scoring the rule candidates generated in the current epoch.

- **Pruning**
  A progress bar indicating the progress of pruning the rule candidates generated in the current epoch. Note that this bar will go from 0 to 100% twice; first for the horizontal cohort pruning stage, and then for the longitudinal cohort pruning stage.

- **Iteration and Total Iterations**
  These are numerical representations of the progress bar of the active stage, being either generation, analysis or pruning. It is meant to be extra feedback for the user to see exactly how many elements need to be handled in a stage, and how many have already been dealt with.

- **Rules Pruned and Pruning Ratio**
  The rules pruned field indicates how many of the generated rule candidates have been pruned so far. Reasons why rules are pruned is that they lack sufficient identifier support, that they do not have an score that is high enough, or that a more specific rule with a higher score has been generated. The pruned ratio is the ratio of rule candidates that have been pruned with respect to the total amount of rule options generated.

Section F

Section F contains a field in which additional feedback is provided for the user, such as error messages or warnings.

Section G

Section G contains the control buttons. They allow the user to start an experiment, or cancel it when it is running.

Part V

Miscellaneous

Samenvatting

In dit proefschrift onderzoeken we hoe technieken en nieuwe ontwikkelingen binnen de software engineering gecombineerd kunnen worden met data mining en knowledge discovery, om prestaties van deze activiteiten te verbeteren en het ontwerp van een experiment te vergemakkelijken. Om de lezer in te leiden in beide kennisvelden presenteren we in Hoofdstuk 1 en Hoofdstuk 2 een algemeen overzicht van software engineering en data mining en geven een overzicht van de belangrijkste technologien die in dit proefschrift aan bod komen.

In Hoofdstuk 3 bespreken we hoe data mining, databases en patternbases kunnen worden gentegreerd en gecombineerd tot inductieve databases. Daarbij presenteren wij modellen voor data-integratie en voor het efficint zoeken in en combineren van data binnen inductieve databases, en beargumenteren hoe en waarom web services goed zou passen binnen het kader van inductieve databases. We bespreken enkele experimenten die illustreren hoe een inductieve database werkt, en hoe knowledge discovery kan worden toegepast op een efficinte en zelfs gedistribueerde wijze.

In Hoofdstuk 4 bespreken we hoe web services passen in het kader van scientific workflows. We vergelijken het ad-hoc geconstrueerde procesmodel met het service-oriented georkestreerd procesmodel, en wijzen op de voordelen van laatstgenoemde. Verder presenteren we een model voor het ontwerpen van service-oriented workflows, evenals voor het ontwerp van individuele services. Dit model testen we vervolgens in een reeks experimenten.

In Hoofdstuk 5 passen we de kennis opgedaan in de eerdere hoofdstukken toe op de Fantom service. Deze service gebruikt subgroup discovery om subgroepen te vinden binnen een set van gerangschikte elementen, die elk een score hebben toebedeeld gekregen. De subgroepen van elementen worden beschreven door conjuncties van ontologische concepten en een score om de mate van interessantheid aan te geven. Vervolgens tonen wij in een aantal experimenten aan hoe de service presteert op het gebied van snelheid en hoe nuttig en interessant de behaalde resultaten zijn.

In Hoofdstuk 6 wordt beschreven hoe de Fantom service gecombineerd kan worden met statistische permutatie-technieken. We tonen aan dat door het uitvoeren van meerdere iteraties van de Fantom service op permutaties van de invoer meer regels achterwege gelaten kunnen worden voor de originele invoer. We tonen ook aan dat door de combinatie van Fantom en permutatie-testen nauwkeurigere drempelwaarden verkregen kunnen worden in het geval van multi-class problemen.

In Hoofdstuk 7 bespreken we de eerste toepassing van de Fantom service. We bespreken experimenten die zijn uitgevoerd op microarray expressie data verkregen van muizen met cardiale overexpressie van de transcriptiefactor TBX3. We voeren meerdere experimenten uit, en voor elk van deze experimenten bespreken we de prestaties en de resultaten. Waar mogelijk hebben we ook de resultaten vergeleken met de uitvoer van de DAVID tool.

In Hoofdstuk 8 passen we de Fantom service toe op gegevens die afkomstig zijn

van een single nucleotide polymorphism onderzoek gedaan op menselijke depressie. Wij voeren twee verschillende soorten experimenten uit en presenteren prestaties van de Fantom service voor beide, alsmede statistieken voor regels en clusteren.

Naast de voorgenoemde acht hoofdstukken kent dit proefschrift ook drie appendices. In Appendix A bespreken we het formaat van de diverse inputs waar Fantom van afhankelijk is. We bespreken de formaten van de inputlijsten, associatiebestanden (”mappings”), ontologieën en de interface van de score-functies. In Appendix B presenteren we de wiskundige achtergron van het Enrichment Score algoritme. We definiëren de wiskundige eigenschappen en presenteren een algoritme om de maximaal mogelijke score voor een subset van een regel te berekenen. Afsluitend presenteren we in Appendix C een handleiding voor de Fantom service.

Acknowledgements

Dr. Michael Muskulus, without whom the optimization of the ES score and its mathematical proof would not have been possible.

Dr. Peter-Bram ’t Hoen and Dr. Tineke van Veen, who have provided me with interesting research questions and enough data to complete my thesis.

Dr. Jelle Goeman, who as an invaluable help in statistical analysis as well as generating ideas for improvement of the Fantom algorithm.

Prof. Dr. Nada Lavrač and Dr. Igor Trajkovski, for providing me with inital ideas and encouraging my research efforts in service orientation.

Drs. Pascal Haazebroek, Drs. Martijn Cox and Dr. Tim Cocx, who provided valuable input over the years, as well as many valuable distractions.

Vid Podpečan, Matjaž Juršič, Petra Kralj Novak, Ivica Slavkov, Darko Cerepnalkoski, Dragi Kocev and Katerina Taskova, for making my visits to Slovenia even more worthwhile.

Finally, the author would like to thank his family and friends for their continuing support.

Curriculum Vitae

Personalia

Full name	Jeroen Sebastiaan de Bruin
Date of birth	12-05-1981
Place of birth	Rotterdam
Nationality	Dutch

Education

M.Sc. in Computer Science, (Graduated Cum Laude) Leiden University, Leiden, the Netherlands	September 2005
Courses in Psychology Leiden University, Leiden, The Netherlands	September – April 2004
Courses in Pattern Recognition Technische Universiteit Delft, Delft, the Netherlands	May – June 2006

Experience

RA in Software Engineering and Formalization, LIACS	2005
TA for Programming Methods, LIACS	2002 – 2009
TA for Operating Systems, LIACS	2003
TA for Multimedia Systems, LIACS	2004
TA for Compiler construction, LIACS	2005
TA for Data Mining, LIACS	2006
TA for Programming & Correctness, LIACS	2004-2007
TA for Concepts of Programming Languages, LIACS	2005 – 2007
TA for Fundamental Informatics III, LIACS	2007 – 2009
TA for Theory of Concurrency 3, LIACS	2007 – 2009

Current Occupation

Currently working as a PostDoc on process and workflow construction in FTICR- and high throughput proteomics at the LUMC in Leiden.

Titles in the IPA Dissertation Series since 2005

E. Ábrahám. *An Assertional Proof System for Multithreaded Java -Theory and Tool Support-* . Faculty of Mathematics and Natural Sciences, UL. 2005-01

R. Ruimerman. *Modeling and Remodeling in Bone Tissue*. Faculty of Biomedical Engineering, TU/e. 2005-02

C.N. Chong. *Experiments in Rights Control - Expression and Enforcement*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-03

H. Gao. *Design and Verification of Lock-free Parallel Algorithms*. Faculty of Mathematics and Computing Sciences, RUG. 2005-04

H.M.A. van Beek. *Specification and Analysis of Internet Applications*. Faculty of Mathematics and Computer Science, TU/e. 2005-05

M.T. Ionita. *Scenario-Based System Architecting - A Systematic Approach to Developing Future-Proof System Architectures*. Faculty of Mathematics and Computing Sciences, TU/e. 2005-06

G. Lenzini. *Integration of Analysis Techniques in Security and Fault-Tolerance*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-07

I. Kurtev. *Adaptability of Model Transformations*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-08

T. Wolle. *Computational Aspects of Treewidth - Lower Bounds and Network Reliability*. Faculty of Science, UU. 2005-09

O. Tveretina. *Decision Procedures for Equality Logic with Uninterpreted Functions*. Faculty of Mathematics and Computer Science, TU/e. 2005-10

A.M.L. Liekens. *Evolution of Finite Populations in Dynamic Environments*. Faculty of Biomedical Engineering, TU/e. 2005-11

J. Eggermont. *Data Mining using Genetic Programming: Classification and Symbolic Regression*. Faculty of Mathematics and Natural Sciences, UL. 2005-12

B.J. Heeren. *Top Quality Type Error Messages*. Faculty of Science, UU. 2005-13

G.F. Frehse. *Compositional Verification of Hybrid Systems using Simulation Relations*. Faculty of Science, Mathematics and Computer Science, RU. 2005-14

M.R. Mousavi. *Structuring Structural Operational Semantics*. Faculty of Mathematics and Computer Science, TU/e. 2005-15

A. Sokolova. *Coalgebraic Analysis of Probabilistic Systems*. Faculty of Mathematics and Computer Science, TU/e. 2005-16

T. Gelsema. *Effective Models for the Structure of pi-Calculus Processes with Replication*. Faculty of Mathematics and Natural Sciences, UL. 2005-17

P. Zoeteweij. *Composing Constraint Solvers*. Faculty of Natural Sciences,

Mathematics, and Computer Science, UvA. 2005-18

J.J. Vinju. *Analysis and Transformation of Source Code by Parsing and Rewriting*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2005-19

M.Valero Espada. *Modal Abstraction and Replication of Processes with Data*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2005-20

A. Dijkstra. *Stepping through Haskell*. Faculty of Science, UU. 2005-21

Y.W. Law. *Key management and link-layer security of wireless sensor networks: energy-efficient attack and defense*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-22

E. Dolstra. *The Purely Functional Software Deployment Model*. Faculty of Science, UU. 2006-01

R.J. Corin. *Analysis Models for Security Protocols*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-02

P.R.A. Verbaan. *The Computational Complexity of Evolving Systems*. Faculty of Science, UU. 2006-03

K.L. Man and R.R.H. Schiffelers. *Formal Specification and Analysis of Hybrid Systems*. Faculty of Mathematics and Computer Science and Faculty of Mechanical Engineering, TU/e. 2006-04

M. Kyas. *Verifying OCL Specifications of UML Models: Tool Support and Compositionality*. Faculty of Mathematics and Natural Sciences, UL. 2006-05

M. Hendriks. *Model Checking Timed Automata - Techniques and Applications*. Faculty of Science, Mathematics and Computer Science, RU. 2006-06

J. Ketema. *Böhm-Like Trees for Rewriting*. Faculty of Sciences, VUA. 2006-07

C.-B. Breunesse. *On JML: topics in tool-assisted verification of JML programs*. Faculty of Science, Mathematics and Computer Science, RU. 2006-08

B. Markvoort. *Towards Hybrid Molecular Simulations*. Faculty of Biomedical Engineering, TU/e. 2006-09

S.G.R. Nijssen. *Mining Structured Data*. Faculty of Mathematics and Natural Sciences, UL. 2006-10

G. Russello. *Separation and Adaptation of Concerns in a Shared Data Space*. Faculty of Mathematics and Computer Science, TU/e. 2006-11

L. Cheung. *Reconciling Nondeterministic and Probabilistic Choices*. Faculty of Science, Mathematics and Computer Science, RU. 2006-12

B. Badban. *Verification techniques for Extensions of Equality Logic*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2006-13

A.J. Mooij. *Constructive formal methods and protocol standardization*. Faculty of Mathematics and Computer Science, TU/e. 2006-14

T. Krilavicius. *Hybrid Techniques for Hybrid Systems*. Faculty of Electrical

Engineering, Mathematics & Computer Science, UT. 2006-15

M.E. Warnier. *Language Based Security for Java and JML*. Faculty of Science, Mathematics and Computer Science, RU. 2006-16

V. Sundramoorthy. *At Home In Service Discovery*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-17

B. Gebremichael. *Expressivity of Timed Automata Models*. Faculty of Science, Mathematics and Computer Science, RU. 2006-18

L.C.M. van Gool. *Formalising Interface Specifications*. Faculty of Mathematics and Computer Science, TU/e. 2006-19

C.J.F. Cremers. *Scyther - Semantics and Verification of Security Protocols*. Faculty of Mathematics and Computer Science, TU/e. 2006-20

J.V. Guillen Scholten. *Mobile Channels for Exogenous Coordination of Distributed Systems: Semantics, Implementation and Composition*. Faculty of Mathematics and Natural Sciences, UL. 2006-21

H.A. de Jong. *Flexible Heterogeneous Software Systems*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2007-01

N.K. Kavaldjiev. *A run-time reconfigurable Network-on-Chip for streaming DSP applications*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-02

M. van Veelen. *Considerations on Modeling for Early Detection of Abnormalities in Locally Autonomous Distributed Systems*. Faculty of Mathematics and Computing Sciences, RUG. 2007-03

T.D. Vu. *Semantics and Applications of Process and Program Algebra*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2007-04

L. Brandán Briones. *Theories for Model-based Testing: Real-time and Coverage*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-05

I. Loeb. *Natural Deduction: Sharing by Presentation*. Faculty of Science, Mathematics and Computer Science, RU. 2007-06

M.W.A. Streppel. *Multifunctional Geometric Data Structures*. Faculty of Mathematics and Computer Science, TU/e. 2007-07

N. Trčka. *Silent Steps in Transition Systems and Markov Chains*. Faculty of Mathematics and Computer Science, TU/e. 2007-08

R. Brinkman. *Searching in encrypted data*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-09

A. van Weelden. *Putting types to good use*. Faculty of Science, Mathematics and Computer Science, RU. 2007-10

J.A.R. Noppen. *Imperfect Information in Software Development Processes*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-11

R. Boumen. *Integration and Test plans for Complex Manufacturing Systems*. Faculty of Mechanical Engineering, TU/e. 2007-12

A.J. Wijs. *What to do Next?: Analysing and Optimising System Behaviour in Time*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2007-13

C.F.J. Lange. *Assessing and Improving the Quality of Modeling: A Series of Empirical Studies about the UML*. Faculty of Mathematics and Computer Science, TU/e. 2007-14

T. van der Storm. *Component-based Configuration, Integration and Delivery*. Faculty of Natural Sciences, Mathematics, and Computer Science,UvA. 2007-15

B.S. Graaf. *Model-Driven Evolution of Software Architectures*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2007-16

A.H.J. Mathijssen. *Logical Calculi for Reasoning with Binding*. Faculty of Mathematics and Computer Science, TU/e. 2007-17

D. Jarnikov. *QoS framework for Video Streaming in Home Networks*. Faculty of Mathematics and Computer Science, TU/e. 2007-18

M. A. Abam. *New Data Structures and Algorithms for Mobile Data*. Faculty of Mathematics and Computer Science, TU/e. 2007-19

W. Pieters. *La Volonté Machinale: Understanding the Electronic Voting Controversy*. Faculty of Science, Mathematics and Computer Science, RU. 2008-01

A.L. de Groot. *Practical Automaton Proofs in PVS*. Faculty of Science, Mathematics and Computer Science, RU. 2008-02

M. Bruntink. *Renovation of Idiomatic Crosscutting Concerns in Embedded Systems*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2008-03

A.M. Marin. *An Integrated System to Manage Crosscutting Concerns in Source Code*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2008-04

N.C.W.M. Braspenning. *Model-based Integration and Testing of High-tech Multi-disciplinary Systems*. Faculty of Mechanical Engineering, TU/e. 2008-05

M. Bravenboer. *Exercises in Free Syntax: Syntax Definition, Parsing, and Assimilation of Language Conglomerates*. Faculty of Science, UU. 2008-06

M. Torabi Dashti. *Keeping Fairness Alive: Design and Formal Verification of Optimistic Fair Exchange Protocols*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2008-07

I.S.M. de Jong. *Integration and Test Strategies for Complex Manufacturing Machines*. Faculty of Mechanical Engineering, TU/e. 2008-08

I. Hasuo. *Tracing Anonymity with Coalgebras*. Faculty of Science, Mathematics and Computer Science, RU. 2008-09

L.G.W.A. Cleophas. *Tree Algorithms: Two Taxonomies and a Toolkit*. Faculty

of Mathematics and Computer Science, TU/e. 2008-10

I.S. Zapreev. *Model Checking Markov Chains: Techniques and Tools*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-11

M. Farshi. *A Theoretical and Experimental Study of Geometric Networks*. Faculty of Mathematics and Computer Science, TU/e. 2008-12

G. Gulesir. *Evolvable Behavior Specifications Using Context-Sensitive Wildcards*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-13

F.D. Garcia. *Formal and Computational Cryptography: Protocols, Hashes and Commitments*. Faculty of Science, Mathematics and Computer Science, RU. 2008-14

P. E. A. Dürr. *Resource-based Verification for Robust Composition of Aspects*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-15

E.M. Bortnik. *Formal Methods in Support of SMC Design*. Faculty of Mechanical Engineering, TU/e. 2008-16

R.H. Mak. *Design and Performance Analysis of Data-Independent Stream Processing Systems*. Faculty of Mathematics and Computer Science, TU/e. 2008-17

M. van der Horst. *Scalable Block Processing Algorithms*. Faculty of Mathematics and Computer Science, TU/e. 2008-18

C.M. Gray. *Algorithms for Fat Objects: Decompositions and Applications*. Faculty of Mathematics and Computer Science, TU/e. 2008-19

J.R. Calamé. *Testing Reactive Systems with Data - Enumerative Methods and Constraint Solving*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-20

E. Mumford. *Drawing Graphs for Cartographic Applications*. Faculty of Mathematics and Computer Science, TU/e. 2008-21

E.H. de Graaf. *Mining Semi-structured Data, Theoretical and Experimental Aspects of Pattern Evaluation*. Faculty of Mathematics and Natural Sciences, UL. 2008-22

R. Brijder. *Models of Natural Computation: Gene Assembly and Membrane Systems*. Faculty of Mathematics and Natural Sciences, UL. 2008-23

A. Koprowski. *Termination of Rewriting and Its Certification*. Faculty of Mathematics and Computer Science, TU/e. 2008-24

U. Khadim. *Process Algebras for Hybrid Systems: Comparison and Development*. Faculty of Mathematics and Computer Science, TU/e. 2008-25

J. Markovski. *Real and Stochastic Time in Process Algebras for Performance Evaluation*. Faculty of Mathematics and Computer Science, TU/e. 2008-26

H. Kastenberg. *Graph-Based Software Specification and Verification*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-27

I.R. Buhan. *Cryptographic Keys from Noisy Data Theory and Applications*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-28

R.S. Marin-Perianu. *Wireless Sensor Networks in Motion: Clustering Algorithms for Service Discovery and Provisioning*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-29

M.H.G. Verhoef. *Modeling and Validating Distributed Embedded Real-Time Control Systems*. Faculty of Science, Mathematics and Computer Science, RU. 2009-01

M. de Mol. *Reasoning about Functional Programs: Sparkle, a proof assistant for Clean*. Faculty of Science, Mathematics and Computer Science, RU. 2009-02

M. Lormans. *Managing Requirements Evolution*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-03

M.P.W.J. van Osch. *Automated Model-based Testing of Hybrid Systems*. Faculty of Mathematics and Computer Science, TU/e. 2009-04

H. Sozer. *Architecting Fault-Tolerant Software Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-05

M.J. van Weerdenburg. *Efficient Rewriting Techniques*. Faculty of Mathematics and Computer Science, TU/e. 2009-06

H.H. Hansen. *Coalgebraic Modelling: Applications in Automata Theory and Modal Logic*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2009-07

A. Mesbah. *Analysis and Testing of Ajax-based Single-page Web Applications*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-08

A.L. Rodriguez Yakushev. *Towards Getting Generic Programming Ready for Prime Time*. Faculty of Science, UU. 2009-9

K.R. Olmos Joffré. *Strategies for Context Sensitive Program Transformation*. Faculty of Science, UU. 2009-10

J.A.G.M. van den Berg. *Reasoning about Java programs in PVS using JML*. Faculty of Science, Mathematics and Computer Science, RU. 2009-11

M.G. Khatib. *MEMS-Based Storage Devices. Integration in Energy-Constrained Mobile Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-12

S.G.M. Cornelissen. *Evaluating Dynamic Analysis Techniques for Program Comprehension*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-13

D. Bolzoni. *Revisiting Anomaly-based Network Intrusion Detection Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-14

H.L. Jonker. *Security Matters: Privacy in Voting and Fairness in Digital Exchange*. Faculty of Mathematics and Computer Science, TU/e. 2009-15

M.R. Czenko. *TuLiP - Reshaping Trust Management*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-16

T. Chen. *Clocks, Dice and Processes*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2009-17

C. Kaliszyk. *Correctness and Availability: Building Computer Algebra on top of Proof Assistants and making Proof Assistants available over the Web*. Faculty of Science, Mathematics and Computer Science, RU. 2009-18

R.S.S. O'Connor. *Incompleteness & Completeness: Formalizing Logic and Analysis in Type Theory*. Faculty of Science, Mathematics and Computer Science, RU. 2009-19

B. Ploeger. *Improved Verification Methods for Concurrent Systems*. Faculty of Mathematics and Computer Science, TU/e. 2009-20

T. Han. *Diagnosis, Synthesis and Analysis of Probabilistic Models*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-21

R. Li. *Mixed-Integer Evolution Strategies for Parameter Optimization and Their Applications to Medical Image Analysis*. Faculty of Mathematics and Natural Sciences, UL. 2009-22

J.H.P. Kwisthout. *The Computational Complexity of Probabilistic Networks*. Faculty of Science, UU. 2009-23

T.K. Cocx. *Algorithmic Tools for Data-Oriented Law Enforcement*. Faculty of Mathematics and Natural Sciences, UL. 2009-24

A.I. Baars. *Embedded Compilers*. Faculty of Science, UU. 2009-25

M.A.C. Dekker. *Flexible Access Control for Dynamic Collaborative Environments*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-26

J.F.J. Laros. *Metrics and Visualisation for Crime Analysis and Genomics*. Faculty of Mathematics and Natural Sciences, UL. 2009-27

C.J. Boogerd. *Focusing Automatic Code Inspections*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2010-01

M.R. Neuhäußer. *Model Checking Nondeterministic and Randomly Timed Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2010-02

J. Endrullis. *Termination and Productivity*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2010-03

T. Staijen. *Graph-Based Specification and Verification for Aspect-Oriented Languages*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2010-04

Y. Wang. *Epistemic Modelling and Protocol Dynamics*. Faculty of Science, UvA. 2010-05

J.K. Berendsen. *Abstraction, Prices and Probability in Model Checking Timed Automata*. Faculty of Science, Mathematics and Computer Science, RU. 2010-06

A. Nugroho. *The Effects of UML Modeling on the Quality of Software*. Faculty of Mathematics and Natural Sciences, UL. 2010-07

A. Silva. *Kleene Coalgebra*. Faculty of Science, Mathematics and Computer Science, RU. 2010-08

J.S. de Bruin. *Service-Oriented Discovery of Knowledge - Foundations, Implementations and Applications*. Faculty of Mathematics and Natural Sciences, UL. 2010-09