Guided Association Mining through Dynamic Constraint Refinement

by Aaron John Ceglar, *BIT.(Hons)*

School of Informatics,
Faculty of Science and Engineering

March 28, 2005

Flinders University of South Australia

in total fulfillment of the requirements for the degree of
Doctor of Philosophy

Adelaide, South Australia, 2005

Abstract

Association mining, the discovery of *interesting* inferences from within a dataset, is ultimately subjective as only the user can assess the practical usefulness of an inference. To this effect, an association mining system harnesses the user's perceptual capabilities and the computer's processing power to improve the quality of a set of inferences. Although current association mining systems tightly involve the user within the pre-processing and presentation stages, the analysis stage of the association mining process remains relatively autonomous and opaque. This lack of user involvement constrains domain space exploration and subsequent inference derivation, potentially reducing inference quality, due to the lack of user-computer synergy.

The theory of guided association mining and its realisation represents a timely and logical step in the progression of association mining research. Early research focused upon algorithmic efficiency, addressing issues such as I/O reduction and scalability, however this seems to have reached a point of diminishing return. The research focus has therefore shifted to improving result quality, or improving inference interest, rather than the speed at which the results are generated, including areas of research such as measures of interestingness and semantic inclusion. However, these areas of research which attempt to incorporate domain knowledge within analysis, fall short of providing user-computer synergy as the specified constraints are statically included within an automated process. Given this static constraint inclusion, the derivation of quality inferences often requires an iterative analysis process, whereby a set of quality inferences is converged upon through iterative constraint refinement.

This thesis argues that by maintaining the user-computer synergy during analysis, the quality of discovered inferences can be improved. This is achieved by opening the opaque "black box" analysis process and providing functionality through which the user can interact, and subsequently guide, domain space exploration. Thus by enabling the user to dynamically focus exploration upon concept areas of specific interest, the quality of the derived inferences will improve.

This thesis addresses the next step in providing *analysis synergy* by enabling the user to dynamically refine constraints during analysis instead of between analysis iterations. To this end a guided mining architecture is proposed that merges the currently accepted knowledge discovery architecture with the model-view-controller architecture, enabling analysis synergy through the provision of a transparent and interactive analysis environment. Furthermore this thesis also makes novel contributions to the foundation fields of analysis and rule presentation, by way of an incremental closed-set association mining algorithm and an association visualisation technique that accommodates hierarchical semantics.

Certification

I certify that this thesis does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person except where due reference is made in the text.

As requested under Clause 14 of Appendix D of the *Flinders University Research Higher Degree Student Information Manual* I hereby agree to waive the conditions referred to in Clause 13(b) and (c), and thus

- Flinders University may lend this thesis to other institutions or individuals for the purpose of scholarly research;
- Flinders University may reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Signed Dated

Aaron John Ceglar

Use of this thesis

I hereby acknowledge that I have been given access to the thesis for consultation only and that no part will be published or paraphrased without the prior consent of the author and that the author's intellectual property rights will be respected.

Acknowledgements

This thesis was inspired by a casual conversation between two gifted scholars: John Roddick and Paul Calder, regarding the potential benefits of incorporating the user within association analysis. This field of research has proved rewarding and I believe the contributions we have made within it are useful. However, my research apprenticeship has been much more than this thesis represents, I believe my true reward is my growth as a person during this time, which I credit to the social environment in which I have been immersed.

I would first like to thank my supervisor, John Roddick for suggesting this area of research, providing invaluable insights, timely encouragement as well as guidance balanced by the freedom to express myself through this work. Most importantly however, I would like to thank John for his patience.

I would like to acknowledge the academic contributions of Paul Calder, David Powers, Scott Vallance, Denise de Vries, Carl Mooney, Darius Pfitzner and Darin Chan. Also to Annette Shillabeer and Vaughan Hobbs for reading my thesis and providing critical feedback.

From a philosophical view I am eternally grateful to Ron Porter, for challenging my philosophical beliefs. I have really enjoyed our heated conversations, you have had a significant impact upon my life.

I would like to thank my colleagues and friends: Darius Pfizner, Trent Lewis, Martin Leurssen, Scott Vallance, Darin Chan, Ron Porter, Denise de Vries, Carl Mooney and Graham Roberts for their support and conversation. I would also like to thank the technical staff Murray, Paul, Tristan, Michael and Rino for their support over the years and Flinders University for the provision of the scholarship that made all this possible.

Most importantly, I would like to thank my family. To Mama, Nathan and Alisha for being there. To Kurtis, Eryn and Corban for always being happy to see me, and to Jo for absolutely everything.

Aaron Ceglar
November 2004

Table of Contents

List of Figures

List of Tables

List of Algorithms

Notation

Set Nomenclature

C	Candidate element subset.
L	Closed lattice.
D	Data set.
E	Element set.
N	GAM model.
Q	GAM queue.
δ	Increment dataset.
I	Increment lattice.
O	Object set.
R	Rule set (inference set).
CL	Set of closed elementsets.
V	Set of valid elementsets.
U	Universal association mining context.

Miscellaneous Nomenclature

$\wedge$	And.
$\vee$	Or.
$\supset$	Superset.
$\supseteq$	Superset or equal.
$\subset$	Subset.
$\subseteq$	Subset or equal.
$\cap$	Intersection.
$\cup$	Union.
$\in$	Exists in.
γ	Confidence.
minconf	Minimum Confidence Threshold.
σ	Support.
minsup	Minimum Support Threshold.
κ	Elementset length.
cl	Elementset closure.
$\Rightarrow$	Infers.
$\Re$	Root Node.
tidList	Object identifier list. Note that *Tid* is inherited from Transaction identifier - a domain specific concept.
ω	The number of increment datasets (δ) specified in the inclusion of windowing functionality.

Acronyms

BFT	Breadth First Traversal.
DFT	Depth First Traversal.
DCR	Dynamic Constraint Refinement.
GUI	Graphical User Interface.
HCI	Human Computer Interaction.
I/O	Input / Output.
LIM	Layered Interaction Model.
MOI	Measures of Interest.
Tid	Transaction Identifier.
UCP	Upward Closure Principle.
MCL	Maintained Closed Lattice analysis algorithm.
CARV	Concentric Association Rule Visualiser.
GAM	Guided Association Mining tool.
HND	Hierarchical Non-monotonic Dynamic analysis algorithm.
HPTid	Hierarchical Prioritised Tidlist analysis algorithm.

Preface

This thesis presents a guided knowledge discovery architecture that facilitates enhanced user-computer synergy within knowledge discovery analysis by providing an interactive analysis environment. Although this architecture has generic connotations, as it is designed to be applicable to all explorative knowledge discovery tasks, the research has been undertaken in the context of association mining, effectively enabling the guidance of association analysis through dynamic constraint refinement. To this end, the thesis builds towards the proposed guided architecture through significant research into the critical foundation areas of analysis and presentation, which has resulted in additional contributions to these areas. The thesis is presented in five logical parts: 1)introduction, 2) association mining, 3) rule presentation, 4) guided association mining and 5) conclusion. Furthermore, for example purposes the thesis uses the simple concept hierarchy presented in Figure 1.

Part I introduces the thesis by providing the problem statement and thesis hypothesis, which is supported by recent statements by prominent researchers regarding the need for further research into interactive analysis. The major areas in which this thesis aims to contribute are then introduced, namely knowledge discovery and association mining, as well as a section that introduces the possible effects of user participation based upon research in the fields of psychology and Human Computer Interaction. The introduction concludes by presenting the general approach of this thesis and addressing issues of terminology.

The next three parts present the thesis contributions, each of which contains a review of the pertinent area and a contribution. Parts II and III present research into the foundation fields of association analysis and rule presentation, while Part

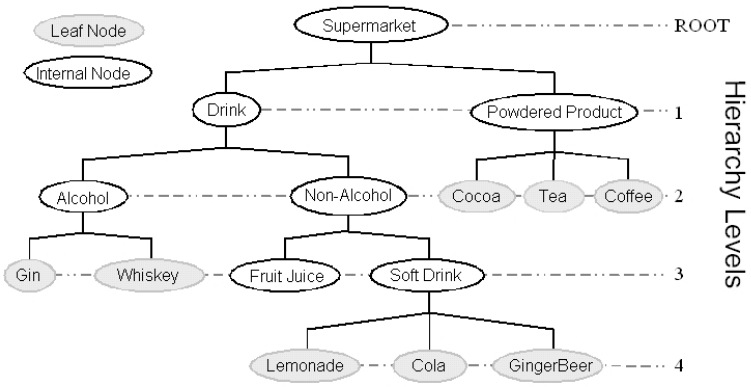


Figure 1: Simple Concept Hierarchy

IV culminates in the presentation of the guided association architecture.

Part II discusses association analysis and is divided into two chapters. The first chapter presents a comprehensive review of current techniques used in the discovery of inferences, focusing upon data structures, traversal strategies and semantic inclusion. The second chapter presents a novel closed set incremental association mining algorithm, MCL, that improves on the state-of-the-art in incremental association mining through the maintenance of a smaller concise representation of the data based upon the concept of closed-sets, defined in Section 1.3.1. Given that knowledge discovery is user centric, reducing the size of the maintained structure facilitates user interpretation. MCL also creates a closed-set representation of the increment dataset, providing the user with insight to the increment's effect upon the maintained lattice and an effective means of incorporating windowing functionality.

Part III discusses the presentation of association rules or inferences and is divided into two chapters. The first presents a review of current presentation techniques, with a focus on graphical visualisation. The second chapter presents CARV, a novel visualisation technique that enables the presentation of inferences within a hierarchical context.

Part IV presents the culmination of this thesis over four chapters, the first two chapters of which are surveys. The first chapter discusses methods by which exploration is constrained within association analysis, presenting a review of current

techniques and identifying the different types of constraints that need to be implemented to realise a holistic guided association analysis environment. The second chapter reviews the current techniques used to enable constraint refinement within a knowledge discovery session, which falls into iterative and interactive refinement. Iterative refinement is discussed in relation to association analysis only, while interactive refinement (or guidance), being central to this thesis, is discussed in relation to the knowledge discovery process itself and in regard to the exploratory tasks of clustering, classification and association mining.

The third chapter of Part IV presents the proposed guided architecture, discussing the role of each architectural component in facilitating user interaction. The final chapter presents GAM, a proof-of-concept tool that, based upon the proposed architecture, provides a guided association mining system that dynamically incorporates the refinement of an example constraint for each constraint class identified (see Chapter 5). The thesis concludes in Part V with a discussion of the thesis contributions, areas of further work and a conclusion.

Publications

The following publications have resulted from material presented within this thesis. Publications 1 and 2 relate to initial research efforts into guided association mining and although much has been superseded, remnants can be found in Chapters 3 and 7. Publication 3 directly relates to material presented in Chapter 1, while publication 4 relates to Chapter 4.

1. Ceglar, A., Roddick, J.F. and Calder, P. (2003), Guiding Knowledge Discovery through Interactive Data Mining in Managing Data Mining Technologies in Organisations: Techniques and Applications, Pendharker, P., IDEA Group Publishing, 45-90.

2. Ceglar, A., Roddick, J.F., Mooney, C.H. and Calder, P. (2003). From Rule Visualisation to Guided Knowledge Discovery. In Proc. Second Australasian Data Mining Workshop (AusDM'03), Canberra. Simoff, S. J., Williams, G. J. and Hegland, M., 59-94.

3. Ceglar, A. and Roddick, J.F (2003), Association Mining, ACM Computing Surveys (submitted), 2003.

4. Ceglar, A., Roddick, J.F., Calder, P and Rainsford, C.P. (2005), Visualising Hierarchical Associations, Knowledge and Information Systems (to appear), 2005.

Part I

Introduction

Instead of allowing an automated data mining process to iterate in a trial-and-error manner, a natural but neglected way to enhance the process is to support human involvement.

Mihael Ankerst, 2002

This thesis argues that current association mining analysis techniques, being autonomous and opaque, are inefficient as they fail to directly involve the user in dataset exploration. The lack of synergy between the user and computer during analysis degrades inference quality, as given a statically constrained search space, many of the derived inferences are of no interest to the user. Interest is thus defined as a subjective quality based upon the practical usefulness of an inference. This thesis therefore discusses the theory, techniques and practice of "guided data mining" a techniques that embeds the user within the analysis stage of the knowledge discovery process in the same way that direct manipulation embeds the user within a graphical user interface.

Hypothesis statement

Maintaining synergy between the user and the computer during association mining analysis can improve result quality.

The need for interactive analysis was first formally presented in a position paper by Mihael Ankerst in 2001 (Ankerst 2001), in which he discussed the advantages of involving the user in the broader context of knowledge discovery analysis, of which association mining is an important task. In particular Ankerst focused upon the associated task of classification analysis or the interactive construction of decision tree classifiers. In the following year Ankerst organised a panel of four leading researchers to discuss the issue of automated vs interactive analysis (Ankerst 2002). The general concensus was that the provision of a flexible interactive analysis environment is an important area of further research within which more work is required. The following quotes are taken from the panel discussion as presented by Ankerst, additional motivating quotes from this discussion are also presented at the beginning of each part of this thesis.

> *Current state of the art data mining tools are automated, but the perfect data mining tool is interactive and highly participatory.*
>
> Georges Grinstein, 2002

> *Data selection and viewing of mining results should be fully interactive, the mining process should be more interactive than the current state of the art and embedded applications should be fairly automated.*
>
> Jiawei Han, 2002

As an introduction, this chapter presents the motivation behind the research and introduces the domain of research exploration. To this end, the following sections introduce the domain of knowledge discovery and the task of association mining. A section on user involvement follows, providing further motivation toward the need for interactive or guided analysis. This part then concludes with a discussion about the general approach used within this thesis and issues of terminology.

I.1 Knowledge Discovery

Knowledge Discovery is the non-trivial process of eliciting interesting knowledge from potentially very large data repositories. The field draws upon the related computer science disciplines of statistics, artificial intelligence, databases and visualisation. A knowledge discovery system combines human perceptual and knowledge-base capabilities with the computational capabilities of computers to discover patterns within datasets that are of practical use. The inclusion of the the user is integral to any knowledge discovery system as it is ultimately the user who decides upon the *interestingness*, or usefulness of a pattern.

The commonly accepted knowledge discovery architecture is presented in Figure I.2. It can be seen that the user is actively involved in the knowledge discovery session, at a high level, through the specification of data sources and the selection of components (or tools) to be used at the different knowledge discovery stages. While user involvement within each component is dependant upon the nature of the tool selected, in current practice the user is tightly integrated in all but the analysis stage, which to date remains relatively autonomous and opaque.

The current architecture follows a typical waterfall model, involving a series of independent steps that may be repeated, allowing constraint refinement, if the

user is unsatisfied with a particular tool's outcome. A knowledge discovery session is instigated through the *collection* of data, from possibly multiple sources, and the *pre-processing* of this into a form acceptable for analysis. The dataset is subsequently *analysed*, discovering patterns (or general descriptions of the data) within the dataset, dependant upon the nature of the knowledge discovery task. These are then *presented* to the user using either textual or graphical representations for subsequent *interpretation* by the user. Depending upon the extent to which these results satisfy the user's goals, refinement and re-processing may be required, from previous process stages.

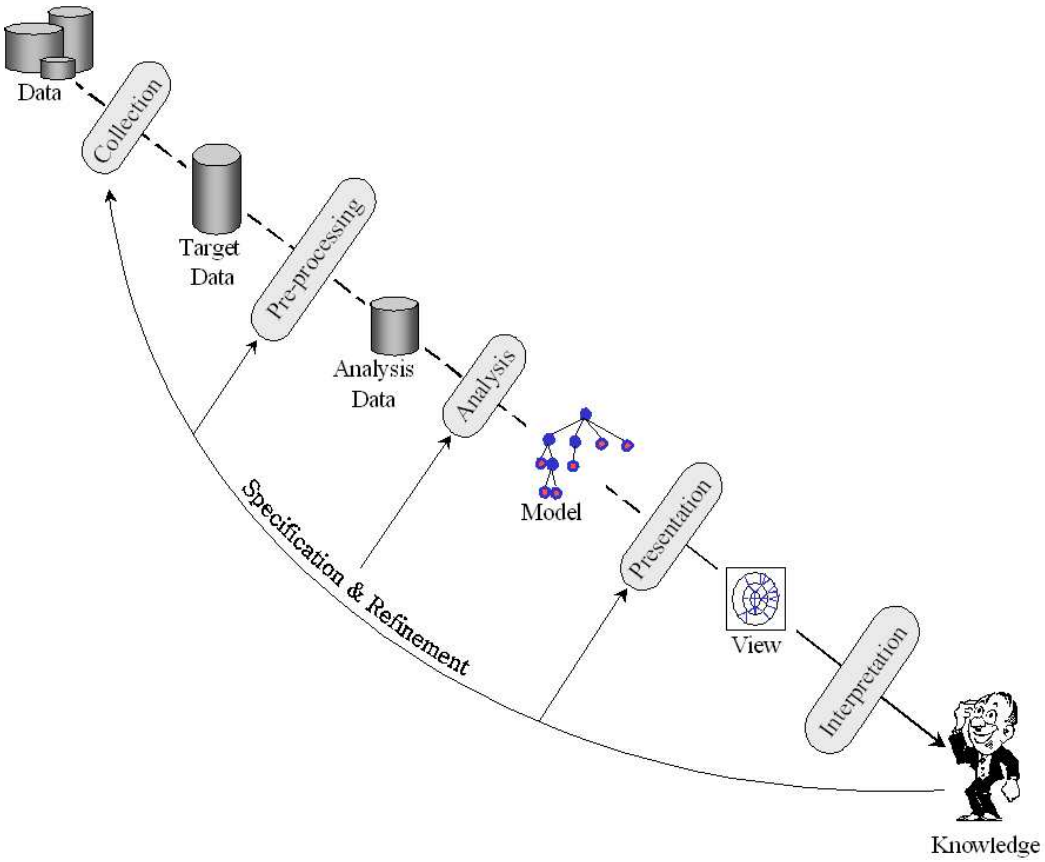


Figure I.2: Knowledge Discovery Architecture

Knowledge discovery functionalities (or tasks) can be classified into two categories - descriptive and predictive - where descriptive techniques characterise general dataset properties, while predictive techniques perform inference upon the dataset to make predictions. Descriptive techniques explore the dataset, generating descriptions through the construction of a model that elicits dataset characteristics depending upon the specific task. Given its basis in artificial intelligence techniques, model construction can be either supervised or unsupervised dependant upon the nature of the task. Association mining, clustering and classification are the three main descriptive tasks within knowledge discovery. While association mining and clustering tasks are unsupervised due to the lack of direction in regard to the nature of outcomes, classification is supervised through the presence of a set of classification labels, the absence of which effectively reduces classification to clustering. For example, within clustering the algorithm derives its own classes, or clusters, while in classification the classes are provided and the task centers upon providing descriptions for the specified classes from the dataset.

Predictive tasks are generally associated with a descriptive task, in that a model of the dataset is first built using a descriptive task that is then used for inference purposes to predict the outcome of a variable given new data based upon the knowledge within the constructed model. The most common example of this is classification, which builds a descriptive model of the dataset, that is then used for predictive purposes.

It is the class of explorative (or descriptive) knowledge discovery tasks that can significantly benefit from the inclusion of guidance during model creation (or analysis), as through guidance, exploration can be dynamically constrained to suit the user's objectives. A review of current techniques that incorporate the the user within exploratory knowledge discovery tasks are presented in Chapter 5. This thesis focuses upon the guidance of association mining analysis, an area in which little research to date has been conducted.

I.2 Association Mining

Association mining is a descriptive knowledge discovery task that discovers inferences of *interest* that exist within objects within a given dataset. The classic use of association mining is in market basket analysis, in which purchasing habits are analysed by discovering the inferences between the different items (or elements) in a person's shopping transaction (or dataset). For example, it may be discovered that people who purchase coffee are 80% likely to also purchase sugar, generally denoted as *coffee* $\Rightarrow$ *sugar* (80%). The discovery of such inferences provides an insight into which items are purchased together and therefore facilitate marketing strategies, such as product placement. Association mining has been applied to many different domains including risk analysis, epidemiology, clinical medicine, fluid dynamics, astrophysics and counter-terrorism, all areas in which the elicitation of inferences between elements within objects is useful.

Association mining is a user-centric process. Its goal is the elicitation of *interesting* dataset inferences, where *interestingness* is a subjective quality ultimately defined by the user. The literature suggests many characteristics that comprise an interesting inference, including: novelty, significance, unexpectedness, non-triviality and actionability (Bayardo & Agrawal 1999, Freitas 1999, Sahar 1999, Silberschatz & Tuzhilin 1996), however these measures can be distilled to one essential characteristic, the practical usefulness of the inference.

Given its user-centric nature, the computer's role is to provide computational power to explore the dataset, or more accurately the *element search space*, from which the inferences are derived. To facilitate the discovery of *interesting* inferences, exploration is restricted through the inclusion of functional and domain-based constraints (see chapter 5). The main research issues within association mining relate to efficient inference discovery and maximising the quality of the resulting inference set as measured by the user. In this regard, quality is viewed as the number of interesting inferences discovered as a proportion of the number of inferences discovered.

$$Inference\ Result\ Set\ Quality \quad = \quad \frac{\#\ interesting\ inferences}{\#\ inferences} \qquad \text{(I.1)}$$

Initial research into association mining analysis focused upon efficient inference discovery through investigation into I/O optimisation and efficient data structures. However, the literature indicates that benefit from these aspects of analysis research is diminishing. This is evident by the apparent change in research focus towards 1) improving inference result quality through the development of specialist algorithms and 2) investigation into constraints and different measures of interestingness that can be included within analysis. This material is covered in depth in Chapter 1.

Measures of interest were initially incorporated as simple heuristics, the most prominent being support (σ), which indicates a concept's level of dataset presence, and confidence (γ), which indicates the strength of an inference. Such measures due to their reflexive nature (see Chapter 1) have been found effective in reducing the exploration space and improving the quality of results. Typically constraints are included as static parameters within an automated and opaque analysis process. This often leads to repeated exploration with refined constraints, as the user seeks to obtain a result set of satisfactory quality. Chapter 5 provides a detailed discussion about association analysis constraint.

I.3 User inclusion

The concept of knowledge-based architectures from HCI and psychology research further motivates the inclusion of the user within analysis. The explorative nature of association mining analysis implies that no hypothesis regarding the inferences contained within the dataset exists prior to analysis. Therefore the initial analysis constraint is based upon, at best, an educated guess, as the user initially has a limited knowledge base regarding the nature of the dataset inferences. The concept of knowledge-based architectures implies that by involving the user in the knowledge discovery process, an implicit communication channel is established. This provides the computer with knowledge of the problem domain and user objectives (see Figure I.3).

Current association analysis techniques are automated and opaque, with inferences being presented to the user upon analysis completion. Through user

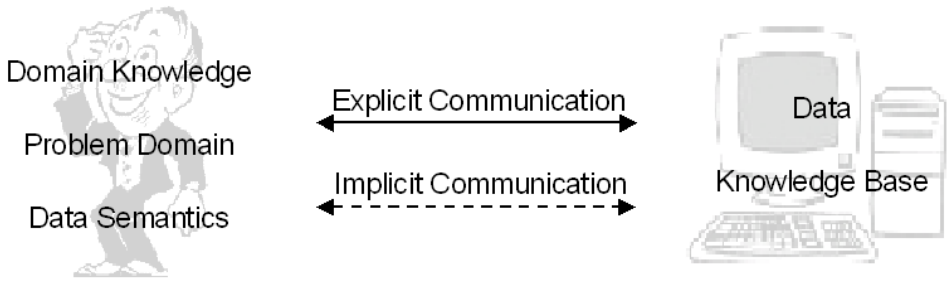


Figure I.3: Knowledge Based HCI

interpretation, semantic understanding of the presented inferences is provided by using the perceptual capabilities of the user, enabling the user to comprehend data semantics and relate them to the current problem, resulting in knowledge-base evolution. If the user is not satisfied with the quality of the discovered inferences, the user can refine constraints and re-instigate analysis, as indicated in Figure I.2, in an attempt to derive an improved inference set.

This current technique is often regarded as a trial-and-error based approach, although each iteration provides opportunity for refinement (not random constraint selection) and should therefore eventually produce a result set of satisfactory quality.

By incorporating the user within the analysis process, knowledge-base evolution becomes finer grained, enabling the semantic understanding of results during analysis which can then be used to guide subsequent analysis to areas of interest. Therefore by enabling guided association mining, the computer's inability to incorporate knowledge about intangible measures such as domain knowledge and data semantics during analysis can be overcome. Theoretically this should not only facilitate the production of higher quality inferences, but also reduce analysis time for two reasons. Firstly, analysis guidance can reduce the exploration space by discarding areas that are of no interest. Secondly, given the current iterative refinement process, guidance through improving result quality will result in reduced iterations and hence reduced analysis time.

Furthermore, psychological research has identified a relevant phenomena known as the Hawthorn Effect, that states that "people tend to work harder when they sense that they are participating in something new, or in something

in which they have more control" (Mayo, 1945). This indicates that the inclusion of the user within analysis will promote better work ethics, as the user feels in control of the process. It is also likely that the user will develop a better understanding and a higher level of trust in the resultant inferences due to this increased control.

I.4 Approach

It is the proposition of this thesis that by allowing the user to guide exploration of the search space through the dynamic refinement of constraints, that analysis time will be reduced. The basis of this thesis relies upon the subjective inclusion of the user within association mining analysis resulting in a qualitative analysis of the hypothesis, as the effect of the user's inclusion cannot be quantified. Therefore the approach taken is to present an architecture that facilitates guidance and to provide a proof-of-concept tool, GAM, that uses this guidance enabled architecture to provide an interactive analysis environment (see Chapter 8).

Such architecture requires an increased coupling of the analysis and presentation stages of the Knowledge Discovery process in order to allow for the creation of analysis interfaces. Through these interfaces the exploration process is dynamically presented to the user and by feeding the user's interactions with the interface back into analysis, guidance is provided. The provision of this functionality will effectively "open up" the traditionally automated and opaque analysis process, and provide a semi-automated, transparent analysis environment with which the user can interact.

It is the intention of this thesis that the proposed guided architecture be generic in its applicability to all descriptive knowledge discovery tasks, such as clustering and classification.

It is not the intention of this thesis to provide quantitative proof of the effectiveness of guided association mining. In fact, due to its subjective premise and the exploratory nature of the algorithm, effective quantitative proof of the hypothesis is not achievable, due to subjective variation and the requirement in association mining for an initial null hypothesis. This thesis however does provide

strong evidence regarding the benefit of guidance inclusion within an association mining framework.

I.5 Notation

The foundation of association mining in the domain of market basket analysis is still evident in current research through the continued use of domain specific terminology. This thesis adopts a more generic terminology to provide a domain independent discussion. Although the adoption of generic terminology is gaining favour in more recent publications, the adopted terms are presented here for clarification.

In the domain of market basket analysis, each customer purchase is represented as a *transaction* that consists of multiple *itemsets*. Therefore association analysis explores the *item* space, incorporating the frequency metric, *support*, as a constraint (see Chapter 1) discovering the *frequent itemsets*, from which the association rules are subsequently derived. Although representative of market basket analysis, association mining has outgrown this single domain application. It is now used in the generic discovery of inferences in any domain, from medical to fluid dynamics, within which this terminology is illogical.

The intuitive replacement for *transaction* is the term *object*, while *element* and *elementset* replace *item* and *itemset*, due to their common use within set theory and their generic applicability. The term *frequent* is specific to the support constraint and therefore has been replaced with the constraint independent term *valid*, which relates to the validity of the rule according to the specified constraints.

Furthermore it is commonly regarded that the result of association mining is the derivation of association rules, however a more applicable term is "inferences" which better describes the nature of the resultant artefacts and which has been adopted within this thesis. Exceptions are made to this terminological replacement in Chapter 1, where the original terminology has been kept in some cases for purposes of clarity and accuracy in presenting previous research.

The issue of terminology may be regarded as trivial, however from a broad perspective, especially from an external viewpoint, naming facilitates understanding, and hence the use of correct terminology is important.

Part II

Association Mining

My view of an attractive data mining (analysis) tool is not a fully automated one, but a user friendly, interactive one, using a high-level graphical user interface to specify and control mining primitives as well as various kinds of visualisation tools.

Jiawei Han 2002

Chapter 1

Association Mining Analysis Review

A decade on from the seminal work of Agrawal, Imielinski and Swami (1993) association mining analysis has become a mature field of research. This survey provides insight to the significant contributions within this field, highlighting the maturity of the fundamental principles within elementset identification and inference generation. Furthermore it provides additional investigation into other established areas of specialisation such as condensed representation, incomplete sets and semantic inclusion.

The fundamentals of association mining are now well established and there appears little current research involving the performance optimisation of classic elementset identification or inference generation in regard to novel exploration techniques. Exceptions to this closure of classic association mining analysis appears to be the adaptation of Inductive Logic Programming (ILP) (Deshaspe & Toivonen 1998) techniques to the field of association mining and the development of parallel and distributed variations of established elementset identification algorithms (Zaki 1999). This survey thus investigates the fundamental principles of association mining analysis.

The majority of current related research involves the specialisation of the fundamental association mining algorithms, while many of these are covered within this review, some specialisations that are still emerging are not presented. These

include areas such as quantitative mining, disassociation mining and higher-order mining. Further research is devoted to user involvement within the analysis process, which is the focus of Part III and the foundation of this thesis.

Association mining analysis is a two part process; 1) the identification of sets of *elements* or *elementsets* within the dataset and 2) the subsequent derivation of *inferences* from these elementsets. The majority of related research to date has focused upon the efficient discovery of elementsets, as its level of complexity is significantly greater than that of inference generation. Given E distinct elements within the search space, there are $2^{|E|}$ possible combinations of elements to explore, given that $|E|$ is often large, naive exploration techniques are often intractable.

To date relevant research has focused upon exploration constraint, I/O reduction and the creation of useful data structures to make analysis more tractable. Exploration constraint has been pursued through the development and incorporation of measures of *interest* (MOI) and efficient pruning strategies. I/O reduction has been facilitated recently through hardware advances, enabling large datasets to become memory resident. Despite this performance still remains an issue for very large datasets, however techniques such as intelligent sampling are being researched to address this issue (see section 1.3.2). Data structure research, initially driven by the need to reduce I/O, has resulted in an evolution of structures to efficiently represent the exploration space. Another line of research involves the production of condensed inference sets from which the entire result set may be inferred, reducing storage and facilitating user interpretation.

Over the past decade a variety of algorithms have been developed that address these issues through the refinement of search strategies, pruning techniques, data structures and the use of alternative dataset organisations. While most algorithms focus upon the explicit discovery of all inferences for a given dataset, increasing consideration is being given to specialised algorithms, that attempt to improve processing time or facilitate user interpretation by reducing result sets and incorporating domain knowledge.

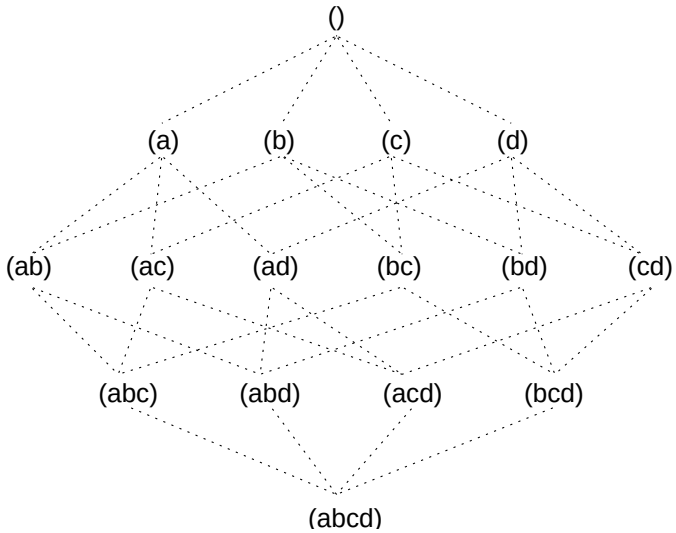


Figure 1.1: Search space lattice

This chapter presents a comprehensive survey of current association mining analysis algorithms organised in the following way. As the majority of research to date has focused upon the identification of elementsets, which are valid in respect to specified constraints, it forms the bulk of the review and has been separated into three sections. The first, elementset identification, Section 1.1, provides the foundation for the review introducing general concepts, presenting common notation and discussing dataset organisation. This is followed by a comprehensive survey of classic algorithms in Section 1.2 and a survey of specialised algorithms in Section 1.3. Each main section is accompanied by an algorithm summarisation table. Section 1.4 completes the survey with a discussion on the derivation of inferences from the identified valid elementsets, including techniques regarding rule inferencing.

1.1 Elementset Identification

The identification of valid elementsets is computationally expensive, requiring the consideration of all combinations of E, or $2^{|E|}$ subsets, resulting in exponential search space growth as $|E|$ increases linearly. This is illustrated in Figure 1.1, which shows the search space lattice resulting from $E = \{a, b, c, d\}$. Further-

more it is the consideration of this lattice in the presence of a specific dataset D that results in an often intractable analysis scenario due to the size of the required structures and the fine grained exploration required. Therefore elementset research focuses upon reducing dataset I/O and exploration constraint to optimise analysis. There are four applicable classes of I/O reduction suggested in the literature: projection, partitioning, pruning and access reduction (presented below).

Projection The projection of D onto an equivalent condensed representation reduces storage requirements possibly enabling memory residency and may also result in computation optimisation through algorithmic exploitation of this new representation. For example, the projection of a corpus into integer or binary representation may result in improved processing time as numeric comparison is computationally faster than string comparison.

Partitioning Dataset partitioning minimises I/O costs by enabling the memory resident processing of large datasets, reducing costly disk access.

Pruning Dataset pruning techniques dynamically reduce the dataset during processing, discarding unnecessary elements or objects, this results in a reduction of D, possibly to the point at which D can achieve memory residency, further reducing processing time.

Access Reduction Reduction in the number of times that disk resident datasets need to be accessed to identify all elementsets, reducing processing time.

Constraint of the search space through user specified heuristic and domain constraints can significantly reduce exploration while improving the quality or interest of the resultant inferences. A detailed review and discussion of constraint inclusion is provided in Chapter 5. The most common constraint used to reduce exploration during elementset identification is support (σ) which denotes the degree of presence of an elementset, (e) within D (such that $e \subset E$), above which the elementset is considered significant and below which the elementset is removed from consideration. This effectively reduces elementset identification to the discovery of only those elementsets that exceed a specified presence within D, or the discovery of *valid* elementsets.

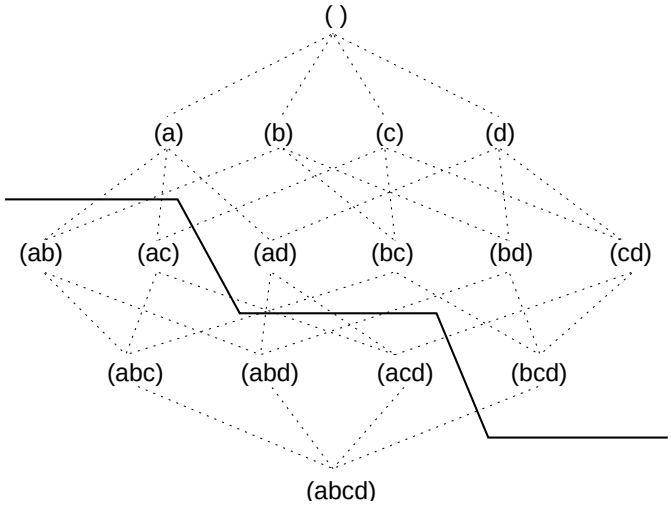


Figure 1.2: Bounded search space lattice

If a constraint is reflexive, or directed, its inclusion can provide significant optimisation due to the definition of a boundary within the search space lattice, above or below which exploration is not required (shown in Figure 1.2). In relation to elementset identification, a reflexive constraint is therefore one that never decreases (*monotonic*) or increases (*non-monotonic*) as the number of elements within an elementset increases (see section 5.1 for formal definition). Reflexive constraint inclusion can therefore result in rapid search space reduction by effectively eliminating all supersets or subsets of an invalid elementset. For example, the heuristic constraint *support* is non-monotonic as given an invalid elementset, $\sigma(e) < minsup$, then all supersets of e can be eliminated from consideration. With respect to support in particular, this reflexive effect was first introduced as the *Apriori* heuristic, due to its initial inclusion within the Apriori analysis algorithm (see Section 1.2) and is also commonly known as the Downward Closure Principle (DCP). While Chapter 5 discusses various constraint types, this review typifies constraint inclusion through the use of *support*, as it is the concept of constraint inclusion not its actual nature that is of interest here. A more formal description of these common measures of interest used in association mining analysis are presented in the following subsection.

1.1.1 Notation

Let $U = \langle O, E, D \rangle$ be the universal association mining context, where O and E are finite sets of objects and elements respectively and E is distinct. $D \subseteq O \times E$, is a binary relation, commonly referred to as the dataset, such that the existence of the couple $< o, e > | o \in O \wedge e \in E$ denotes that e is related to o. Given the sets $x, y \subseteq E$, commonly called elementsets, then let $t(x)$ be the subset of O containing x, and $k = |x|$. An inference R is of the form $x \Rightarrow y$ such that $x, y \subset E \wedge x \cap y = \emptyset \wedge x, y \neq \emptyset$.

Given that the algorithm incorporates the fundamental quality heuristics of support and confidence. Then support $\sigma(x)$ is defined as the fraction of objects O that contain elementset x with respect to D ($\sigma(x) = |t(x)|/|D|$), while confidence $\gamma(x)$, or inference strength, is the frequency of O containing x that also contains y ($\gamma(x \Rightarrow y) = \sigma(x \cup y)/\sigma(x)$). Alternatively confidence can be represented as $\gamma(x \rightarrow y) = P(y|x)$, where P is the conditional probability. Therefore the support of an inference is the fraction of O that contains either x and y, $\sigma(x \Rightarrow y) = |t(x) \cap t(y)|/|D|$.

Taking these quality heuristics into account results in constraining elementset identification to the discovery of all valid elementsets V such that $V = \bigcup_i \{x | x \subseteq E | \sigma(x) \geq \mathit{minsup}\}$, where *minsup* is the specified minimum support threshold above which an elementset is considered frequent. The inferences are then derived from partitions based upon the permutations of each V_i such that the inference $x \Rightarrow y$ satisfies the conditions $x \cup y = V_i, x \cap y = \emptyset, \emptyset \neq x \neq y, \gamma(V_i) \geq \mathit{minconf}$.

1.1.2 Dataset Organisation

Association mining data is generally obtained from databases created for other uses and 'massaged' into a suitable representation, through pre-processing techniques. The resulting dataset is expressed as either a sparse vector matrix or a set of enumerations and can be organised either in regard to the objects or elements that they contain.

Within the vector matrix, each cell represents the existence or absence of an element (binary) within a particular object, while an enumerated format is a

condensed representation containing only elements positively associated with an object. The preferred format to use is largely dependant upon dataset density as although the vector format is computationally faster, as combinational processing is reduced to logic operators, the enumerated format becomes more viable as the dataset density decreases. For example, given a dataset in which $|E|$ is large and each object only contains a few elements, the enumerated format will require significantly less space and smaller combinatoric operations during elementset identification. Additionally the enumerated format can be constrained to a binary representation consisting of ordered enumerated pairings of object and element identifiers, hence if horizontally organised an object will be represented over a number of rows.

Dataset organisation is typically horizontal as each row contains an object, while vertical organisation refers to the representation of objects as columns and therefore each row within the dataset represents an element. Since a dataset is normally processed on a row by row basis, horizontal organisation is said to maintain an *object focus*, while vertical organisation maintains an *element focus*. Both organisations can be represented using either an enumerated or vector format, where they are referred to as *Tidlists* and *Tidsets* respectively when using vertical organisation. Where the Transaction Identifier (Tid) refers to the contents of each row. Previous research has shown that the vertical organisation can lead to more efficient algorithms as the search for element based inferences is better served by an element focused layout. This was summarised by Shenoy *et al.* (2000) who describe four advantages of the vertical organisation over the horizontal organisation, presented on the following page.

Recent attention has been given to the optimisation of the vertical vector representation in the case of sparse matrices through projection, reducing storage and processing. Burdick *et al.* (2001) propose a technique whereby vector reduction is achieved by projecting onto a smaller vector, consisting of positive associations only, when an elementset's support reaches a defined rebuilding threshold. The new vector is then used to calculate the support for the elementset's supersets by applying the new projection, optimising support calculation through vector reduction. Shenoy *et al.* (2000) introduce compressed vertical bitmaps or *snakes* that reduce the vertical representation in comparison to the equivalent horizontal

- **Improved elementset validation**. Elementset validation is computationally faster in a vertical layout due to its element focus. For example, using *support* the validation of ab using vertical organisation is achieved by finding the size of the intersection of two elements a and b, Equation 1.1. Using a horizontal organisation, validation becomes the sum of those objects that contain both a and b, requiring a full scan as the elements are scattered throughout O, Equation 1.2.

$$\sigma(ab) = |t(a) \cap t(b)| \tag{1.1}$$

$$\sigma(ab) = |\{d \mid a \in o \wedge b \in o \wedge \forall o \in O\}| \tag{1.2}$$

- **Automatic dataset reduction**. Given non-monotonic constraint inclusion elements can be easily removed from the dataset, reducing its size.
- **Improved vector compression.** Vector compression is greater in larger datasets and hence better in a vertical layout as typically the number of objects exceeds the number of elements.
- **Asynchronous computation.** Given non-monotonic constraint inclusion, vertical organisation allows the computation of an elementset once its subsets have been deemed valid. Whereas using horizontal layout *all* elementsets must be computed before any supersets are considered.

representation. Additionally their proposed algorithm VIPER uses a combination of horizontal and vertical layouts during processing. Zaki & Gouda (2001) introduce the concept of *diffsets* that only keep track of the differences in the Tidlists of an elementset from its generating valid elementsets. This technique is particularly efficient when dealing with dense datasets as the difference between the Tidlist of an elementset and that of its direct superset is often significantly smaller than the Tidlist itself.

Dataset organisation also influences the type of lattice traversal strategy used to explore the search space. Given the two options of either breadth or depth first, BFT and DFT respectively, BFT traversals can use either organisation format,

however DFT is optimised through the use of a vertical organisation. BFT strategies generate all valid elementsets of size κ, V_κ before generating V_κ. Therefore only a single scan of D is required for each κ using BFT and both organisation formats are acceptable. However using DFT and a horizontal organisation a scan of D is required for each recursive call during the traversal to validate the candidate elementsets, resulting in large processing overheads to validate a small number of elementsets. Whereas in using a vertical organisation a full scan of D is not required.

1.2 Classic Algorithms

This section discusses the classic elementset identification algorithms, which are regarded as those that discover all distinct valid elementsets within a dataset. Participant algorithms can be separated into two classes, candidate generation and pattern growth, within which further division is based upon traversal and underlying data structures. The majority of classic algorithms are candidate generation, where candidate elementsets are constructed and then validated. Pattern growth techniques however, eliminate the need for candidate generation by constructing complex hyper-structures that contain representations of the elementsets within the dataset.

1.2.1 Candidate Generation Algorithms

Candidate generation algorithms identify candidate elementsets before validating them with respect to incorporated constraints, where the generation of candidates is based upon previously identified valid itemsets. The core algorithm of this genre is Apriori upon which many later algorithms are based.

This section provides a comprehensive survey of candidate generation algorithms providing a detailed description of the core algorithms and highlighting the significant contributions of others. Firstly, there is a short discussion on the common data structures used within candidate generation algorithms. The following survey is then split into merge and extension based algorithms that

roughly correlate to BFT and DFT. Finally there is a section relating to hybrid algorithms that attempt to provide further optimisations by switching between traversal and accrual methods during processing.

Tree Data structures

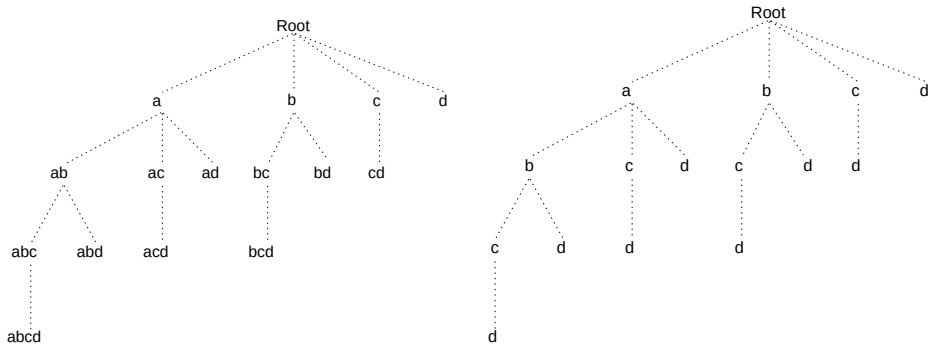
Candidate generation algorithms generally use tree based data structures of which there are three common types: 1) hash trees, 2) enumeration-set trees and 3) prefix trees.

Hash-trees, a combination of b-tree and hashtable structures, was introduced by Agrawal & Srikant (1994) as an effective candidate elementset storage structure. The structure is effectively a b-tree for which every internal node is a hashtable and every leaf node or bucket contains a set of elementsets. When a bucket reaches its quota of elementsets, the hash-tree extends by replacing the bucket with a new hashtable into whose buckets the elementsets are placed.

The enumeration-set tree (Rymon 1992), is an ordered tree where each node n represents an elementset, $e \in V$, and an edge represents a single element extension of that elementset. Therefore each tree level contains elementsets of the same length. For example third level nodes contain valid elementsets of length three (V_3).

An enumeration-set tree evolves through the single element extension of leaf nodes, each resulting in a new child node. This process is ordered and typically constrained so that, given a particular element ordering, only those elements occurring after the last element of the current element set are considered during extension of the current elementset. Given this subset of possible extensions the process is typically further constrained so that only new valid elementsets are inserted into the tree. The presence of reflexive constraints, in this case non-monotonic constraints, further reduces the set of possible element extensions to those elements that resulted in valid extensions of its direct ancestor (parent node).

For example, given a node n and an ordered element set E, then n's extension set, n^{ext} consists of all $e \in E$ that occur after e_i where $n = \{e_1, e_2 \dots e_i\}$.



(a) Enumeration Set (b) Prefix Tree (trie)

Figure 1.3: Common storage structures

This process is illustrated in Figure 1.3(a) for the element set $E = \{a, b, c, d\}$. Furthermore if the process is non-monotonically constrained then n^{ext} is further constrained to those valid extensions of its parent p.

Prefix trees, also known as tries, (Figure 1.3(b)) although fundamentally equivalent, differ from the enumeration-set tree in that each node specifies the projected prefix element for a particular sub-tree, instead of the entire prefix. This is evident within Figure 1.3 where the two trees are structurally equivalent, the difference being that while within the enumeration-set tree all elementset information is contained at the node, within the prefix tree the same information is accrued during traversal to the node (elementset) in question.

Merge Algorithms

The fundamental merge algorithm is Apriori (Agrawal & Srikant 1994), commonly regarded as the classic association mining algorithm, which introduced reflexive constraint inclusion, namely *support*, to reduce exploration. The algorithm derives candidate elementsets C_κ from $V_{\kappa-1} \mid \kappa > 1$ incorporating *support* to reduce $|C_\kappa|$. From this V_κ is derived through a scan of the dataset, accruing counts for each $c \in C_\kappa$. Therefore given a *support* threshold $minsup$, $V_\kappa = \{C_\kappa : \sigma(C_\kappa) \geq minsup\}$.

Algorithm 1.1 presents the iterative Apriori algorithm that requires a dataset for each κ. The algorithm has two main parts: 1)candidate generation and 2) validation. The set of candidates is formed by the set of elements, E, given $\kappa = 1$,

Algorithm 1.1 Apriori Elementset Generation

Apriori Elementset Generation

1: $\kappa = 0$
2: **repeat**
3: κ^{++}
4: **if** $\kappa > 1$ **then**
5: $\{C_\kappa = \text{generate\_candidates}(V_{\kappa-1})\}$
6: **else**
7: $\{C_1 = E\}$
8: **end if**
9: **for all** O **do**
10: $C_o = \{c \mid c \subseteq o \ \wedge \ c \in C_\kappa \ \wedge \ o \in O\}$
11: **for all** C_o **do**
12: $C_\kappa^i.count^{++} \mid C_o \in C_\kappa$
13: **end for**
14: **end for**
15: $L_\kappa = \{C_\kappa \mid C_\kappa^i.count \geq minsup\}$
16: **until** $V_\kappa = \emptyset$
17: return $L = \bigcup_{k=o}^{n} L_\kappa$

otherwise it is based upon a merge function involving members of $V_{\kappa-1}$. Subsequent accrual determines the candidate elementset *support* and those meeting the nominated threshold *minsup* are appended to the valid set V_κ. Therefore $V_1 = \{e \mid e \in E \wedge \sigma(e) \geq minsup\}$.

Subsequent $L_\kappa \mid \kappa > 1$ is based upon $V_{\kappa-1}$, given the inclusion of *support*, so that an elementset can only be valid if all subsets are valid. For example given a valid elementset $a \mid a \in V$ then all subsets of a also exist in V. Given this, C_κ is constructed from the constrained merge of $V_{\kappa-1}$ pairs, where the pairs only differ by a single element and all other $\kappa - 1$ permutations of the new elementset also exist in $V_{\kappa-1}$.

For example, given two elementsets $a = \{a_1, a_2 ... a_n\}$ and $b = \{b_1, b_2 ... b_n\}$ a merge only occurs if a and b's last elements differ (Equation 1.3). The newly created candidate elementset $c \mid c = \{a \cup b\}, |c| = k$ is then appended to the set of candidate items C_κ if all subsets of length $k - 1$ exist in $V_{\kappa-1}$. Given that

$V_3 = \{abc, bdg, abf\}$ then since abc and abf differ by only their last element a new elementset $abcf$ is constructed. However since not all V_3 subsets of $abcf$ in $V_{\kappa-1}$ exist (e.g. bcf), $abcf$ is not appended to C_κ as it cannot be valid due to the reflexive *support* constraint.

Once the set of candidates, C_κ, has been constructed, accrual is undertaken to find the valid candidates to be appended to V_κ, using the same accrual technique as for V_1. This process repeats, with elementset length incrementing, until the newly generated V_κ is empty.

$$c = \{a \cup b | a, b \in V_{\kappa-1} \wedge \{a_1 ... a_{n-1}\} = \{b_1 ... b_{n-1}\} \wedge a_n \neq b_n\} \tag{1.3}$$

Independent association mining research occurring at the same time (Mannila, Toivonen & Verkamo 1994), also came to the same conclusion as Agrawal & Srikant in regard to the reflexive constraint inclusion. However this implementation, Offline Candidate Determination (OCD), is less efficient as the join conditions are not as tightly constrained, resulting in superfluous sets. Two join operations were devised, the first variant (Equation 1.4) specifies that any but only two members could differ while the second variant (Equation 1.5) extended each member of $V_{\kappa-1}$ with every member in V_1.

$$c = \{a \cup b | a, b \in V_{\kappa-1} \wedge |a \cap b| = k - 2\} \tag{1.4}$$

$$c = \{a \cup b | a \in V_{\kappa-1} \wedge b \in V_1 \wedge b \not\subseteq a\} \tag{1.5}$$

The realisation of reflexive constraints enabled Apriori to surpass earlier frequent elementset algorithms, namely AIS and SETM. AIS (Agrawal, Imielinski & Swami 1993) creates V_κ from $V_{\kappa-1}$ by scanning D and for each elementset $a \mid a \subset o \wedge o \in O \wedge a \in V_{\kappa-1}$, a new group of candidate elementsets are considered by extending a with each element in o that lexicographically occurs after the last element of a. If the new elementset already exists then its count is incremented else it is appended to C_κ with a count of 1. After parsing O, $V_\kappa \subset C_\kappa \mid C_\kappa^i.count \geq minsup$.

SETM (Houtsma & Swami 1993), also generates candidate elementsets on the fly, however the need to iterate through an object's $V_{\kappa-1}$ permutations to

find elementsets is alleviated by separating the algorithm's candidate generation and counting functions. During candidate generation the algorithm stores each candidate elementset and its object identifier in an array based structure A_κ. Once the dataset scan is complete, candidate validity is determined by sorting and aggregating A_κ, removing any A^i_κ, that do not meet the criteria. The subsequent generation of $A_{\kappa+1}$ is facilitated by A_κ as all valid κ permutations of $o \in O$ are readily available.

The advent of Apriori and the reflexive constraint inclusion provided the standard pruning or search space reduction technique, mainly through the use of the quality heuristic *support*. Subsequent research in analysis optimisation focused upon reducing I/O through condensed representations, dataset partitioning, dataset pruning and dataset access reduction.

AprioriTID (Agrawal & Srikant 1994) extends Apriori by eliminating multiple scans of D through the construction of a counting-base set $\bar{C}_\kappa$, during construction of V_1. The counting base is of the form $< Tid, C^i_\kappa >$ where Tid is the object identifier and C^i_κ denotes the set of C_κ present in an object $o \in O$. For example, if $o = \{a, b, d, f\}$ then equivalently $\bar{C}^i_1 = \{\{a\}, \{b\}, \{d\}, \{f\}\}$, however $\bar{C}^i_2$ contains all potential V_2 within the object o hence $\bar{C}^i_2 = \{\{ab\}, \{ad\}, \{bd\}, \{bf\}, \{df\}\}$. It is apparent from this example that $\bar{C}_\kappa$ may potentially be larger than D for small $\kappa > 1$ however this is quickly reduced as κ increases as if o does not contain a C_κ then $\bar{C}_\kappa$ will not have an entry for that object, in effect incorporating a form of dataset pruning.

Another Apriori extension proposed by Agrawal & Srikant (1994) is delayed accrual (or pass bundling (Mueller 1995)), which is based upon the observation that any $\sigma(C_i) \mid |C_i| = \{\kappa+1, .., 2\kappa\}$ can be represented as the union of some pair of V_κ elementsets. Therefore from a single scan of D the *support* of all candidates of length $\kappa+1...2\kappa$ can be computed. A trade-off exists however between the time saved by reducing the dataset access and the number of false positives generated through the projection of C_κ instead of V_κ.

Dynamic Itemset Counting (DIC) (Brin, Motwani, Ullman & Tsur 1997) reduce I/O by enabling the calculation of V_κ elementsets during earlier scans, $\leq \kappa$, through dataset partitioning and the use of checkpoints. If during processing all immediate $\kappa - 1$ subsets of a larger elementset $a \in C_\kappa^i$ have been determined valid at a checkpoint (end of a dataset partition) then the calculation of $\sigma(a)$ can begin and will terminate when it reaches the same checkpoint during the next dataset iteration. This reduces dataset access, as within a single scan elementsets of differing lengths can be counted, where the length of the active elementsets is always $\geq$ the current scan number.

Dataset-global pruning is based upon the premise that if an element participates in V_κ^i then it must participate in $|\kappa - 1|$ $V_{\kappa-1}$ elementsets. Therefore any e that does not appear in at least $|\kappa - 1|$ $V_{\kappa-1}$ elementsets can be removed from o. If this process subsequently results in $|o| < |\kappa|$ then o is removed from D.

Dataset-local pruning, also used in DHP, is applied to each object during subset counting. The premise of which states that an element e can only participate in an elementset $V_{\kappa+1}^i$ if e exists in $|\kappa|V_\kappa$ subsets within o. However since V_κ is unknown and since $C_\kappa \supseteq V_\kappa$ then C_κ is used instead. Hence given $c \mid c \in C_\kappa$ then any element e such that $|e \in c| < \kappa$ is removed from the object.

Direct Hashing and Pruning (DHP) (Park, Chen & Yu 1997), introduces the concept of *possible frequent itemsets* C'_κ to optimise the generation of candidate elementsets and to also provide a novel dataset trimming technique. The possible frequent elementset $C'_{\kappa+1}$ accrues the count of each possible $C_{\kappa+1}$ elementset within each $o \in O$ for which all κ subsets exist in C'_κ during the construction of V_κ. This in effect accumulates information about C_{k+1} in advance so that all possible C_{k+1} elementsets are encoded within the C'_{k+1} hashtable and results in the production of a significantly smaller but accurate candidate set C_κ. However, because of the possible collisions on C'_{k+1} entries, each derived C_κ^i still needs to be checked for *minsup*. The algorithm also incorporates progressive dataset prun-

ing to discard elements and objects that are of no further use. The two pruning techniques - dataset-global and dataset-local (presented above) - are incorporated during accrual to reduce $|D|$ during each scan .

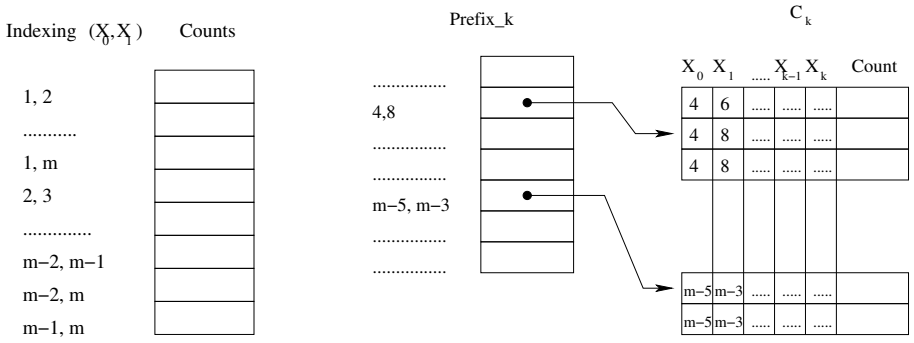
DHP also incorporates the concept of delayed accrual, discussed earlier. The incorporation of progressive trimming, delayed accrual and the identification of C'_κ results in significant speedup over Apriori for small κ due in main to the reduced size of C_κ.

Perfect Hashing and Pruning (PHP) (Ozel & Guvenir 2001) proposes an optimisation to DHP by using perfect hashing in the creation of the hash table for $C'_{\kappa+1}$. This effectively eliminates the hash table collisions that were apparent in DHP and hence $C'_{\kappa+1}$ will contain the actual counts of the $C_{\kappa+1}$ elementsets alleviating the need to re-count the occurrence of $C_{\kappa+1}$ elementsets in D.

Direct Count and Prune (DCP) (Orlando, Palmerini & Perego 2001*b*) attempts to optimise elementset discovery by incorporating the dataset pruning techniques introduced in DHP and through using direct counting in the validation of candidates. Instead of the usual hash-tree structure to perform accrual, DCP uses a *direct counting* method, as for small κ the hash-tree structure is ineffective as C_κ is generally large and because tree depth is based on κ only a few partitions will be created. The direct count method is a generalisation of Apriori's accrual technique, differing for $\kappa \geq 2$.

V_2 generation, based upon V_1, uses a vector *Counts* to accrue each C_2^i candidate pair, in lexicographical order, using a perfect hash function, shown in Figure 1.4(a). The counting of subsequent C_κ is accomplished by extending *Counts* to create a directly accessible prefix tree, of shared 2-prefix. This allows direct access to the subset of candidates specified by their 2-prefix. This structure exploits spatial locality in that once a prefix $\{a_{i_1}, a_{i_2}\} \mid a_{i_1} < a_{i_2}$ is selected the possible κ completions to the prefix existing in o will be found in the subsequent contiguous section of C_κ.

The Partition algorithm (Savasere, Omiecinski & Navathe 1995) discovers all valid elementsets in two dataset scans. This is accomplished by partitioning the dataset, each of which fits into memory, and then finding the valid elementsets that exist in each partition p. The theory being that any globally valid elementset



(a) Counts (κ=2) (b) Prefix ($\kappa > 2$)

Figure 1.4: DCP: Data Structures

V_e must be valid within at least one partition V^p. During the first dataset scan all valid elementsets for each partition V^p are generated using AprioriTid, the vertical organisation of which is constructed during the generation of V_1^p. This is achievable as each partition can fit in memory and hence D is only scanned once during the generation of V_p. Based upon this the global set of candidates C is derived from the union of V_κ^p (Equation 1.6). The second dataset parse determines V from C through accrual of each C_i within each partition.

$$C = \bigcup_{\kappa=1}^{n} c_\kappa \;\mid\; C_\kappa = \bigcup_{p=1}^{n} V_\kappa^p \tag{1.6}$$

SPINC and AS-CPA algorithms provide optimizations to the Partition algorithm. SPINC (Mueller 1995) reduces process time by incrementally constructing C, adding V_κ^p to C whenever it is available, instead of waiting until the end of the first scan as in Partition. This allows SPINC to start global occurrence accrual for each C_i during the first scan, potentially resulting in scan reduction. AS-CPA (Lin & Dunham 1998) attempts to eliminate any data skew apparent in Partition's results. Data skew refers to the irregularity of data distribution over D that may cause the generation of false candidates. For example, seasonal trends, in which a product's sales are high for a short period of time only, may only meet a support threshold for a particular temporal partition but not globally. AS-CPA achieves skew reduction by introducing early local pruning and through two global anti-skew techniques. Early local pruning is based upon *better* local valid elementsets B_κ^p, where B_1^p is derived from $V_1^p \cap V_1^{2...p}$ and used to construct

the local set of valid elementsets (B_κ^p) using regular Apriori. Since $B_1^p \subseteq V_1^p$ generally $|B_\kappa^p| < |V_\kappa^p|$ results in reduced processing. The two global anti-skew techniques refer to the first and second dataset scans and assume the incorporation of SPINC's incremental C construction within the algorithm. They are based upon the premise that if an elementset is not valid over a number of partitions then it should be removed from consideration. If, during further processing, the same elementset is found to be valid within a partition it can again be considered potentially valid (see (Mueller 1995) for a full discourse).

Hierarchical Bit Maps (HBM) (Gardarin, Pucheral & Wu 1998) uses a vertically organised vector format, calculating support efficiently by performing 2-way intersections of *relevant* bitmaps. The algorithm encodes the bitmap into an unsigned *short* by creating groups of 16 bits upon which the subsequent intersection is performed. It was realised however, that in a typical mining run many *short*s were empty due to sparsity, hence performing the intersection calculation was inefficient. To alleviate this, a second level of bitmap indexing was created, in which each bit represents whether or not a *short* is empty. Hence when calculating the support of two elementsets, an intersection of the second level indexing is done first, providing a list of the non-zero *short*s to consider further. Prior to HBM, Gardarin *et al.* developed NBM which used the same format however did not make use of the second level of indexing.

ColumnWise (Dunkel & Soparkar 1999) uses a column based, instead of row based, iteration of D, hence although d remains horizontally organised it uses intersections in the construction of C_κ. The actual processing is similar to Apriori using a horizontal bit vector layout and intersection accrual. If the results of prior $(\kappa - n : n > 1)$ joining operations are preserved, $\sigma(C_\kappa^i)$ is the intersection of 2 immediate subsets of C_κ^i, otherwise a κ-way join of each participant element is required.

DLG (Yen & Chen 1996) is a novel graph-based technique based upon a lexicographically ordered vertical bit vector layout. Through bit vector summation all infrequent elements are removed resulting in V_1, from which the association graph is constructed. The algorithm constructs an edge between any two elements a and b such that $a < b, |a \cap b| \geq minsup$, effectively resulting in the identification

of V_2. Subsequent V_κ generation where $\kappa > 2$ is based upon direct extension, where the edges from the last node of V_κ^i, are used to create $C_{\kappa+1}$. If no edges exist, then no $\kappa + 1$ elementsets based upon it can exist. For example, given a valid V_κ elementset $a = \{a_1, a_2..a_n\}$, if no additional directed edges from a_n exist, apart from a_{n-1}, then no $V_{\kappa+1}$ that are supersets of a can exist. Therefore for all directed edges from a_n, a $C_{\kappa+1}$ is generated, the actual support for which is calculated through the κ-way intersection of participants.

Extension Algorithms

Extension algorithms rely upon the single element extension of valid elementsets to derive new candidate elementsets. This section discusses the contributions of various extension based algorithms, organised in relation to contribution, providing a detailed discussion of core algorithms and summarising the contribution of other algorithms that extend these core algorithms.

Tree Projection (Agrawal, Aggrawal & Prasad 1999), discovers valid elementsets through the construction of an enumeration-set tree and by incorporating object projection and matrix based accrual techniques. The enumeration-set tree is built in the typical DFT manner described in Section 1.2.1. Matrix based accrual is facilitated through object projection, where only objects relevant to a node's elementset are projected to the node.

Tree Projection is initiated by the regular BFT generation of V_1, which forms the basis of the enumeration-set tree and the initial extension set of the root node $\Re^{ext}$. From this the frequency matrix $\Re^{freq}$ is constructed and populated by $\Re^{project}$, which is a projection of D containing only those elements in $\Re^{ext}$. Object projection is progressive so that n^{freq} is based upon $n^{project}$ which has been derived from $\bar{n}^{project}$ and filtered to only those elements in n^{ext}. The technique of projecting the relevant object information at each node significantly reduces processing through object reduction.

Subsequently the calculation of a node's extension set n^{ext} is derived from its parent's frequency matrix $\bar{n}^{freq}$, within which, each cell contains the co-occurrence frequency of the intersecting elements within $\bar{n}^{project}$. Given $minsup = 2$, Figure 1.5 presents a set of related matrices for $\{r, a, ab\}$. From which it can be seen that

	a	b	c	d	e
a					
b	2				
c	1	3			
d	3	4	4		
e	2	1	2	2	

(a) $\Re^{freq}$

	b	d	e
b			
d	3		
e	2	3	

(b) a^{freq}

	d	e
d		
e	1	

(c) ab^{freq}

Figure 1.5: Tree Projection: Matrices

the extension set of a ($a^{ext} = \{b, d, e\}$) is derived from rows in column a of $\Re^{freq}$ that are greater than or equal to $minsup$ and which lexicographically occur after element a. From which the projected dataset $a^{project}$ is derived and subsequently a^{freq} constructed. This recursive process continues until $x^{ext} = \emptyset$. The resulting constructed enumeration-set tree uniquely identifies all valid elementsets, where validity (based upon frequency) is derived from projected matrices.

The authors additionally propose different tree construction techniques based upon BFT and DFT. Using BFT all n_{κ}^{ext} are found before $n_{\kappa+1}^{ext}$ by re-initiating the object projection process for each subsequent level of the tree. Each object $\Re^{project}$ is in turn recursively projected to all $n_{\kappa-1}$ matrices. Using DFT the projected datasets are maintained for all nodes and their siblings on the path from the root node to the node currently being extended. Hence matrix creation becomes an independent problem with a significantly reduced dataset. The choice of strategy relates to a trade-off between I/O and processing, BFT requires more processing as each object must be re-projected for each successive tree level, while DFT requires that all projected datasets be maintained. While BFT is more efficient in the discovery of short elementsets, DFT is better for finding long elementsets. A hybrid technique is also suggested that uses BFT until each projected dataset for a tree level can be held in memory. At which point each projected dataset is saved to a separate file, sequentially read into memory and independently processed using DFT, discovering all descendants.

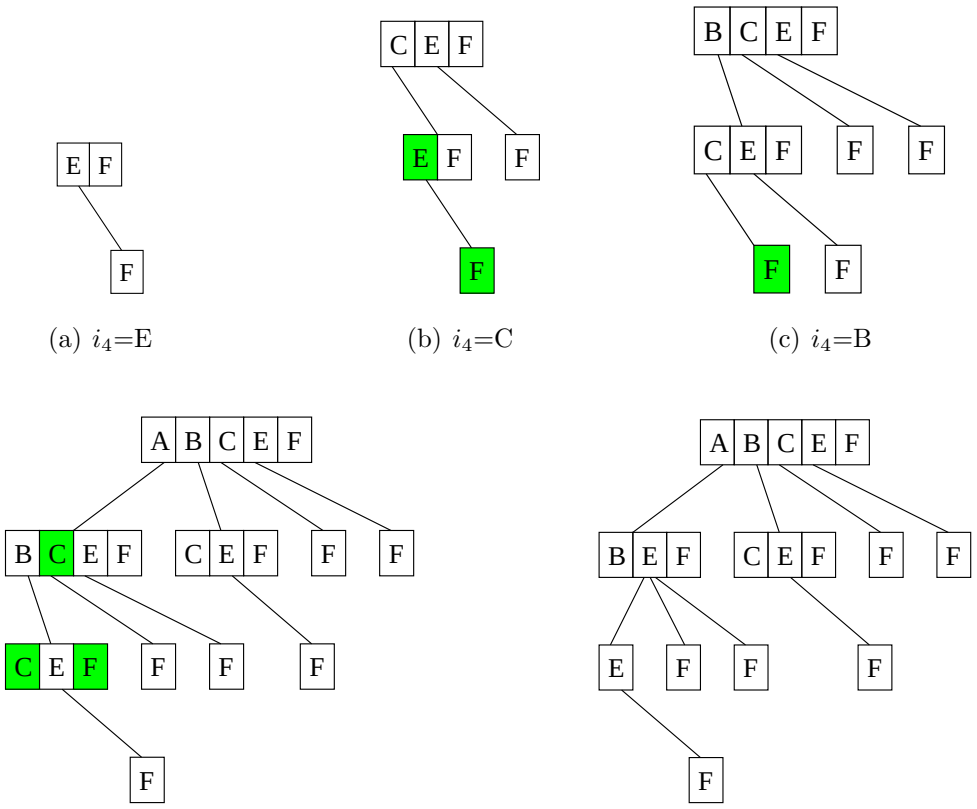
In the pursuit of optimising long elementset discovery, the same authors developed Depth Project (Agrawal, Aggrawal & Prasad 2000). The algorithm has similar foundations in its reliance upon the enumeration-set tree and a DFT strategy, however it differs in regard to dataset projection occurrence and ac-

crual. The algorithm uses a bit-vector structure in which each bit represents a projected object to provide efficient accrual, if a node's value or elementset is a subset of an object the relevant bit is turned on. A node's bit-vector is derived from its parent's bit vector by turning off all objects that do not contain the extension element. This structure is ineffective where $|E|$ is *much* larger than the average number of elements within an object as most bit's will be off. However it becomes efficient due to the re-projection of the objects when a minimum *on* threshold is reached that results in the derivation of a new, reduced vector with all element bits on.

Furthermore Depth Project proposes the generation of the extension set n^{ext} that is similarly derived from $\bar{n}^{ext}$, without using projected matrices. The projected dataset $n^{project}$ is first filtered so only valid $\bar{n}^{ext}$'s objects remain. If there exists a large number of distinct projected objects at n, accrual is through byte counting of $a \mid a \in \bar{n}^{project}$. If the distinct projected set is small, typically at lower tree levels, the repetitiveness is exploited through a novel bucketing technique that counts the entire sub-tree in a single projected dataset scan.

Apriori-*DF* (Pijls & Bioch 1999) propose a novel trie extension technique in which the valid subtree of V_1^i used to calculate the valid subtree of V_1^{i-1}. The algorithm constructs V_1 in regular Apriori fashion and stores them in order of ascending frequency. Subsequent processing is undertaken from most to least frequent (left to right), with a dataset scan required for each V_1 member, therefore a total of $V_1 - 1$ scans of the dataset are required. This process, given the ordered set V_1={A,B,C,E,F} , is highlighted in Figure 1.6(from (Kosters & Pijls 2003)), showing the replication of V_1^n's subtree under V_1^{n-1}, which provides the candidate set based upon V_1^{n-1}. Any invalid nodes are then removed from this subtree, by performing dataset accrual. A subsequent implementation described in (Kosters & Pijls 2003) claims to further optimise the algorithm through memory management by undertaking accrual in the subtree before appending it to V_1^{n-1}, thereby alleviating the need to delete invalid elementsets.

Eclat and Clique (Zaki 2000*b*) are similar techniques that improve valid elementset discovery by recursively decomposing an association lattice into disjoint subsets, until each can be independently solved in memory, reducing I/O. Both



(a) i_4=E (b) i_4=C (c) i_4=B

(d) i_4=A (e) complete trie

Figure 1.6: Apriori-df trie construction with invalid candidates highlighted

algorithms use the association lattice, BFT traversal and the same accrual technique, however they differ in regard to the method of decomposition.

The association lattice used by these algorithms is derived from V_2, which is constructed using a classical Apriori approach. Within the association lattice, each element is represented by a node and edges between nodes represent the presence of an association within V_2 (Figure 1.8(a)). Typical organisation is vertical and hence accrual is based upon Tidlist intersections for each sub-lattice constructed. The authors also introduce a variation (AprClique) that uses a horizontal dataset organisation, and hence uses counting instead of accrual. Furthermore, Eclat's accrual is undertaken in reverse lexicographical order to optimise pruning.

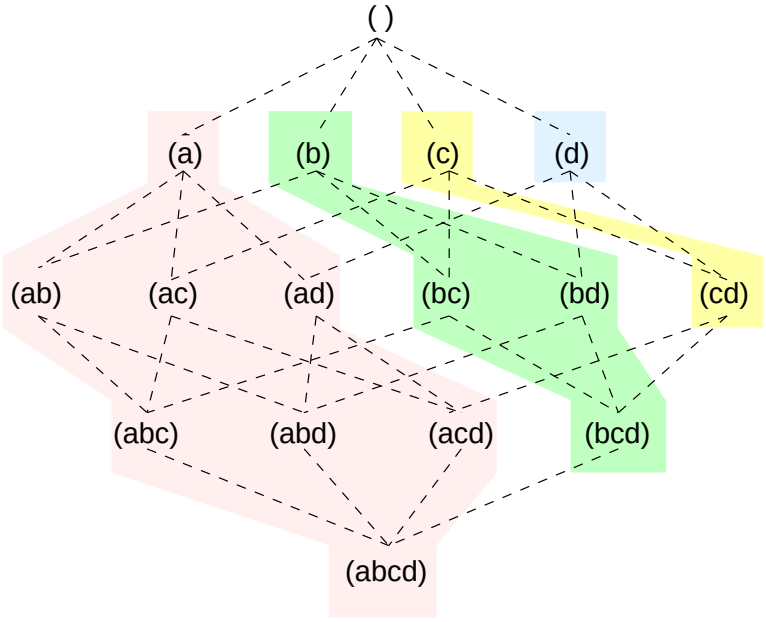


Figure 1.7: Eclat: Equivalence classes

Eclat decomposes the search space lattice using the concept of prefix-based equivalence relations, where members of a relation share the same κ-prefix. For example, given $V_2 = \{ab, ad, bc, bd, cd\}$ the resulting equivalence classes are $[a] = \{b, d\}, [b] = \{c, d\}, [c] = \{d\}$. Furthermore each equivalence class is a lattice in its own right and can undergo recursive decomposition until each sub-lattice fits into memory. Figure 1.7 illustrates the search space lattice presented in Figure 1.1 segregated into equivalence classes.

Once decomposed, all valid elementsets are then discovered in each class using a BFT Tidlist intersection strategy incorporating *support*. In addition, it is apparent from Figure 1.7 that dependencies exist between the classes, i.e. *acd* is dependant upon *cd* as well as subsets from within its own class. By solving the classes in the reverse order to which they were created all subset information required for effective pruning is available.

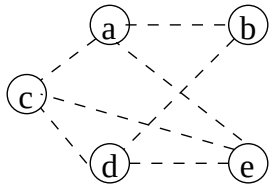
Clique decomposes the search space using the concept of maximal cliques. A clique is a complete lattice, where completeness is denoted by an edge between all pairs of vertices. A maximal clique set is then the set of cliques that represent the search space lattice within which no clique is a superset of any other. This

technique uses additional information to form smaller sub-lattices than Eclat's prefix-based approach and hence is often more effective as less accrual is required.

Maximal clique enumeration is based upon finding the covering elements of a class, each of which is said to cover a subset of the class. Given a class $[x]$, where $y \in [x]$, then y is said to cover the subset of $[x]$ given by $cov(y) = [x] \cap [y]$. From the covering set of a class (given in Equation 1.7) the maximal cliques for that class are derived by recursively discovering the maximal cliques of each covering set element. The maximal clique set of $[x]$ is then each covering set element's maximal clique set intersected with $[x]$ and prefixed with $[x]$'s class identifier.

$$[x]_{cov} = \{y : y \in [x] \wedge cov(y) \neq \emptyset \wedge cov(y) \not\subseteq cov(z) \forall z : z \in [x] \wedge z < y\} \qquad (1.7)$$

This process is illustrated in Figure 1.8, in which Figure 1.8(a) and Figure 1.8(c) present the association lattice derived from $V_2 = \{ab, ac, ae, bd, cd, ce, de\}$ and the associated prefix classes. The cliques, presented in Figure 1.8(c), are then found by recursively calculating cover sets for each prefix class, from which the maximal clique sets in Figure 1.8(d) are derived.



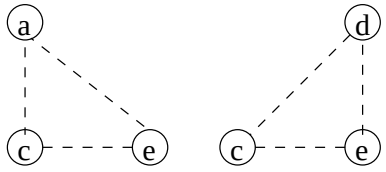
(a) V_2 Association Lattice

[a] = b,c,e
[b] = d
[c] = d,e
[d] = e

(b) Prefix classes

[a] = ace
[b] = ϕ
[c] = cde
[d] = de

(c) Class cliques



(d) Derived Maximal Cliques

Figure 1.8: Maximal Clique Derivation from: $V_2 = \{ab, ac, ae, bd, cd, ce, de\}$

The decision between using Eclat or Clique is based upon the nature of the dataset as the enumeration of maximal cliques can be computationally expensive for dense datasets. As density increases prefix-based decomposition, Eclat becomes more competitive. An extension to Eclat (dEclat) was subsequently proposed by Zaki & Gouda (2001) that used the concept of diffsets (see Section 1.1.2) to reduce memory requirements by only storing the differences in the tidList of a candidate elementset from its generating elementset.

Partial Support Tree (Goulbourne, Coenen & Leng 2000) requires only a single pass of a horizontally organised dataset, during which an enumeration-set tree is dynamically constructed and *support* counts are partially calculated. *Support* calculation is then completed using DFT. This algorithm is effective for tasks in which $|E|$ is small or where many duplicate objects exist. The tree is constructed during the dataset scan by creating new nodes, in the appropriate location, for each unique object. Dummy nodes are also inserted as required to maintain an enumeration-set tree structure. During this process interim node counts are incremented, resulting in a partial support calculation of each node $\bar{\sigma}(n)$ (Equation 1.8), where d_o is the *support* in D of the unique object o and the tree's ordering is lexicographic. Based upon $\bar{\sigma}(n)$ it becomes possible to compute *support* $\sigma(n)$ by appending to $\bar{\sigma}(n)$ the count of n's participation within nodes lexicographically proceeding n in the enumeration-set tree by using DFT (Equation 1.9).

$$\bar{\sigma}(n) = \sum d_o\{\forall o : o \subseteq n, o\ follows\ n\} \tag{1.8}$$

$$\sigma(n) = \bar{\sigma}(n) + \sum d_o\{\forall o : o \subset n, o\ precedes\ \ n\} \tag{1.9}$$

Hybrid Algorithms

Hybrid algorithms, those that combine both counting and intersection accrual strategies, are based upon the premise that although counting is more effective than intersection calculation for small κ, as κ increases intersection techniques become more viable. This occurs as the cost of finding and accruing small κ elementsets is less, in a hash-tree, than the cost of calculating the intersection of

large elementsets (i.e. if κ is small then generally $|a.tidList|$ is large). However as κ increases, hash-tree location becomes more expensive and intersection becomes faster.

Based upon this premise (Agrawal & Srikant 1994) developed the first hybrid approach, AprioriHybrid, which switches from counting to intersection accrual when $|C_\kappa| << |O|$, using Apriori for counting accrual and AprioriTid for intersection accrual. This algorithm was subsequently optimised by Hipp, Guntzer & Nakhaeizadeh (2000) and by Bodon (2003). Hipp *et al.* eliminating the need to build a new structure upon organisation change. This was achieved by using a hash-tree like structure containing pointers to tidList sets instead of counters, that could then be used for both accrual techniques. While Bodon developed Apriori-Brave, in which both delayed accrual and organisation switching are based upon memory usage, furthermore the algorithm implements dataset pruning removing invalid elements from the dataset.

Dynamic Counting & Intersect (DCI) (Orlando, Palmerini & Perego 2001*a*) is an extension of DCP that switches to intersection accrual, from DCP's direct counting method, when the dataset is able to fit into memory. The technique then transforms the dataset into vertical organisation and uses a κ-way intersection of the elementset's participants to calculate elementset *support*. This was adopted to reduce memory usage, as it requires the maintenance of V_1 only, instead of all $V_{\kappa-1}$ elementsets, which can require orders of magnitude more memory. From a processing perspective the κ-way intersection technique is more expensive than 2-way intersection technique. This led to the development of the set of computation optimisations, presented on the next page, that have shown to be of significant benefit when many complex patterns are being discovered due to inherent overheads. Therefore these optimisations are introduced only if $|C_\kappa| \gg |\bar{E}| \mid \bar{E} = E \in C_\kappa$.

kDCI (Lucchese, Orlando, Palmerini, Perego & Silvestri 2003) subsequently extends DCI by using density heuristics and efficient data structures in order to adapt algorithm behaviour based upon dataset characteristics. Furthermore a novel counting inference strategy is proposed based upon the concept of *key patterns*, discussed in Section 1.3.1, through which an elementset's support can be

- Since C_κ is lexicographically ordered the number of intersections required is reduced by caching intermediate intersections.
- As bit vectors are relatively sparse, speed up is achieved by skipping the runs of 0's, so that only vector segments containing 1's are actually intersected (similar to HBM).
- Dataset object pruning (column removal) is conducted when none of its elements participate in V_κ, this is achieved by maintaining a bit vector initialised to 0's that is *OR*'ed with every discovered valid elementset. At the end of the scan any element's containing a 0 are deleted.
- When the vertical layout is created it is organised in increasing order of *support* as the higher the frequency the greater the likelihood of the element participating in V_κ where κ is large. Therefore as the section of the dataset accessed shrinks, spatial locality is enhanced.
- During pruning the columns are re-ordered to increase the length of runs of 0's and 1's to facilitate pruning based upon runs of 0's.

inferred, alleviating accrual of some elementsets. The effectiveness of this strategy is based upon the ratio of valid elementsets to *key patterns* a fact only known upon computation completion. However the authors discovered that this effectiveness metric can be derived from the average element support (after the first dataset scan), providing early justification as to the benefit of applying counting inference.

VIPER (Shenoy *et al.* 2000) uses accrual to calculate V_1 and V_2, then changes to an intersection based technique, storing elementsets in a compressed bit-vector format called *Snakes* to minimise storage costs. Candidate generation is optimised by incorporating equivalence-class concepts, grouping together all V_κ with a common $\kappa - 1$ prefix, from which $C_{\kappa+1}$ will be derived. By taking advantage of the spatial locality of elementsets this technique allows simultaneous subset search, reducing processing time. The accrual process incorporates delayed accrual by creating a tree structure with leaf nodes V_κ and internal nodes containing supersets of these leaf elementsets. This structure is then used to propagate accrual,

bottom-up, starting from the leaf elements to all relevant supersets in a single dataset scan. Once complete, nodes with a count exceeding *minsup* are appended to the relevant V. Furthermore the algorithm introduces techniques to minimise the number and size of *Snakes* required and also deletes all *Snakes* created in previous scans that are no longer required for future computation.

1.2.2 Pattern Growth Algorithms

In contrast to the more prolific candidate generation techniques, pattern growth algorithms eliminate the need for candidate generation through the creation of complex hyper-structures. Hyper-structure is a term used to describe the general data storage structure used in pattern growth algorithms. The structure is comprised of two linked structures a *pattern-frame* and an *element-list*, which together provide a concise representation of the relevant information contained within D. The first stage of analysis populates the hyper-structure and, so long as the representation can be maintained in memory, further dataset access is not required. Subsequent mining involves DFT analysis of the pattern-frame, accessed through the element-list. Figure 1.9 presents a sample tree form of hyper-structure, from the FP-growth algorithm, illustrating the linkage between the element list and pattern frame. However the nature of the hyper-structure is algorithm dependant, varying in relation to the sub-structures and the underlying semantics.

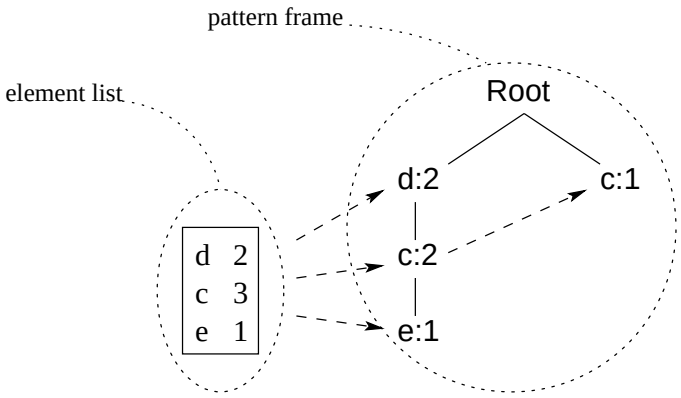


Figure 1.9: Example Hyper-structure

The fundamental pattern growth algorithm FP-Growth was proposed by Han and Pei (2000) and uses a tree based pattern-frame (*FP-Tree*) and an associated link structure (*FP-Link*) within the analysis process. FP-Tree construction begins with the classic BFT calculation of V_1, the contents of which are sorted in order of descending *support*. A second dataset scan then reduces and reorders each object to comply with the ordering of V_1 and is inserted into the hyper-structure (*FP-structure*) the result of which is presented in Figure 1.10.

Given a reduced and ordered object $\bar{o} = \{e_1, e_2, ..., e_n\}$ its insertion into FP-structure is recursive requiring a search and update process within FP-Tree and then an update of FP-Link for each element in $\bar{o}$. FP-Tree insertion starts with $\Re$ being the active node and proceeds by checking the object's first element (e_1) against the existing first level FP-nodes. If e_1 does not exist, a new first level FP-Tree node is created for e_1, with a count of 1, else the existing nodes count is incremented. The element e_1, is then inserted into FP-Link in a similar manner, creating if it does not exist and incrementing if it does. Finally a reference, if the FP-tree node is new, is established between it and the FP-Link entry. Therefore FP-Link maintains the total *support* of an element, and an FP-Tree maintains the *support* of an element in a particular sequence.

The insertion recursively continues inserting each element from $\bar{o}$ into the FP-structure in a similar manner. This is achieved by removing e_1 from $\bar{o}$ and making it the active node, thereby progressing down a level of the FP-Tree structure for each element in $\bar{o}$ until $\bar{o} = \emptyset$. This process is undertaken for each reduced and ordered object, resulting in an effective representation through the common path sharing of multiple objects. Due to path sharing within FP-Tree, the FP-structure representation typically becomes more compressed (and therefore optimised) as dataset density increases.

The subsequent mining process recursively constrains the FP-structure until the resultant constrained FP-Tree contains a single path, at which point valid elementsets are generated from the combinations of the set of participant elements. Each elementset's *support* being that of the least frequent element in that combination.

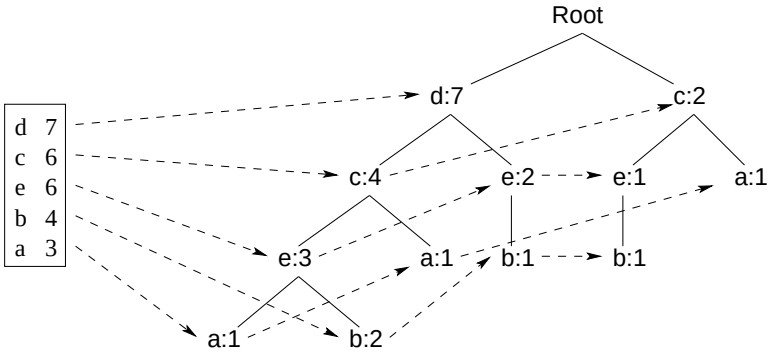
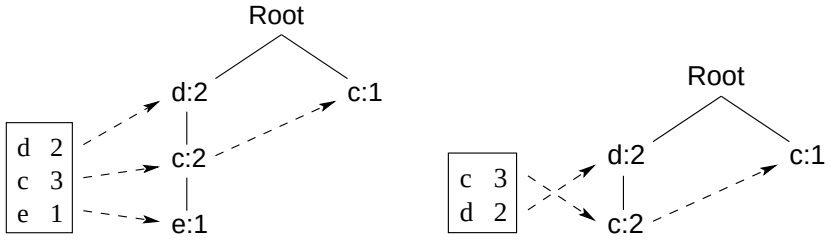


Figure 1.10: FP-Growth: FP-Tree and FP-Link

Given a branched FP-Tree, a set of constrained FP-Trees are generated for each participant elements, represented within FP-Link and processed from least to most frequent. For each element a conditional pattern base is derived that contains the prefix patterns of the specific element within the current FP-Tree. This pattern base then replaces the current object set ($\bar{o}$) as the data source for the constrained FP-structure. Subsequently the new FP-structures are built from the conditional pattern base and then pruned removing any elements with insufficient *support*.

For example, the FP-Tree in Figure 1.10 consists of many paths and therefore conditional FP-structures are required to derived the valid elementsets. By processing the FP-Link bottom-up the algorithm first generates the a-constrained FP-structure, or a-struct (illustrated in Figure 1.11(a)). The conditional pattern base of a-struct is based upon a's prefix patterns in FP-Tree, $\{(dce : 1), (dc : 1), (c : 1)\}$, where the count relates to the co-occurrence of a with these elementsets within FP-Tree. After construction a-struct is pruned, removing all elements with insufficient support, resulting in Figure 1.11(b).

As a-Tree is still branched, the constraint processes repeats. Starting with element d, this results in an empty conditional pattern base and the derivation of the elementset $ad : 2$. The next element, c is then processed resulting the conditional pattern base $\{(d : 2)\}$ and the elementset $ac : 3$. The ac-struct is then generated resulting in ac-Tree that consists of a single node $(d : 2)$. Since ac-Tree has a single path, further constraint is not required and valid elementsets are derived from all path combinations in union with the constraint set ac, resulting



(a) Initial (b) Pruned & Ordered

Figure 1.11: a-constrained FP-structures

in the generation of the elementset $acd : 2$. The valid elementsets therefore generated in relation to a are $\{(a : 3), (ad : 3), (ac : 3), (acd : 2)\}$. The algorithm recursively continues until FP-Link has been traversed.

FP-Growth has been shown to be effective in the mining of dense datasets as the FP-structure concisely encapsulates valid element information. However as the number of different elements, $|E|$, increases linearly, the size of FP-Tree typically expands exponentially due to the reduced sharing of common prefixes, reducing efficiency. To address this FP-growth* was proposed by Grahne & Zhu (2003), which uses an additional array based structure to reduce the number of tree traversals required during analysis. This array based structure saves on general traversal times and enables the direct initialisation of the next level of FP-Trees. However its instantiation is generally not warranted in dense datasets or in the first levels of recursively constructed FP-Trees from sparse datasets as they are based upon the most common prefixes available. Therefore a density heuristic was devised to determine the benefit of constructing the array.

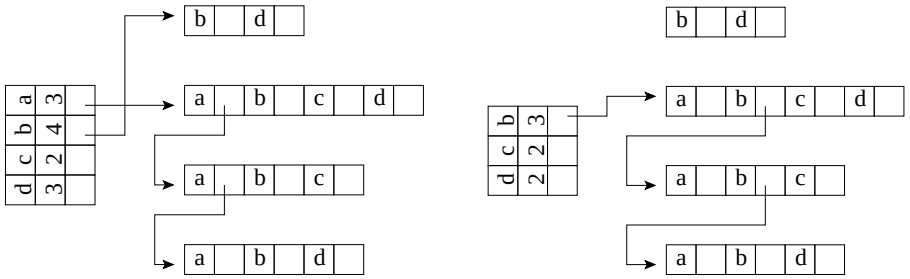
A top-down variation of FP-Growth (TD-FP-Growth) is proposed by Wang *et al.* (2002) that alleviates the need to generate conditional pattern bases and physical projections of the trie. The algorithm constructs the trie in a similar fashion to FP-Growth, however it then processes the FP-List top-down, mining through the recursive creation of conditional FP-Lists, which all refer to the global FP-Tree.

H-Mine (Pei, Han & Lakshmanan 2001) extends the pattern growth concepts introduced in FP-Growth, however an array based hyper-structure is used. Population of the hyper-structure occurs in a similar manner to FP-Growth with the

first scan identifying V_1 and the second creating the H-struct hyper-structure from it. However if the constructed H-struct cannot fit into memory, D is partitioned, in a similar manner to the Partition Algorithm, each of which is individually analysed and subsequently consolidated.

The two parts of H-struct are the linked projected object set P and header table H, relating to the pattern-frame and element-list respectively. Each $p \in P$ contains the valid elements of an object, derived from V_1. The header table H contains a list of the valid elements F and their counts. During construction all P that begin with the same element are linked together in a queue, the head of which resides in the header table H (Figure 1.12(a)).

Analysis is conducted through a traversal of H creating conditional H-structs for each header element that is a queue head within P. This process is recursive, progressively creating further constrained H-struct until it is empty. This process is effectively the same as the DFT of an enumeration-set structure, where all V_κ are identified at that level of constraint through analysis of the header table. This analysis process is illustrated in Figure 1.12, with Figure 1.12(a) illustrating a populated initial H-struct and Figure 1.12(b) presenting its constraint in relation to element a.



(a) H-Struct (b) Constrained header H_a

Figure 1.12: H-Mine data structure

From this the advantage of H-Mine over FP-Growth is apparent as the same pattern-frame structure is used with semantics being changed through pointer manipulation, rather than the creation of new constrained FP-structures required in FP-Growth. Figure 1.12(b) presents a-struct from which valid elementsets

containing a of length 2 are represented in H, resulting in $ab:3, ac:2, ad:2$. Analysis proceeds in this manner until all of H has been processed resulting in the identification of V.

ITL-Mine (Gopalan & Sucahyo 2002) optimises H-Mine by requiring only a single dataset scan, reducing I/O, and through the maintenance of a static set of links in the hyper-structure. The single scan of D creates the underlying structures, that are similar to H-struct, except that the header tables maintain all elements not just those that occur first in any $p \in P$. This additional linkage avoids the progressive recalculation of linkages during processing that is apparent in H-Mine. After the first pass V_1 is derived from the header counts and the associated list structure is reduced by removing the invalid elements.

Subsequent analysis is recursive, for $\kappa = 2$ the analysis is prefix based, involving a traversal of P, during which a temporary structure, *ITL-List*, is created and subsequently used in $\kappa > 2$ analysis. ITL-List initially contains those elements that co-occur with the specified prefix element, their co-occurrence count and relevant tidList. After generating V_2 the tidLists of each elementset are available in the temporary structure and hence, $V_\kappa \mid \kappa > 2$, that extend the designated prefix can be recursively generated using intersection and ITL-List.

CT-ITL (Sucahyo & Gopalan 2003) extends ITL-mine by using a compressed pattern framework structure to reduce storage space and traversal overheads, although requiring two dataset scans it has shown to become more scalable than ITL-Mine as dataset size increases. Based upon a reordered header structure built during the first pass, the algorithm creates a compressed prefix tree, containing all possible elementsets, based upon V_1. This structure compresses the regular prefix tree structure (Figure 1.3(b)) by storing identical sub-trees within a single set of nodes and storing the additional required information within a node associated array.

Given $E = abcd$, Figure 1.13 illustrates the resulting prefix structure. The figure's array structure (annotated with dashed structures to facilitate understanding) shows that each node array element maintains the count for a unique elementset, with the array index specifying the tree level at which accrual for that particular elementset begins. For example, given the node *abc*, the array consists

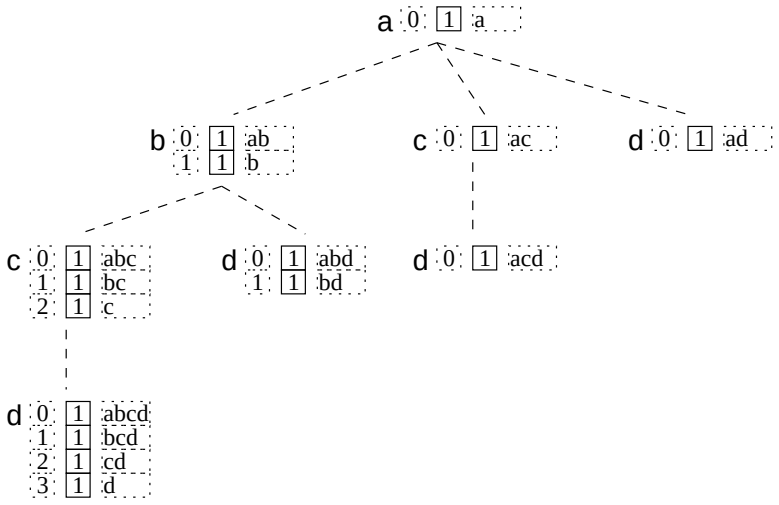


Figure 1.13: CT-ITL compressed tree structure

of three elements indicating the counts for the elementsets abc, bc and c respectively. The resulting compressed prefix structure is then populated by scanning D and inserting all objects, which have been pruned so that all invalid elements are removed, resulting in $\bar{D}$. The tree then represents an object summary that is used to create a similarly compressed projected object list structure.

Instead of representing each object as a row within P, as in H-Mine, compressed ITL, derived from the compressed prefix-tree structure, is comprised of the set of Maximal Object Sets (MOS) that encapsulate the required object information. In a similar manner to the compressed tree, an associated array maintains each MOS's subset count, thereby maintaining the count of each unique object. In this way the array index specifies the MOS element from which accrual occurs, in order to produce the elementset to which the specific count refers. For example, given the MOS abd the count associated with the second array element refers to $|o| \mid o = \{b, d\} \ \wedge \ o \in \bar{D}$. Subsequent mining is recursive and similar to ITL-mine, however the temporary structure used and the intersection method are slightly modified to accommodate summarised objects.

Opportunistic Projection (Liu, Pan, Wang & Han 2002) is a hybrid pattern growth algorithm that extends the authors prior work, namely FP-Growth and H-Mine. Mining occurs through the construction of *FIST*, a prefix tree with associated counts. The algorithm constructs the tree using classical BFT and

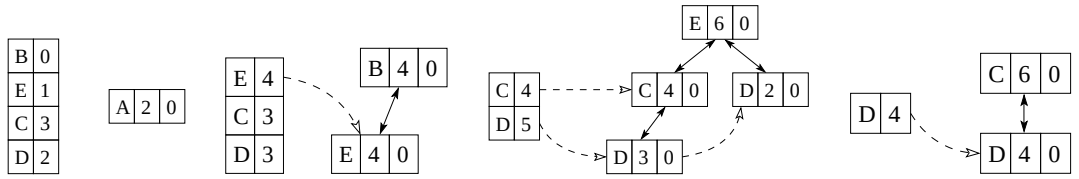
dataset reduction techniques until the data structures required for further mining can be held in memory, from which point memory resident projection based techniques are used.

Processing is based upon FP-Tree or H-struct, depending upon the density of the projected transaction set *PTS*. For small κ, the *PTS* elements have less chance of prefix sharing and hence relative *support* and density is low, however as κ increases the size of *PTS* decreases and relative support increases. As noted by the authors, FP-Tree compresses poorly in low relative support situations and therefore is used for high κ, whilst H-struct, which has a linear growth, is a better choice for small κ. *PTS* elements reduce quickly for small κ and therefore H-mine filtering is incorporated whilst using H-struct. However *PTS* reduction slows as κ increases, reducing the filtering benefit and making the creation of conditional FP-Trees unwarranted. To reduce FP-Tree filtering costs, the algorithm introduces the concept of Pseudo Projections, whereby the reduction factor can be determined and only if warranted will a conditional FP-Tree be created.

COFI proposed by Osmar and El-Hajj (2003) uses the concept of Co-Occurrence Frequent Item trees to provide an efficient pattern growth algorithm that uses a top-down non-recursive technique. The algorithm first constructs structures similar to FP-Growth, except that the FP-Tree is double-linked allowing for fast traversal without recursion. Mining proceeds through the construction and analysis of COFI trees, each of which is based upon a valid FP-Link element (processed in order of ascending frequency). Each COFI tree is then mined independently, without recursively building further constrained subtrees.

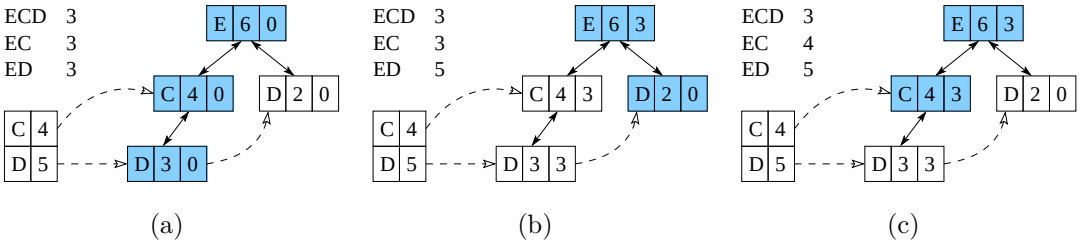
A COFI tree, based upon an element x, is constructed by traversing FP-Tree for each element occurring after x in FP-Link, finding their level of co-occurrence with x. If locally valid with respect to a the element is subsequently used in the construction of x-COFI. Given the FP-Tree illustrated in Figure 1.10, subsequent analysis (given a support of 4) results in the COFI trees presented in Figure 1.14, from which the similarity to constrained FP-Trees is evident. Each COFI node consists of an element its occurrence in relation to the base element and a participatory count used in subsequent mining.

Each COFI tree is sequentially constructed, mined if locally valid elements exist and discarded before the invocation of the next COFI tree, minimising memory usage. This mining process is discussed in respect to Figure 1.15, which is based upon *e*-COFI (Figure 1.14(c)). Processing occurs in descending FP-Link frequency order, where each element's link list is traversed and each participant branch analysed separately. The first step (Figure 1.15(a)) therefore involves analysis of the branch ECD from element D, where the frequency of a branch is the frequency value less the participation of the specific element, therefore the frequency of ECD is 3. The participation values of all nodes in the branch are incremented by 3 and all sub-patterns of the branch are generated with the same frequency value.



(a) *a*-COFI (b) *b*-COFI (c) *e*-COFI (d) *c*-COFI

Figure 1.14: COFI trees based upon FP-Tree in Figure 1.10



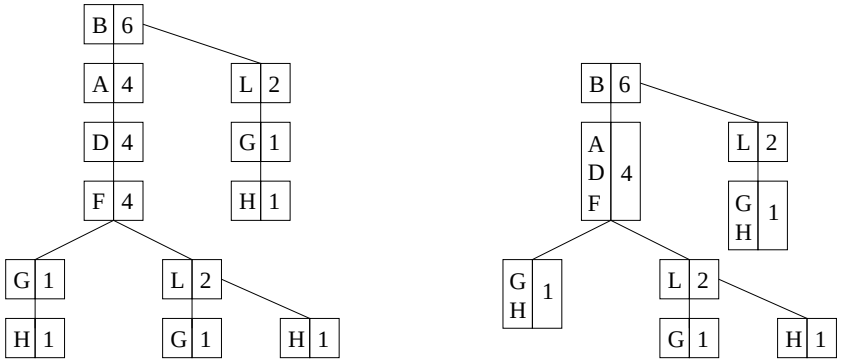
(a) (b) (c)

Figure 1.15: COFI tree mining based upon *e*-COFI in Figure 1.14

In Figure 1.15(b) the branch ED is analysed, as this elementset already exists, it's count is consequently updated ($ED : 5$) and the participation count of D in branch ED updated. Figure 1.15(c) presents the analysis of branch EC which has a frequency value of 1 and already exists, resulting in $EC = 4$. *e*-COFI is then complete resulting in the generation of the elementsets $\{ECD : 3, EC : 4\, ED : 5\}$ and the next COFI tree can then be constructed and mined. COFI results in the

accurate generation of all valid elementsets, while using less memory than FP-Growth due to it's non-recursive analysis process.

PatriciaMine (Pietracaprina & Zandolin 2003) proposes the use of a compressed trie, known as a Patricia trie, to alleviate the need to swap between trie and array based data structures based upon dataset density as proposed in H-mine and Opportunistic Projection. This single structure maintenance eliminates the need to switch between structures significantly reducing processing overheads. Furthermore, it is claimed that the compression technique used results in a structure comparable (with regard to size) with array based structures in the presence of sparse datasets.



(a) Regular Trie (b) Patricia Trie

Figure 1.16: Patricia Trie example from Pietracaprina & Zandolin (2003)

A Patricia trie is a modification of a regular trie in which each maximal chain of nodes that have a common count (support) are coalesced into a single node that inherits the count and stores the elements in the same sequence, Figure 1.16 illustrates this through a common dataset representation using both a regular and Patricia trie.

The implementation of PatriciaMine also incorporates the constraint of physical projections in a similar manner to Opportunistic Projection and uses novel mechanisms to directly generate groups of elementsets supported by the same transaction subset.

Table 1.1: Summary of Classic Algorithms

Name	Author	Year	Org.	Structure	Scans	Contribution
Candidate generation algorithms: Merge based or BFT						
AIS	Agrawal	1993	HL	-	k	First association mining algorithm
SETM	Houtsma & Swami	1993	HL	-	k	Merge & count
Apriori	Agrawal & Srikant	1994	HL	Hashtree	k	DCP
OCD	Mannila etal.	1994	HL	Hashtree	k	Merge operators
AprioriTID	Agrawal & Srikant	1994	VL	Hashtree	1	TID
Apriori *ext*	Agrawal & Srikant	1994	HL	Hashtree	$\leq k$	Delayed accrual
Partition	Savasere et al.	1995	VL	Hashtree	≤ 2	Partitioning
SPINC	Mueller	1995	-	-	≤ 2	Partitioning & Incremental candidate construction optimisation
DLG	Yen & Chen	1996	VV	Graph	$\leq k$	Association graph
DIC	Brin et al.	1997	HL	Hashtree	$\leq k$	Partitioning & checkpoints
DHP	Park et al.	1997	HL	Hashtree	k	Possible frequent elementsets & delayed accrual
AS-CPA	Lin & Dunham	1998	-	-	≤ 2	Partitioning & Anti-skew
NBM	Gardarin et al.	1998	VV	-	k	Indexing
HBM	Gardarin et al.	1998	VV	-	k	Second level indexing
ColumnWise	Dunkel & Soparker	1999	HV	-	k	Column based intersections
PHP	Ozel & Guvenir	2001	HL	Hashtree	k	Perfect hashing
DCP	Orlando et al.	2001	HL	Hyper	k	Dataset pruning

Table 1.1: Summary of Classic Algorithms (continued)

Name	Author	Year	Org.	Structure	Scans	Contribution
Candidate generation algorithms: Extension based or DFT						
Eclat	Zaki	1997	VV	Lattice	2	Prefix-based equivalence relations
Clique	Zaki	1997	VV	Lattice	2	Maximal cliques
AprClique	Zaki	1997	HV	Lattice	2	Maximal cliques
Tree Projection	Agrawal et al.	1999	HL	Trie	1	Projection, matrix accrual & BFT and Hybrid versions.
Apriori-df	Pijls	1999	VV	Trie	$V_1 - 1$	DFT trie construction
Depth Project	Agrawal et al.	2000	VV	Trie	1	Novel accrual techniques
Part. Sup. Tree	Goulbourne et al.	2000	HL	Trie	1	Novel accrual
dEclat	Zaki & Gouda	2001	VV	Lattice	2	Inclusion of diffsets
Candidate generation algorithms: Hybrid						
AprioriHybrid	Agrawal & Srikant	1994	HL⇒VL	Hashtree	$< k$	
Hybrid	Hipp et al.	2000	HL	Hashtree	$< k$	Improved strategy switching
VIPER	Shenoy et al.	2000	HL⇒VV	Hyper	$< k$	Compression
DCI	Orlando et al.	2001	HL⇒VV	Hyper	$< k$	Reduced memory useage
kDCI	Lucchese et al.	2003	HL⇒VV	Hyper	$< k$	Adaptive algorithm behaviour
Apriori-Brave	Bodon	2003	HL⇒VL	Trie	$< k$	Reduced Storage & memory management

Table 1.1: Summary of Classic Algorithms (continued)

Name	Author	Year	Org.	Structure	Scans	Contribution
Pattern Growth Algorithms						
FP-Growth	Han & Pe	2000	HL	Hyper	2	Trie based
H-Mine	Pei et al.	2001	HL	Hyper	2	Array based & partitioning
TD-FP-Growth	Wang et al	2002	HL	Hyper	2	Top down processing
ITL-Mine	Gopalan & Sucahyo	2002	HL	Hyper	1	
Opp. Projection	Liu et al.	2002	HL	Hyper	2	Hybrid: FP-growth & H-Mine
FP-Growth*	Grahne & Zhu	2003	HL	Hyper	2	Traversal reduction for sparse datasets
CT-ITL	Gopalan & Sucahyo	2003	HL	Hyper	2	Compressed structures
COFI	Osmar & El-Hajj	2003	HL	Hyper	2	Co-occurence Frequent Itemset trees
PatriciaMine	Pietracaprina & Zandolin	2003	HL	Hyper	2	Patricia trie

1.3 Specialised Algorithms

In conjunction with classic elementset identification algorithms, significant research has been undertaken into variations that attempt to improve the quality of results either by facilitating user interpretation or by incorporating domain knowledge. There are four prominent areas of specialisation research that this section provides a survey of: 1) condensed representations, 2) incomplete sets, 3) semantic inclusion and 4) incremental mining.

Condensed representation and incomplete set algorithms improve mining time and reduce result set size improving usability. Condensed representation algorithms produce a reduced result set from which the complete set of valid elementsets can be derived. Incomplete set algorithms produce a reduced result set that can provide useful (although incomplete) information about dataset inferences. Semantic inclusion focuses upon incorporating domain knowledge within the analysis process to improve the result usefulness.

1.3.1 Condensed Representation Algorithms

Condensed representation algorithms produce a reduced result set from which all valid elementsets can be derived. There are four techniques used to produce condensed representations: closed sets, counting inference, deduction rules and freesets.

Closed Elementset Algorithms

Closed elementset algorithms identify a subset of valid elementsets from which all valid elementsets can be derived without further mining. The theoretical foundation of closed elementset algorithms is based upon the closure of the Galois connection (Ganter & Wille 1999), in which a closed pattern is the largest pattern common to a set of objects within D. All non-closed patterns have the same critical properties (in this case *support*) as its closure, which is the smallest closed pattern containing it.

The closure of an elementset a, denoted $cl(a)$ is then the smallest closed pattern containing a. Given that $g(o)$ is the set of elements common to all object in $o \mid o \subset O$ and $f(a)$ is the tidList of all elements in a. The closure of any elementset a is then found by constructing the set of objects in which the elementset a appears in $f(a)$ and from this calculating the set of elements $\bar{a}$ that are common to all objects in $f(a)$ by applying $g(f(a))$. From this $\bar{a}$ must be the closure of a as $g(f(a)) = \bar{a} = cl(a)$, as it identifies the set of elements that always occur with a, hence $\bar{a} \supseteq a$ and $\sigma(\bar{a}) = \sigma(a)$.

The following algorithms discover the closed set of valid elementsets, Cl within D using a variety of techniques, from which V can be derived. The inclusion of closure constraints optimise analysis through search space reduction, especially for highly correlated datasets.

The derivation of V from Cl is simple, however since Cl implies V the generation and presentation of closed rules may facilitate user interpretation due to the reduced number. Closed rules, proposed independantly by (Pasquier, Bastide, Taouil & Lakhal 1999*b*) and (Zaki & Ogihara 1998), are a reduced set of inferences derived from Cl, where given a closed elementset a, an inference r is of the form $x \Rightarrow a - x \mid x \subset a \wedge a \in Cl \wedge x \in Cl$ the confidence of which, $\gamma(\sigma(a)/\sigma(x))$, is available.

A-Close (Pasquier et al. 1999*b*) uses a candidate generation BFT approach consisting of two stages, generator set discovery and closed set derivation. The generator set (G) is constructed in regular Apriori fashion however it incorporates additional pruning such that if a $G^i_{\kappa+1}$ has the same *support* as any of its κ subsets, then based upon the Galois connection it has the same closure and is removed from $G_{\kappa+1}$. Therefore G contains all locally closed sets.

The second stage calculates the global closure of each generator by performing an intersection of all objects in which the generator occurs. Figure 1.17 illustrates the identification of G_1 and G_2 with pruned elementsets, due to infrequency and subset *support* equivalency, crossed out. The rightmost table presents the final closed set Cl that identifies all global closed elementsets.

A preceding algorithm Close (Pasquier, Bastide, Taouil & Lakhal 1999*c*) identifies the closed elementset for each generator as it is discovered, instead separating out global identification. However apparently A-Close is more efficient upon dense datasets.

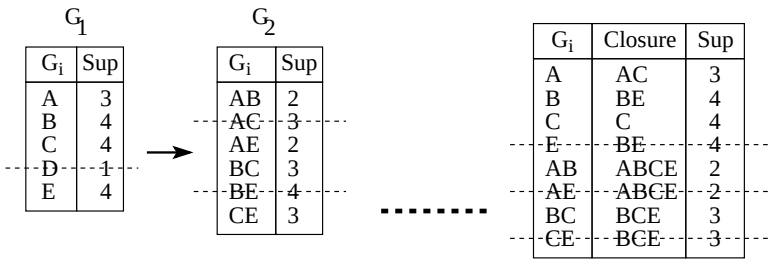


Figure 1.17: A-Close Process diagram (Pasquier *et al.* 1999*b*)

CLOSET (Pei, Han & Mao 2000) is based upon pattern growth concepts and uses similar data structures and processes as described in FP-Growth, however it uses introduces additional pruning techniques to discover only the set of closed elementsets (CL). This subsumption testing ensures that a new valid elementset is only appended to Cl if it is not subsumed by an existing member, while candidates cannot subsume existing members due to the bottom-up order of processing implemented within FP-Growth. Furthermore CLOSET facilitates closed set discovery through the inclusion of two process reduction techniques that result in search space reduction based upon closed set characteristics, presented below.

- Given a (a)-constrained dataset D_a, if an element or elementset ,b, appears in all D_a objects then the closure of $a \cup b$ will subsume the closure of a. Therefore b is removed from D_a and appended to the constraint, therefore D_a becomes $D_{a \cup b}$.
- Given a valid closed set a and the subsequent discovery of a closed elementset b such that a subsumes b then D_b is not processed as any resultant closed elementsets will be subsumed by a.

Subsequent extensions to CLOSET have been proposed in CLOSET+ (Wang, Han & Pei 2003) and FPClose (Grahne & Zhu 2003*a*). CLOSET+ introduces a hybrid tree projection method, incorporates item (or element) skipping and proposes two efficient subset-checking structures. The hybrid tree projection method proposes the use of a top-down pseudo-projection technique in the case of sparse datasets, through which the FP-Tree is not physically replicated for each conditional base. Item skipping prunes the search space by recognising that if an element has the same *support* in different conditional FP-Lists then that element can be removed from consideration in the higher level FP-Lists. FPClose extends CLOSET through the creation of a local Closed Frequent Set trees (CFI-tree) for each conditional FP-Tree. Therefore instead of performing global subset checking, local checking is performed first, significantly reducing subset testing in large datasets (see FPMax*, Section 1.3.2).

---

1. If $o(a) = o(b) \mid a \cap b = \emptyset$ then $cl(a) = cl(b) = cl(a \cup b)$. This implies that all occurrences of a can be replaced with $a \cup b$ and b can be removed from further consideration as its closure $cl(b) = cl(a \cup b)$.

2. If $o(a) \subset o(b)$ then $cl(a) \neq cl(b)$ but $cl(a) = cl(a \cup b)$ implies similar replacement, however as $cl(b) \neq cl(a)$, b cannot be removed from further consideration.

3. If $o(a) \supset o(b)$ then $cl(a) \neq cl(b)$ but $cl(b) = cl(a \cup b)$, the inverse of Property 2 is true in which b is replaced by $a \cup b$ instead of a, but a is not removed from consideration.

4. If $o(a) \neq o(b)$ then $cl(a) \neq cl(b) \neq cl(a \cup b)$, indicates that both a and b lead to different closures and hence no replacement occurs.

CHARM (Zaki & Hsiao 2002) is similar to CLOSET in its general approach, however it uses an enumeration-set tree and the concept of equivalence classes introduced in Eclat (Section 1.2.1). Where two elementsets belong to the same κ equivalence class if they share a common $|\kappa|$ prefix. CHARM extends Eclat to discover only closed elementsets, through the introduction of a set of closure prop-

erties that are presented above where $o(a)$ indicates the set of objects containing elementset a. Where properties 1 and 2 result in equivalence class reduction and hence facilitate closed-set convergence, whilst 3 and 4 result in new equivalence class information that generally requires additional processing. The inclusion of these properties within Eclat based processing occurs after the discovery of a new valid elementset a, whereby the closure properties are then applied to control a's influence upon the generation of the closed-set equivalence class lattice.

Counting Inference

Counting inference (Bastide, Taouil, Pasquier, Stumme & Lakhal 2000) (implemented as PASCAL, an Apriori extension) is based upon the observation that elementsets can be considered equivalent if they are included in the same set of objects, referred to as a class. Mining can then be reduced to the identification and validation (*support*) of the set of *Key Patterns*, where a *Key Pattern* is a minimal elementset within a class from which the validity (*support*) of all other elementsets can be inferred. A pattern can only be a *Key Pattern* if all its subsets are *Key Patterns*, that is, no subset belongs to the same equivalence class. This theory leads to two analysis optimisations, especially in highly correlated datasets (presented below).

- Only the Key Patterns need to be considered during each dataset scan as non-key elementset support is equal to the minimum support of its $\kappa - 1$ subsets.
- The number of dataset scans will often be reduced because the set of Key Patterns is often found before the last iteration because if a single $\kappa - 1$ subset of a candidate elementset is non-key then the elementset is non-key, hence from a certain level all elementsets may be known to be non-key and therefore can be derived from subsets without requiring another dataset scan.

Deduction Rules

Calders and Goethals (2002) concisely represent the set of valid elementsets through a set of deduction rules using the mathematical inclusion-exclusion principle. The rules are used to derive tight bounds on the *support* of an elementset given that the *support* of its subsets are known. The concise representation used is the set of valid elementsets whose *support* cannot be derived, the set of non-derivable elementsets, from the deductive rules. All other rules can be derived from these without scanning the dataset.

The implementation used extends Apriori, incorporating deductive rules to eliminate from consideration those elementsets whose *support* can be deduced, resulting in less mining time and a more concise result set of all non-derivable elementsets. From this result set, the *support* of all other valid elementsets can be found by using deductive rules. The algorithm also includes optimisations to calculate the complete set of valid elementsets.

Free-sets

Disjunction free-sets (Bykowski & Rigotti 2001) is a condensed representation based upon the theory of disjunction. A disjunctive rule is founded upon the concept that given elementset $A = \{abcd\}$, then $\sigma(A)$ can be derived from $\sigma(ab)$, $\sigma(a, b, c)$ and $\sigma(abd)$ as the sum of $\sigma(A)$ and $\sigma(ab)$ is equal to the sum of $\sigma(abc)$ and $\sigma(abd)$, given that ab does not exist in D without either c or d. The calculation of $\sigma(a)$ can therefore be derived from the support of these particular subsets, given the presence constraint, without scanning D, and is thus referred to as a non-disjunction free-set as it can be calculated from disjunctions. Hence mining is reduced to discovery of the set of disjunction free-sets within the search space, where disjunction free-sets also display non-monotonicity in that if an elementset is a not disjunction free then no superset of it will be either.

By using the discovered disjunction-free sets in conjunction with the negative border set, which consists of the smallest elementsets that are not valid disjunction-freesets, the validity of any elementset can be derived without scanning the dataset. The authors implement both DFT and BFT implementations of the algorithm and claim it superior to closed-sets.

δ-Freesets (Boulicant, Bykowski & Rigotti 2001) provide a condensed representation from which the *support* of any valid elementset can be closely approximated. The level of approximation is shown to be acceptable for many mining tasks and the trade-off against mining time advantageous. A δ-Freeset is an elementset such that its participant elements cannot be used to form a strong rule, where strength is a user defined metric (δ). For example, given elementset abc if $ab \Rightarrow c$ is a strong rule (it holds with less than δ exceptions), then $\sigma(abc)$ can be approximated using $\sigma(ab)$. δ-Freesets are also anti-monotonic as if an elementset A is not a δ-Freeset then B is not either given that $B \supset A$.

The implementation, MineEx, is an Apriori based extension, constraining results to those elementsets that are δ-Free. Once analysis is complete any elementset's validity can be approximated by finding the smallest support amongst its valid δ-free subsets.

1.3.2 Incomplete Set Algorithms

Incomplete set algorithms reduce analysis by only discovering incomplete information about the complete set of valid itemsets within D. There are two significant types of incomplete set algorithms: sampling and maximal valid sets. Sampling algorithms analyse only a portion of the dataset and Maximal Frequent Set algorithms (MFS) identify only those valid elementsets, for which no valid supersets exist. Therefore sampling works by reducing $|D|$ and MFS works through the introduction of new pruning strategies. [1]

Maximal Frequent Set Algorithms

Maximal Frequent Set (MFS) algorithms identify all elementsets within D for which no valid supersets exist. Therefore although the result set implies all valid

[1] While MFS relates to the inclusion of the *support* heuristic (Frequent), the pruning strategies employed are generically applicable to any analysis that incorporates non-monotonic heuristics. However due to general usage this algorithmic group will be referred to as MFS instead of *maximal valid sets*, which is more correct given the broader range on heuristics that can actually be applied.

Upward Closure Principle (UCP) (Bayardo 1998) Given a valid elementset A with a set of possible extensions B, if the elementset $C \mid C = A \cup B$ is valid then C becomes the *terminal* MFS of A and no further exploration of A's supersets is required.

Superset Checking (Bayardo 1998) An extension of UCP that avoids direct counting of A, by checking if A is subsumed by or is a subset of an existing MFS, if so no further exploration of A's supersets is required.

Parent Equivalence Pruning (Burdick, Calimlim & Gehrke 2001) Given an extension based analysis algorithm incorporating dataset projection, then for each child elementset generated in the enumeration-set tree, the projected object set is compared with that of its parent. If they match, the child can replace the parent node.

---

elementsets through the identification of the search space boundary, the actual *support* and hence inference strength of internal elementsets (those within the search space boundary) remain unknown. The implementation of MFS algorithms result in reduced analysis time because MFS properties enable the inclusion of additional pruning strategies (presented on the next page) that facilitate search space reduction.

Maxminer (Bayardo 1998) extends Apriori through the inclusion of UCP and subsumption testing to derive the set of maximal frequent sets. Furthermore MaxMiner is underpinned by an an enumeration-set tree instead of a hash-tree, within which elements are dynamically sorted in order of increasing *support*.

Pincer Search (Lin & Kedem 1998) also extends Apriori through the introduction of a bi-directional search that has proven efficient in the discovery of long maximal valid elementsets. The algorithm uses a typical Apriori bottom-up search, but extends it by incorporating additional pruning through the use of a Maximal Frequent Candidate Sets (MFCS) that approaches the valid search space border and hence the discovery of MFS from the top-down.

Given that initially MFCS contains a single elementset of the union of all elements, analysis proceeds by iteratively generating the next bottom-up valid set V_κ, refining MFCS and then pruning V_κ, or $V_{\kappa+1}$ candidates, based upon those members of MFCS determined valid. For example, given $E = \{a, b, c, d, e\}$ and $V_1 = \{a, b, d, e\}$ then MFCS is refined, ensuring that no member of MFCS is a superset of an invalid elementset, which is provided by $\bar{V}_\kappa \mid \bar{V}_\kappa = C_\kappa - V_\kappa$. Therefore $MFCS_1 = \{abde\}$ as element c is determined invalid. The refined members of MFCS are then validated, those found valid are MFS and all subsets within $C_{\kappa+1}$ are removed from further consideration. Therefore if $\{abde\}$ is found valid, then through pruning $V_1 = \emptyset$ and $\{abde\}$ is identified as the only MFS.

This process of MFCS refinement and $C_{\kappa+1}$ pruning greatly reduces $|C|$ and facilitates MFS convergence, however in some cases it can be pre-emptive and recovery of some candidates may be required. The recovery process is achieved by creating additional $C_{\kappa+1}$'s for each MFCS member, where it is a superset of the $\kappa - 1$ prefix of a V_κ^i. Given this, a new candidate is created by merging each element in the MFCS that occurs after the last element of the $\kappa-1$ prefix with the $\kappa - 1$ prefix. Once recovery has been undertaken regular Pincer Search analysis continues with the derivation of $V_{\kappa+1}$.

Max-Eclat and Max-Clique (Zaki, Parthasarathy, Ogihara & Li 1997, Zaki 2000*b*) are based upon the Eclat and Clique algorithms (Section 1.2.1), however, by using either a DFT or hybrid, DFT-BFT, approach, all MFS are efficiently identified. Using DFT, processing of each decomposed lattice begins with the identified potential maximal elementset, whose validity (*support*) is determined from tidList intersection. If valid, the processing of that lattice is complete, otherwise each lattice subset is checked at the next level of decomposition until all maximal valid elementsets have been identified. The benefit of Max-Clique over Max-Eclat is that the identified potential maximal elementsets are more refined, as maximal cliques identify smaller sub-lattices than the equivalence class method used within Eclat. Hence less traversal is often required to discover all MFS using Max-Clique.

The Hybrid traversal search is based upon the concept that the greater the *support* of a valid elementset the more likely it is to be part of a longer valid

elementset. The sub-lattice elementsets are placed in descending frequency order, the hybrid phase then intersects the elementsets individually, stopping when an extension becomes invalid, identifying that sub-lattice's MFS. The benefit of this technique is that each extension only requires a two-way Tidlist intersection. However in both techniques the search space is not global and hence some of the discovered MFS may not be globally maximal; hence some post-processing may be required.

All-MFS (Gunopulos, Mannila & Saluja 1997) performs random walks within an enumeration-set tree to discover MFS. A walk starts with an initial elementset and subsequently extends using random element selection from those elements that lexicographically occur after the previously selected (extension) element. Each extension is subsequently validated using Tidlist intersection and, if invalid, a MFS has been discovered.

Analysis proceeds by deriving V_1 and removing from D all invalid elements. After a specified number of walks All-MFS calculates the set of Minimal Orthogonal Elements (MOE) or those elements that are not subsets of discovered MFS. MOE's are then used as the starting points for a set of random walks. Although All-MFS cannot guarantee the discovery of all MFS it does find a large portion of them.

Mafia (Burdick et al. 2001) extends DepthProject (Agrawal et al. 2000) (see Section 1.2.1) through the use of *Superset Checking* and *Parent Equivalency Pruning* to discover the set of MFI's. While GenMax (Gouda & Zaki 2001) takes a novel approach to maximality testing denoted *progressive focusing* where valid elementsets are first tested against a locally maintained set of MFI's (LMFI). Most non-maximal valid elementsets are discovered using the local testing therefore reducing the number of subset tests required. Furthermore GenMax also uses diffsets (See Section 1.1.2), which become more effective as density increases.

FPMax (Grahne & Zhu 2003*b*) provides an MFI version of FP-Growth, which uses superset checking to construct a axillary MFI-Tree. The algorithm proceeds in a similar fashion to FP-Growth beginning with the initial construction of a FP-Tree and FP-List, however subsequent conditional FP-Tree processing is only conducted where the conditional head together with all valid elements in the head-

conditional pattern base (tail) is not a subset of an existing MFI, stored within MFI-Tree. If it is not subsumed the conditional FP-Tree is constructed and if only consisting of a single path is appended to MFI-Tree. Analysis proceeds as for FP-Tree, incorporating superset checking optimisations based upon FP-structure and processing idiosyncrasies with the final set of MFI's represented in the MFI-Tree.

FPMax* (Grahne & Zhu 2003*a*) extends FPMax through the inclusion of progressive focusing (as introduced in GenMax) resulting in the maintenance of local MFI-Tree's and reducing subset testing in large datasets. Furthermore, additional subset testing and processing optimisations are implemented based upon the nature of the conditional-base *tail*.

Sampling Based Algorithms

Sampling based algorithms are based upon the premise that approximate answers often suffice and therefore adequate answers can be obtained by mining a lossy compressed representation of the data. The sampling approach addresses the issue of scalability by only analysing a representative subset of D. Work in the associated field of approximate aggregation (Gibbons 2001, Choudhuri, Datar, Motwani & Narasayya 2001) has shown that the benefits of sampling are improved when the sample selection is guided by the user, hence improving its relevancy to the current problem.

The main issue in developing sampling techniques is to maximise the extent to which the sample reflects the generic characteristics of D while maintaining efficiency through sample size constraint. A trade-off is therefore apparent between accuracy and efficiency in naive sampling, a formal discussion of which is presented in (Kivinen & Mannila 1994). Both OCD (Mannila et al. 1994) and AS-CPA (Lin & Dunham 1998) (see Section 1.2.1) propose naive sampling extensions to their classic algorithms. However the following algorithms go further by attempting to reduce the errors resulting from naive sample selection.

Toivonen (1996) introduces a sample based analysis technique that typically only requires a single scan of D to discover all elementsets. The algorithm first uses a random sample and a reduced validity threshold in an attempt to calculate

the superset of valid elementsets within D. Based upon this, the subsequent completion of the D scan is used to discover the true set of valid elementsets.

The FAST sampling algorithm (Chen, Haas & Scheuermann 2002), attempts to reduce sampling errors by introducing a two-stage sampling technique. The first stage quickly estimates the validity of each element within D through the use of a large initial sample. The second stage derives from this a smaller sample set, within which each element's validity closely represents its validity within D. By calculating element validity based upon a large sample size in the first stage, it is proposed that the derived valid elementsets in the second stage will be close to the actual set of valid elementsets.

Table 1.2: Summary of Condensed Representation & Incomplete Algorithms

Name	Author	Year	Base	Principle	Contribution
Condensed Representation Algorithms					
Close	Pasquier et al.	1999	Apriori	Closed Sets	Galois Connection
A-Close	Pasquier et al.	1999	Close	Closed Set	
CLOSET	Pei et al.	2000	FP-Growth	Closed Set	
PASCAL	Bastide et al.	2000	Apriori	Counting Inference	
MinEx	Boulicant et al.	2001	Apriori	δ-free sets	
HLinEx	Bykowski & Rigotti	2001	Apriori	Disjunction free sets	
VLinEx	Bykowski & Rigotti	2001	Depth Project	Disjunction free sets	
CHARM	Zaki & Hsiao	2002	Eclat	Closed Set	
NDI	Calders & Goethals	2002	Apriori	Deduction rules	
CLOSET+	Wang et al.	2003	CLOSET	Closed Set	Hybrid tree projection, item skipping & efficient subset-checking structures
FPClose	Grahne & Zhu	2003	CLOSET	Closed Set	Optimised subset testing

Table 1.2: Summary of Condensed Representation & Incomplete Algorithms (continued).

Name	Author	Year	Base	Principle	Contribution
Incomplete Set Algorithms					
OCD	Mannila et al.	1994	OCD	Sampling	Naive extension
-	Toivonen	1996	Partition	Sampling	Negative border
All-MFS	Gunopulos et al	1997	Depth Project	MFS	Random walks
MaxMiner	Bayardo	1998	Apriori	MFS	Upward Closure Principle
Pincer Search	Lin & Kedem	1998	Apriori	MFS	Bi-directional search
A-CPA	Lin & Dunham	1998	AS-CPA	sampling	Naive extension
Max-eclat	Zaki et al.	2000	Eclat	MFS	
Max-clique	Zaki et al.	2000	Clique	MFS	
MAFIA	Burdick et al.	2001	Depth Project	MFS	Accrual optimisation
GenMax	Gouda & Zaki	2001	Depth Project	MFS	Progressive focusing & diffsets
FAST	Chen et al.	2002	-	sampling	Two-phase sampling technique
FPMax	Grahne & Zhu	2003	FP-Growth	MFS	
FPMax*	Grahne & Zhu	2003	FPMax	FP-Growth	Progressive focusing & Optimised subset testing

1.3.3 Accommodating Domain Knowledge

The incorporation of domain knowledge or semantic information within valid elementset identification can significantly improve the quality of results through the supplement of pertinent information. There are four common types of semantics embedded within association analysis that are discussed within this review; hierarchical, temporal, spatial and privacy. While privacy semantics are generally incorporated within the dataset at a pre-processing stage, the other types require the inclusion of additional datasets or semantically supplemented datasets within analysis.

Hierarchical

The inclusion of hierarchical semantics within elementset identification enables the discovery of the inferences at different levels of concept abstraction as supplied by a defined concept hierarchy. A concept hierarchy provides a representation of a hierarchical grouping of items within a system. For example, according to the example hierarchy shown in Figure 1, *Whiskey* is a child of *Alcohol*, which in turn is classified as a *Drink*. However for any set of elements it is possible to define many different concept hierarchies. For example, *Cocoa* could be classified as a *Chocolate* as well as a *PowderedProduct*.

The incorporation of concept hierarchies or hierarchical semantics within association mining results in the production of generalised inferences, which provides two main benefits. Firstly, generalised inferences minimise information loss that results from increasing the support threshold and secondly they facilitate interpretation of the associations within the inference set.

For example, given $\sigma(x)$, the valid elementset [*Whiskey,Lemonade*] is discovered, resulting in the inference *Whiskey* $\Rightarrow$ *Lemonade*. However by increasing the support threshold to $\sigma(x + \epsilon)$, the elementset becomes invalid and is therefore excluded from consideration. The inclusion of a concept hierarchy allows the consideration of elementsets derived from different levels of abstraction, such as [*Whiskey, Soft Drink*] from which inferences such as *Whiskey* $\Rightarrow$ *Soft Drink* can be derived given that $\sigma($*Whiskey, Soft Drink*$) \geq max\{\sigma($*Whiskey, Lemon-*

ade), σ(*Whiskey, Cola*), σ(*Whiskey,Ginger Beer*)}. This elementset containing an abstract concept, although introducing a degree of uncertainty, implies that inferences may exist between the implied elements, such as *Whiskey* $\Rightarrow$ *Lemonade*, that did not meet the validity threshold. Therefore the information loss that occurs through increasing the support threshold is reduced through the inclusion of hierarchical semantics.

Hierarchical semantics can also facilitate result interpretation by presenting inferences that incorporate abstract concepts as defined by the concept hierarchy. This additional dimension produces inferences between groups of elements that may be of greater interest than inferences of a more specific nature. For example, within a particular mining session generalised inferences such as *Alcohol* $\Rightarrow$ *Soft Drink* $\sigma(84\%)$ may be more interesting to a marketing department than more specific inferences such as *Whiskey* $\Rightarrow$ *Lemonade* $\sigma(56\%)$.

Hierarchical association algorithms have been an area of research since 1994 when Han and Fu introduced the concept of the dynamic generation of concept hierarchies for knowledge discovery (Han & Fu 1994). The following year they also presented a series of algorithms for hierarchical mining known as the ML-T* family (Han & Fu 1995).

These algorithms were mainly concerned with intra-level mining, or mining within concept levels only, however this research did provide an algorithm for inter-level mining known as ML-T2. This algorithm incorporates the hierarchical semantics within D, so that each element is replaced with its hierarchical position. For example, the element *Whiskey* in Figure 1 would be replaced by the hierarchical element *Drink-Alcohol-Whiskey*. This integration of D and the concept hierarchy before analysis, reduced I/O and simplified analysis by locating all required information within a single source.

Subsequent analysis is based upon Apriori extensions. The algorithm begins by discovering the valid elements, V_1, relating to the first hierarchical level and then progressively deepening the level of hierarchical inclusion. The discovery of valid elementsets involving different level concepts occurs during the creation of V_2, by deriving V_2 from n concepts at all levels of abstraction. Derivation of the rest of V is based upon typical Apriori analysis.

Subsequent algorithms include work by Srikant and Agrawal (1997), Hipp *et al.* (1998), Thomas and Sarawagi (1998), Mao (2001), Ong *et al.* (2001) and (Ceglar, Roddick, Calder & Rainsford 2004). Srikant and Agrawal devised the algorithms Basic, Cumulate, Stratify, Estimate and Estmerge that extend the Apriori algorithm. Basic appends the generalised element concepts to each object in D, and then treats all elements the same way during Apriori processing. Cumulate improves this by filtering out redundant candidate elementsets containing an element as well as its ancestors. Stratify uses the hierarchy during processing to optimise pruning during candidate generation, by ensuring that the more generalised candidate sets, C^i, are processed first. This is achieved by associating each C^i with a concept depth and performing a scan of D for each concept level in increasing order. If an elementset is found invalid then its hierarchical descendant elementsets are removed from consideration. The Estimate and Estmerge algorithms extend the previous algorithms by incorporating sampling to estimate the support of candidates.

HND and HPtid (Ceglar et al. 2004) (see Appendix A) and Prutax (Hipp *et al.* 1998) are also based upon the Apriori algorithms but allow the dynamic generation of inferences as the elementsets are discovered. HND also provides variable support enabling the user to vary the support threshold across hierarchical levels, while HPtid and Prutax both use vertical dataset organisation. Furthermore, HPtid provides an effective technique for prioritised mining, allowing the user to specify elementsets of greater interest that they would like explored first.

Most hierarchical mining algorithms are based upon Apriori. However, Thomas and Sarawagi (1998) have devised techniques that accomplish analysis through the SQL querying of databases with incorporated hierarchical semantics. Recently, Mao (2001) and Ong *et al.* (2001) have developed hierarchical algorithms based upon the FP-Growth algorithm (Han, Pei & Yin 2000).

Spatial

The inclusion of spatial semantics allows the discovery of inferences between spatially and possibly non-spatially bound objects, enabling the implication of spatial connotations between elementsets. Examples of spatial connotations include

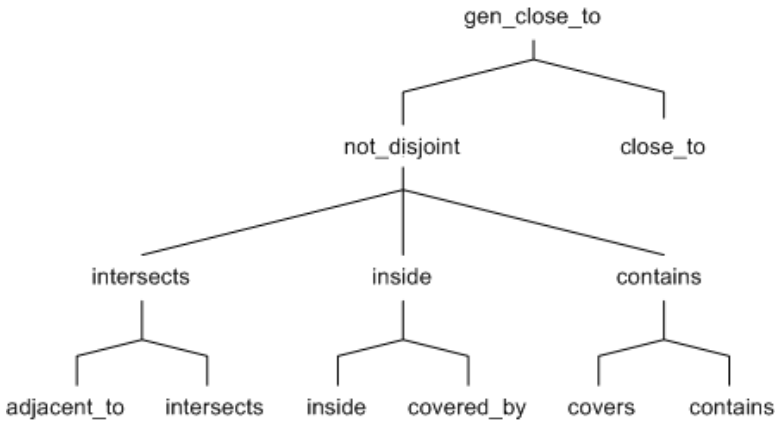


Figure 1.18: Topological Hierarchy (Koperski & Han 1995)

topology (*intersect, above*), orientation (*west_of, left_of*) and distance (*close_to*). Therefore given a discovered inference, the inclusion of spatial semantics results in the derivation of possibly multiple spatial inferences.

The seminal research into spatial inclusion was undertaken by Koperski & Han (1995) who proposed an Apriori extension that accommodated spatial semantics. The algorithm requires as input, a spatial dataset and a set of concept hierarchies. The spatial dataset contains a set of spatial objects with possibly additional non-spatial objects, where the spatial component refers to the object's existence within a geographic map, for example point or area. The inclusion of pre-defined concept hierarchies of both spatial and non-spatial attributes optimises the analysis by providing generalised concepts. An example topological hierarchy is illustrated in Figure 1.18. The subsequent analysis contains two stages of analysis, high-level spatial inference identification and refined mining.

The first stage identifies the high-level spatial inferences between the individual spatial objects using inexpensive spatial algorithms, the focus of which is to reduce the number of spatial inferences to be considered during further processing. At this point, simple inferences based upon these high-level spatial concepts may be derived, Rule 1, containing a single antecedent and consequent. However, often more detailed spatial inferences are required and hence refined mining is required.

Refinement occurs through the replacement of identified high-level spatial inferences with concrete spatial inferences that exist within the abstract spatial

concept. For example, given the spatial topology in Figure 1.18, through refinement the abstract spatial concept *gen_close_to* is replaced *by adjacent_to*, resulting in respective inference refinement shown in Rule 1 and 2. This derivation is based upon a new refined V^2 consisting of more concrete spatial concepts.

Subsequent mining is similar to Han and Fu's hierarchical mining extension of Apriori (Han & Fu 1995), in which after finding all valid elementsets at the highest concept level, valid elementsets are discovered at lower concept levels through continued concept substitution (see Section 1.3.3). For example, Rule 3, presents a more concrete inference based upon Rule 2.

Rule 1 $is\_a(x, tourist\_accomodation) \overset{gen\_close\_to}{\Rightarrow} (x, water)84\%$

Rule 2 $is\_a(x, tourist\_accomodation) \overset{adjacent\_to}{\Rightarrow} (x, water)72\%$

Rule 3 $is\_a(x, caravan\_park) \overset{adjacent\_to}{\Rightarrow} (x, lake)\%60$

Related research by Shekhar and Huang (2001) discovers co-location associations between area-based spatial features, where an area based spatial feature is referenced by a polygon, not a single point. Co-location inferences attempt to reflect spatial feature density by discovering the subsets of spatial features that are frequently located together. This process is accomplished by defining spatial neighbourhoods in which an event occurs and then mining the neighbourhoods to discover co-location inferences, using a BFT candidate generation approach. The algorithm incorporates two novel interestingness measures *prevalence* and *conditional probability* to reduce the search space. A further discussion of spatial data mining can be found in Abraham & Roddick (1999) and Miller & Han (2000).

Temporal

Often objects have a temporal aspect that is either explicit, such as a time-stamp attribute, or implied, such as a defined ordering or sequence of objects. The inclusion of temporal semantics within association mining can be sub-classed into the incorporation of temporal-qualifiers and temporal-predicates within association mining and sequence mining. Temporally qualified analysis relates to the

discovery of inferences that hold over a particular temporal interval. Temporal predicate analysis, in a similar manner to spatial mining, refers to the discovery of temporarily constrained inferences. For example, the inference $a \overset{before}{\Rightarrow} b$. Sequence mining refers to the discovery of inferences within a temporally ordered sequence of objects.

Temporally qualified analysis encompasses inferences which have a temporal existence or pertain to a temporal period. For example, "newspapers and milk are purchased in the morning", is indicative of a temporally qualified inference where *morning* represents a temporal concept. Hence temporarily qualified inferences can lead to the discovery of inferences that although having low global validity within the dataset, have a high validity when constrained to a particular temporal subset of objects. In the literature, specialisations of temporally qualified mining exist and are known as interval, cyclic and calendric mining, and although they result in temporally qualified inferences, different temporal schemas are applied.

Temporally qualified analysis attempts to discover valid elementset and temporal pattern pairs, through proposed extensions to Apriori (Li, Ning, Wang & Jajodia 2001, Ramaswamy, Mahajan & Silberschatz 1998) and Partition (Ale & Rossi 2000). The inclusion of temporal hierarchies within association mining is discussed in Bettini *et al.* (1998) to discover temporal inferences of differing granularities, while Ale & Rossi (2000) introduce the concept of an element's *lifetime* (or existence), which enables the exclusion of elementsets from consideration during times at which they did not exist within the dataset.

Temporal predicate inferences differ to temporally qualified analysis as the temporal quality is represented within the inference rather than through a temporal qualification of the valid elementset. Instead of finding that an inference has an existence (or a cyclic nature) temporal predicate inferences indicate a temporal relationship within a valid elementset. For example, $\textit{Policy-A} \overset{meets}{\Rightarrow} \textit{Policy-B}$ 82%, indicates that 82% of clients that closed their *Policy-A* did so as part of the change to *Policy-B*. Some critical work in this field was undertaken by Rainsford & Roddick (Rainsford & Roddick 1999) through the development of a post-analysis process that incorporated the temporal relationships identified by Allen (1993).

owner	sequence
a	(3) (6,9)
b	(1,6) (4,7,9) (5,6)
c	(4,9) (3,6) (6,9)
d	(2,7) (4) (2,5,8,9)

Figure 1.19: Sequential mining example dataset, Agrawal & Srikant (1995)

The process generates all pairings of elements within an inference that have a temporal component. The dataset (D) is subsequently scanned, calculating the validity for the each pairing in reference to the temporal relationships defined by Allen (1983). All valid temporal relationships are then appended to the original inference as illustrated in Rule 4.

Rule 4 $Policy\_A \Rightarrow Policy\_B, Policy\_C$ (79%),
$(Policy\_A \overset{during}{\rightarrow} Policy\_C$ (74%)),
$(Policy\_A \overset{meets}{\rightarrow} Policy\_B$ (82%))

Sequence mining concerns the discovery of valid elementsets across ordered sequences of elements, within which the ordering has temporal implications. A sequence is a temporally ordered set of objects and hence sequence mining involves the discovery of ordered elementsets between objects within sequences. In this context each object $o \in O$ has an associated time-stamp and owner, and a sequence is the time-stamped ordered set of O for a particular owner, and a sequential pattern is valid if it is contained in at least $minsup$ sequences. For example, given the sequential mining dataset in Figure 1.19(from Agrawal & Srikant (1995)) and a $minsup = 50\%$, V contains those ordered elementsets that exist in two or more sequences, such as [(4)(6)] and [(3)(6,9)].

Seminal sequential mining research (Agrawal & Srikant 1995) led to the development of a set of algorithms based upon Apriori. AprioriAll discovers all sequential patterns within D, while AprioriSome and DynamicSome discover the maximal sequential patterns, in a like manner to MFS, with DynamicSome generating the candidate elementsets on the fly. This work was later extended, through

the development of GSP, which incorporated hierarchies and non-contiguous elementsets (Srikant & Agrawal 1996).

Alternatively, SPADE and PrefixSpan by Zaki (2002) and Pei *et al.* (2001a) respectively, use pattern growth variations to perform sequential mining. SPADE is based upon equivalence classes, in a similar fashion to Eclat and Charm. PrefixSpan is based upon concepts developed in FP-Tree, however since ordering is significant, the reordering and prefix sharing undertaken by FP-Tree cannot be incorporated. This results in excessively large structures. To overcome this PrefixSpan recursively partitions (through projection) the dataset and the relevant patterns to a smaller environment, resulting in the creation of smaller structures.

Episode mining is a generalised form of sequential mining, that discovers valid patterns within simple sequences of events. Mannila *et al.* (1997) have developed a number of episodic mining algorithms based upon candidate generation techniques and incorporating *support* as a quality heuristic. A full syurvey of temporal data mining is given by Roddick & Spiliopoulou (2003).

Privacy

A recent inclusion to semantic incorporation has been privacy semantics, the goal of which is to develop accurate models without access to precise data. The difficulty lies in the fact that the two metrics of accuracy and privacy are contradictory, and consequently improving one generally incurs a cost to the other. Privacy mining attempts to find optimal solutions to this dilemma. There are two main approaches evident in the literature, data distortion and secure multi-party computation. Data distortion involves the development of pre-process techniques that distort the exact data while still allowing the elicitation of *true* valid elementsets. Secure multi-party computation involves the distributed computation of inferences without sharing information between sources (Vaidya & Clifton 2002).

A common data distortion technique, (Warner 1965), is based upon randomised response and uses a probability heuristic p to dictate the replacement of an element with other elements not in that object. Where if p is significantly large, most elements will be randomised, obscuring element ownership and improving privacy. However *true* valid elementsets will still be visible, though to a lesser

extent, in large datasets with high variance, where $|E|$ is large. For example, from (Evfimievski, Srikant, Agrawal & Gehrke 2002), given $p = 80\%$ and an elementset of length 3 that occurs in 1% of objects ($\sigma(i) = 1\%$), then after randomisation it will appear in about $1\% \cdot (0.2)^3$, or 0.008% of objects. The opposite chance of *false* elementsets becoming more frequent is determined by $|E|$, where as $|E|$ increases the chance of *false* positives diminishes rapidly. Although generally effective, uniform randomisation has been shown to allow privacy breaches in certain situations. This issue is addressed by Evfimievski *et al.* with the development of more advanced randomisation operators that reduce the possibility of security breaches.

Saygin *et al.* (2002), present an alternative technique in which instead of distorting the data through the use of false values, the concept of unknown values (or uncertainty) is introduced. Therefore the regular binary matrix of 1's and 0's is extended to a ternary matrix incorporating ?'s, in effect hiding the presence of sensitive elements within objects. The inclusion of uncertainty ensures privacy, however it subsequently results in uncertainty within validation metrics. This is addressed within Saygin *et al.* (2002) by modifying the algorithm to provide heuristic intervals, within which the actual value lies. For further information regarding the state-of-art in privacy preserving data mining, see Verykios *et al.* (2004).

1.3.4 Incremental mining

Incremental mining is concerned with the maintenance of an inference set in an evolving dataset, a dataset that is updated with new data. Hence if the inferences are calculated for the initial dataset D, then after the arrival of new data (δ) some of the inferences initially calculated may no longer be valid, due to a change in their degree of presence. The naive approach to compute the updated set of inferences is to re-mine the entire dataset $D + \delta$ using a classic algorithm, however, this is inefficient due to the re-analysis of existing data. The aim of incremental mining algorithms is to minimise the expense of updating V by using previously mined information.

Cheung *et al.* (1996) proposed the first incremental association mining algorithm *Fast UPdate* (FUP) that extends *Apriori* with a set of incremental inclusion rules, presented in the subsequent list, to incorporate incremental functionality. Therefore, given an initial valid elementset V based upon D and an update or increment dataset δ such that $\delta \ll D$, *FUP* finds all valid elementsets in $D \cup \delta$, denoted $\bar{V}$. This research was subsequently extended in *FUP2* (Cheung, Lee & Kao 1997) to handle increment removal and to use statistical sampling to decide when to incorporate the increment datasets. Like *Apriori*, both FUP and FUP2 require κ scans of D, however subsequent algorithms have been able to reduce this to at most a single scan.

- Elementset $V_\kappa^i \in D$ is eliminated if and only if $\sigma(V_\kappa^i) \in (D \cup \delta) < minsup$.
- Elementset $A \notin V$ can only be included in $\bar{V}$ if $\sigma(A) \in \delta \geq minsup$

Other incremental extensions have been proposed to the classic association mining algorithms, Partition (Savasere et al. 1995), FP-Growth (Han & Pe 2000) and also to the notion of negative borders introduced by Toivonen (1996). An incremental Partition extension is proposed by Omiecinski & Savasere (1998) and a fundamentally equivalent foundation is used as the basis for the sliding window filtering algorithm (SWF) proposed by Lee, Lin & Chen (2001) and its proposed extension *FI-SWF*, (Chang & Yang 2003). Two incremental extensions to FP-Growth have been proposed, both of which avoid rebuilding the FP-Tree structure when relative frequencies change. This is achieved through the development of a modified tree structure, *CATS-Tree*, (Cheung & Zaiane 2003) and the development of an associated *Pattern Repository* structure from which FP-Tree can be derived (Relue, Wu & Huang 2001).

Thomas *et al.* (1997) propose the *ULI* algorithm that uses the principle of *negative borders*, proposed by Toivonen (1996), to facilitate incremental update. Where the negative border is the set of candidate elementsets C that were not included within V, e.g. $C|\sigma(C_i) < minsup$. Through maintenance of the negative border ULI (which requires κ scans of δ) only requires a single scan of D if

the negative border changes. Ganti *et al.* (2000) extend ULI by using Tidlists and introducing data stream monitoring to determine when to update. They also present a generic model *GEMM* that provides a general framework through which incremental algorithms can be extended to incorporate *windowing*. Where windowing functionality constrains the active dataset, or the dataset from which the current inferences are derived, to be the most ω blocks of data. The use of windowing indicates that data, within specific domains, has a time span of relevancy and is removed after a specified period.

Ayan *et al.* (1999) present the Update with Early Pruning (*UWEP*) algorithm that based upon Apriori scans D at most once and the new dataset $\bar{D} \mid \bar{D} = D \cup \delta$ exactly once. UWEP incorporates Tidlists and look-ahead strategies to improve pruning, however it only promotes candidate elementsets if they are valid in both D and δ, resulting in the possible removal of valid elementsets.

Recently, incremental extensions (Veloso, Possas, Meira Jr & de Carvalho 2001, Veloso, Meira Jr, de Carvalho, Possas, Parthasarathy & Zaki 2002) have been proposed for MFS algorithms (Section 1.3.2). *Pelican*, (Veloso et al. 2001), extends MaxEclat through the common use of prefix-based equivalence relations. Subsequent work by the same team proposed *Zigzag*, (Veloso et al. 2002), that extends GenMax in using naive backtracking for searching and also introduces two novel quality heuristics. *Zigzag* was further extended in *Wave* (Veloso et al. 2002) to incorporate estimation techniques and trend analysis to efficiently maintain an approximate data model, although further improvements to the processing time comes at the expense of accuracy.

Although not based upon the principle of MFS, *Maap* (Zhou & Ezeife 2001) is similar in its approximation of *small* valid elementsets. The algorithm uses an Apriori based framework, whereby given the high-level valid elementsets from D [2], V_D, it is able to efficiently compute the equivalent high-level valid elementsets V_δ, and also infer some of the lower-level V_δ.

[2]Refer to Zhou & Ezeife (2001) for detailed discussion of high and low level elementsets.

Table 1.3: Summary of Semantic & Incremental Algorithms.

Name	Author	Year	Base	Contribution
Hierarchical Algorithms				
ML-T*	Han & Fu	1995	Apriori	Element replacement
Basic	Srikant & Agrawal	1997	Apriori	Transaction extension
Cumulate	Srikant & Agrawal	1997	Basic	Redundancy filtering
Stratify	Srikant & Agrawal	1997	Basic	Pruning Optimisation
Estimate	Srikant & Agrawal	1997	Apriori	Sampling
Estmerge	Srikant &Agrawal	1997	Apriori	Sampling
Prutax	Hipp et al.	1998	AprioriTID	-
Adaptive-FP	Mao	2001	FP-Growth	-
FP-Tree	Ong et al.	2001	FP-Growth	Recurrency
HND	Ceglar et al.	2003	Apriori	Non-monotonic support & dynamic rule gen.
HPTid	Ceglar et al.	2003	AprioriTID	Prioritisation
Privacy Algorithms				
-	Vaidya &Clifton	2002	-	Secure multi-party computation
-	Evfimievski et al.	2002	-	Data Distortion
-	Saygin et al.	2002	-	Uncertainty
Spatial Algorithms				
-	Koperski & Han	1995	Apriori	Spatial

Table 1.3: Summary of Semantic & Incremental Algorithms (continued).

Name	Author	Year	Base	Contribution
Sequence & Episodic Algorithms				
AprioriAll	Agrawal & Srikant	1995	Apriori	-
AprioriSome	Agrawal & Srikant	1995	AprioriAll	Maximal sequential patterns
DynamicSome	Agrawal & Srikant	1995	AprioriSome	Dynamic candidate generation
GSP	Srikant &Agrawal	1996	DynamicSome	Hierarchies & Non-contiguous elementsets
-	Mannila et al	1997	Apriori	-
SPADE	Zaki	1998	ECLAT	-
PrefixSpan	Pei et al.	2001	FP-Growth	-
Temporal Algorithms				
-	Rainsford & Roddick	1999	-	Post mining process
-	Ale & Rossi	2000	Apriori	Element lifetime
Temporal-Apriori	Li et al.	2001	Apriori	-
Incremental Algorithms				
FUP	Cheung et al.	1996	Apriori	Deletion & Sampling
ULI	Thomas et al.	1997	-	Negative borders
Borders	Gant et al.	2000	ULI	Tidlists
UWEP	Ayan et al.	1999	Apriori	Look ahead strategies
ZIGZAG	Veloso et al.	2002	GenMax	Incremental MFS
WAVE	Veloso et al.	2002	ZIGZAG	Estimation

1.4 Inference Generation

Inferences are derived (or generated) from the set of identified valid elementsets. For each valid elementset (v) a set of inferences R are produced, derived from partitions based upon the permutations of v, where an inference $r \in R$ is of the form $A \Rightarrow B \mid A \subset v \wedge A \neq \emptyset \wedge B = (v - B)$, where A is the set of antecedents and B is the set of consequents. All inferences have an associated quality heuristic confidence (γ) that is derived from the inference's strength, $\gamma(R) = \frac{\sigma(v)}{\sigma(A)}$. The inferences that meet a specified confidence threshold (θ), or strength, are appended to the result set.

Agrawal and Srikant (1994) propose an optimisation to this technique that can potentially reduce the number of elementset permutations considered. This is based upon the observation that given the derivation from v of an inference $A \Rightarrow B$ that does not meet the confidence threshold θ, then all subsets of A need not be considered for generating inferences using v. For example, if $\gamma(ab \Rightarrow cd) < \theta$ then $\gamma(a \Rightarrow bcd) < \theta$ and $\gamma(b \Rightarrow acd) < \theta$, as $\sigma(A)$ is the denominator in the calculation of γ and $\sigma(a) \wedge \sigma(b) \geq \sigma(ab)$.

In order to effectively incorporate this pruning technique the inferences R for v must be produced in order of descending antecedent length, so that if $\gamma(A \Rightarrow B) < \theta$ then all antecedents $\bar{A} \mid \bar{A} \subset A$ will not be considered. The process is more intuitive if considered from the other direction, that of generating the inferences in order of increasing consequent length, due to its non-monotonicity. For example, given the derivation of a valid inference $\gamma(A \Rightarrow B) \geq \theta$ from v, then all other inferences derived from v of form $\bar{A} \Rightarrow \bar{B}$ must be valid where $\bar{B} \subset B$. This is true, as for the same case given that $\bar{B} \subset B$, then $\bar{A} \supset A$. Inference generation then becomes an iterative process incorporating a non-monotonic constraint (in a similar manner to Apriori) where the inferences of consequent length κ are based upon the valid inferences of consequent length $\kappa - 1$ discovered during the previous iteration

1.4.1 Rule inferencing

The number of inferences generated in any mining run is often too many to be effectively presented and interpreted by the user.Three current general methods

exist to facilitate user interpretation: generalisation (Section 1.3.3), visualisation (see Chapter 3) and rule inferencing. Rule inferencing constrains inference generation to non-redundant inferences, where an inference is redundant if it conveys the same or less general information than another inference of the same usefulness and relevance (Pasquier, Bastide, Taouil & Lakhal 1999*a*). Hence inference $r = (A \Rightarrow B)$ is non-redundant if there is no other inference $\bar{r} = (\bar{A} \Rightarrow \bar{B})$ with the same validity (level of *support*) such that $A \subset \bar{A}$ and $B \supset \bar{B}$. This minimal set of non-redundant inferences has been denoted the inference *basis*.

Initial research by Toivonen *et al.* (1995) developed a simple inferencing technique in which an inference r is removed if another inference $\bar{r}$ exists, where $\bar{B} = B$, $\bar{A} \subset A$ and the relevant elementsets Tidlists are equal, $Tidlist(r) = Tidlist(\bar{r})$. An inference basis clustering technique is also proposed, where distance between bases is based upon the difference between their associated tidsets, further facilitating user interpretation. This initial pruning technique was subsequently incorporated into Shah *et al.* (1999) rule pruning list (rule 1), presented below.

---

1. If two inferences $A \Rightarrow C$ and $A \wedge B \Rightarrow C$ of similar strength exist then $A \wedge B \Rightarrow C$ is redundant.
2. If two inferences $A \Rightarrow C$ and $B \Rightarrow C$ with similar strength exist, $B \Rightarrow C$ is redundant if $A \Rightarrow B$ is valid and $B \Rightarrow A$ is not.
3. If two inferences $A \Rightarrow C$ and $B \Rightarrow C$ with similar strength exist they are considered *weak* inferences if both $A \Rightarrow B$ and $B \Rightarrow A$ are valid. Where a weak inference is one which is valid in terms of the quality heuristic however due to the possible presence of alternative causes, their validity may be questionable.
4. If two inferences $A \Rightarrow C$ and $A \Rightarrow B \wedge C$ exist, then $A \Rightarrow C$ is redundant.
5. If two inferences $A \Rightarrow B$ and $A \Rightarrow C$ exist, and further $B \Rightarrow C$ exists, then $A \Rightarrow C$ is redundant.

Following their previous work both Pasquier *et al.* (1999a) and Zaki (2000a) proposed similar rule inferencing techniques based upon the theory of closed elementsets (see Section 1.3.1), which they show to be a generating set for all inferences holding within D. Based upon this, *basis* techniques are defined for both exact (inferences with 100% confidence) and inexact inferences.

1.5 Summary

This chapter provides a comprehensive review of significant algorithmic contributions within the field of association analysis. It shows (through chronological presentation) the field's evolution to it's current level of maturity, in which research emphasis has moved from performance optimisation to domain specific specialisations. Although comprehensive, the review could never be complete due to the size of the current body of research, instead it focuses upon providing coverage of the different fundamentals used within association mining. For new algorithms, the yearly ICDM workshop on Frequent Itemset Mining Implementations (FIMI) provides an excellent source (Goethals & Zaki 2003).

This review serves as a foundation upon which further research into the hypothesis of guided association mining is based. As a result of research undertaken in the construction of this survey, the novel concept of incremental (maintained) closed-set association mining (*MCL*) was elicited. This approach shows promise in both process optimisation and facilitating user interpretation of results and is presented in the next chapter (Chapter 2).

Chapter 2

MCL: Maintained Closed-Lattice Association Analysis

Incremental association mining research concerns the maintenance of the set of valid elementsets (V) in an evolving dataset. Given that V_{D^i} is the set of valid elementsets generated from the evolving dataset D^i, where i signifies an evolutionary step of D, then the inclusion of an increment dataset δ such that $|\delta| \ll |D^i|$) will effect the degree of element presence in the combined dataset $D^1 = D^0 + \delta$. The naive computation of V_{D^i} involves re-mining D^i using a classic association mining algorithm (such as Apriori) however, as a significant part of the knowledge produced by the mining of D^i, is already available in $V_{D^{i-1}}$, this results in process replication. Hence incremental mining attempts to facilitate the inclusion of δ into V_{D^i} by using currently available information in $V_{D^{i-1}}$ and D^{i-1}. This evolution results in the alteration of participant states (summarised in Table 2.1), where participants are the elements in D and the elementsets in V.

This chapter presents a novel technique that advances the state of incremental association mining. Maintained Closed-Lattice Association Analysis (*MCL*) like previous approaches, uses currently available information. Unlike previous algorithms however, it uses the concept of the *closure of the Gaulois connection* (Ganter & Wille 1999), or closed-sets, basing the incremental mining upon a condensed representative lattice from which V can be inferred (Section 1.3.1).

Such an approach shows promise in both process optimisation and facilitating user interpretation of results. The processing is optimised (especially in highly

Shape	Description
Static	No relative change in participant presence.
Strengthened	The presence of the participant increases.
Weakened	The presence of the participant decreases.
Emergent	An invalid participant becomes valid.
Declined	A valid participant becomes invalid.

Table 2.1: Alteration of Participant state

correlated or dense datasets) as increment datasets are applied to a smaller maintained lattice, reducing the search space. The maintenance of a closed-set lattice also aids user interpretation due to its reduced size. The other significant contribution of this algorithm is the production of an increment lattice I during the mining process. This provides the user with insight into the increment's effect upon the maintained lattice L. Furthermore, it provides an effective means of incorporating *windowing* functionality (Cheung et al. 1997), or the removal of previous increments, without the need for further mining.

This chapter presents the theory behind MCL, its implementation and the results of some some preliminary testing. The foundations for this presentation of MCL lie in reviews of closed-set and incremental association mining, presented in the previous chapter (see Sections 1.3.1 and 1.3.4). The following section discusses MCL in detail with subsequent sections presenting some preliminary test results and a conclusion.

2.1 MCL Framework

MCL provides a novel and efficient incremental association mining method by maintaining a closed-set lattice (L) from which the set of valid elementsets (V) can be easily derived. The maintenance of L^i uses an increment closed-set lattice I that is derived from the increment dataset δ in the presence of D^{i-1} and L^{i-1}. I is then appended or removed from L^{i-1} resulting in L^i, an evolution of the maintained closed-set lattice. Assuming that an increment dataset must be appended to the maintained lattice before it can be removed, the removal process

Object	*Elements*
1	a,c,t,w
2	c,d,w
3	a,c,t,w
4	a,c,d,w
5	a,c,d,t,w
6	c,d,t

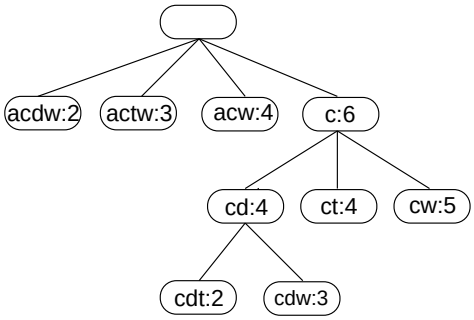


Figure 2.1: Example dataset & resulting closed set lattice derived from (Zaki & Hsaio 2002)

can be optimised by using the previously derived I, alleviating the need to remine δ. Furthermore the generation of I during the append process provides the user with an effective insight to δ's effect upon L.

The algorithm assumes an initial L^0 and D^0. The internal representation of the increment dataset δ, referred to as d (see section 2.2.1), and D^i are vertically organised Tidlists and the subsumption table, used as part of closed-set validation takes the tuple form $\{Tidlist, Array[L_i]\}$. The lattice structures L and I are prefix trees, of node form $\{elementset, Tidlist\}$ that use lexicographic ordering to maintain consistency during evolution. The output from both the append and remove process is an updated L^i and D^i. The append process also produces I and d to facilitate user interpretation and subsequent increment removal. These processes are presented in the following sections.

2.2 Append Function

The following discussion of the append process is logically divided, based upon the fundamental processing steps into three sections: *generate*, *merge* and *strip*. *Generate* prepares the required data structures including the increment lattice, I, while *merge* and *strip* perform the actual update, focusing upon the update of existent and emergent closed-sets respectively. Figure 2.1 presents L^0 and D^0, from Zaki & Hsiao (2002) , which is used as the working example.

2.2.1 Generate

This stage prepares for the subsequent merging of L and I through the preparation of data structures and the subsequent generation of I. Data structure preparation requires a scan of δ which is used to update $D \,\forall\, e \mid e \in \delta \,\wedge\, e \in E$, generating D^1 from D^0. During this traversal the data structures d and ext are also populated. Given $\sigma(D_e^0) > minsup$ and $\sigma(D_e^1) > minsup$ then $e \in \delta$ is appended to d, resulting in a listing of δ elements that exist in L and the increment objects within which they participate. If $\sigma(D_e^0) < minsup$ and $\sigma(D_e^1) > minsup$ then e is emergent and D_e^1 is appended to ext, resulting in a listing of emergent elements in D^1 that may extend L^0.

Subsequently, closed-set mining based upon the closure principles identified by Zaki and Hsiao (Section 1.3.1) is applied to d and ext resulting in I and X respectively. While the mining of ext incorporates quality heuristics (eg support), this does not apply to the mining of d, as all elementsets founded upon current valid elements (those existing within L^0) must be reported in I to accurately update all elementsets in L. Once constructed, I and X (presented in Figure 2.2 with δ) contain all the information required to accurately update L. A summarisation of this process is presented in Algorithm 2.1.

Algorithm 2.1 Generate

```
1: for all d_e ∈ d do
2:    D_e^1 = D_e^0 + d_e
3:    if σ(D_e^1) > minsup then
4:       di.append(d_e)
5:       if σ(D_e^0) < minsup then
6:          ext.append(d_e)
7:       end if
8:    end if
9: end for
10: I = closeMine(di, 1)
11: X = closeMine(ext, minsup)
```

Object	*Elements*
1	a,c,d,t,z,x
2	a,t,x
3	c,d,z,x

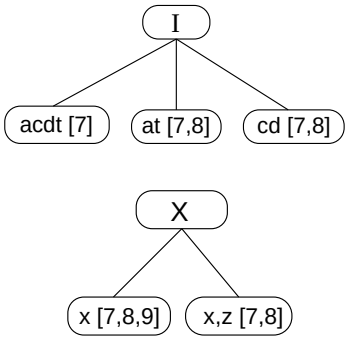


Figure 2.2: Example increment dataset and derived data-structures

2.2.2 Merge

The incorporation of I and X within L requires a scan of L for which all pertinent $x \in X$ and $i \in I$ are applied to $l \in L$. Given that $|\delta| \ll |D^i|$ and the set of elements $e \in E$ is common to both D and δ, relatively few new elementsets are appended to L during any given increment. The result is that an increment's effect upon L will often be internal to the lattice, as most valid elementsets are already represented. However, although less common, lattice boundary extension does occur through emergent elementsets and superset extension, therefore each l must be tested against X for emergent supersets and against E to update its Tidlist and to discover any emergent subsets. This process is optimised by considering only relevant X and E for each l by dynamically altering these structures as processing occurs.

Each l is then checked against an evolving X and I during a pre-order traversal of L^0, resulting in the construction of a new closed lattice L^1, which eliminates updating complexity apparent when using a single lattice. The emergent set X applied to each l contains the emergent elementsets valid for the parent of l, denoted $X_{\bar{l}}$, therefore $X_{\Re} = X$.

For example, given $X = \{x, z\}$ and $X_a = \{ax\}$, then X_a is the extension set passed to all children of a, quickly reducing X_l as the traversal deepens. Hence each l is merged with $X_{\bar{l}}$ and if valid and not subsumed (through closed set pruning), it is appended to L^1, X_l and R (discussed in Section 2.2.3).

The increment lattice is pruned during processing (see section 2.2.3) so that the pertinent $i \in I$ for the current l are the first nodes encountered using a pre-order traversal. Thus I is traversed until i lexicographically exceeds l at which point no subsequent i are pertinent to l.

(a) Partial L (b) Partial I

Figure 2.3: Partial lattice structures: L and I

The inclusion of further search space reduction depends upon the effect of I and X upon l and the relationship between i and l. For each pertinent i, if $i \supseteq l$ then L is updated with $i.Tidlist$ and i's descendants are removed from the search space as the Tidlists, all supersets of i, are subsets of $i.Tidlist$ and cannot further affect l. If $i \subset l$ then descend to i's children, with no action taken upon l. If i is neither a superset, a subset nor equal to l, then its subtree is not processed. For example, given the partial lattices in Figure 2.3, when $l = \{ad\}$ it is first compared to $i = \{a\}$ (a subset) therefore no update occurs and i descends to $i = \{adt\}$ a superset, l is updated, but i's subtree is not processed. Processing then moves to $i = \{adt\}$, also a superset, therefore once again l is updated. Assuming that the next i exceeds l, since l was updated, its descendants are processed, namely $\{adw\}$. As $i = \{a\}$ is a subset i moves to $i = \{adt\}$, according to this algorithm an unrelated set, therefore its descendants are ignored and i moves to $\{adw\}$ which is equivalent to the current l which is therefore updated.

The incorporation of these search constraints optimises the update (strengthening and weakening) of existing $l \in L^0$. If there is no change to l through this processing, then no superset of l will be modified by X or I due to the inclusion of the non-monotonic heuristic, *support*. Hence the subtree of l is appended to L^1, without further comparison against I or X. However before any l is appended to L^1 a final check against *minsup* is made to identify declined elementsets, and if

declined, l and its descendants are eliminated from consideration. Algorithm 2.2 presents the *merge* process, including the *update* method that details the traversal of pertinent I for l.

2.2.3 Strip

Strip is instigated from *merge* when a level-1 node is encountered during L^0's traversal. It provides two functions, the reduction (or stripping) of I to eliminate unwanted elements and the discovery of all remaining emergent elementsets. Before instigating the reduction process a copy of I is made, denoted R, to report the increments effect upon L^0 and to facilitate the removal process. Any emergent elementsets that are subsequently appended to L^1 are also inserted into R.

Due to lexicographic ordering, once all $l \in L^0$ beginning with a particular element (e) have been merged with I and X, all subsequent l exceed e and it therefore takes no further part in the update of L^0. Therefore e can be stripped from all $i \in I$, allowing the progressive reduction of I.

Strip occurs before the *merge* of level-1 L^0 nodes, removing all elements from I that lexicographically precede the first element of the current l. Once the preceding elements have been removed, the modified elementsets (if not subsumed) are re-inserted into I. For example, given a level-1 $l = \{d\}$ and $I = \{\{a, c\}, \{c, d\}, \{d, w\}\}$, then after stripping $I = \{\{d\}, \{d, w\}\}$. This dynamically reduces I and facilitates the update of L^0 by ensuring that the relevant increment closed-sets are the first encountered during *merge*.

The discovery of emergent closed-sets is undertaken in conjunction with stripping due to the focus upon the relevant structures. All candidate emergent set information is represented within I and X, and by removing their discovery from the traversal of L significant process duplication is avoided. The emergent closed-sets comprised of valid elements in D^0 are represented in I, while emergent elements are represented in X.

The candidate emergent sets in which the stripped elements (S) participate are identified by deriving the set of elementsets from the closed-sets in I in which an

Algorithm 2.2 Merge

merge(l,X)
1: **for all** $l' \mid l' \in l$.child **do**
2: **if** $l_{parent} == L_{\Re}$ **then** strip(l')
3: modified = l'.update(l',I)
4: **if** $\sigma(l')$> minsup **then**
5: **if** modified **then**
6: **if** !subsumed(l') **then**
7: L^1.append(l')
8: X' = extCheck(l',X)
9: **end if**
10: merge(l', X')
11: **else**
12: L^1.appendTree(l',minsup)
13: **end if**
14: **end if**
15: **end for**

update(l,i)
1: **for all** $i' \mid i' \in i$.child **do**
2: **if** i'.exceeds(l) **then** break
3: mod = false
4: **if** $i' \supseteq l$ **then**
5: l.updateTid(i')
6: mod = true
7: **else**
8: **if** $i' \subset l$ **then** mod=update(l,i')
9: **end if**
10: **end for**
11: **return** mod

$s \in S$ participates. For example, given $S = \{a\}$ and $I = \{\{a, c\}, \{a, w, x\}, \{c, d\},$-$\{d, w\}\}$, then the candidate emergent elementsets, C, are $\{\{a, c\}, \{a, w\}, \{a, x\},$-$\{a, w, x\}\}$. The potential candidate emergent elementsets p are generated through a preorder traversal of I, incorporating DCP. If $p \notin C$, its support in D^1 is discovered and is appended to C, irrespective of whether $\sigma(p) > minsup$. The representation of all p in C irrespective of support optimises processing as duplicate p can be quickly removed without calculating support.

Algorithm 2.3 Strip

strip(l)
1: **for all** $i \mid i \in I \wedge i_1 . < l_1$ **do**
2: C = genCandidateEmerg(i,l)
3: i.removeFromParent()
4: i.removePreceding(l)
5: **if** !subsumed(i) **then**
6: I.insert(i)
7: **end if**
8: generateEmergent(C)
9: **end for**

generateEmergent(C)
1: **for all** $c \mid c \in C$ **do**
2: **if** $\sigma(c) > minsup$ & !subsumed(c) **then**
3: L^1.append(c)
4: R.append(c)
5: extCheck(l',X)
6: **end if**
7: **end for**

The resulting C is then appended to L^1 and R where c is valid and not subsumed. Furthermore, if appended to L^1, c is then checked against the emergent elements X in a similar manner to l (see Section 2.2.2) to possibly generate further emergent supersets. The identification of the emergent closed-sets, although algorithmically complex, consumes relatively little processor time due to the small number of candidate sets generated and the pruning techniques incorporated.

After L^0 has been updated through the merge process the remaining emergent closed-sets are discovered by applying the same process to what remains of I. Finally the emergent elements, $x \in X$, are appended to L^1 and R if they are not subsumed. This process is summarised in Algorithm 2.3 and the resulting L and I are presented in Figure 2.4

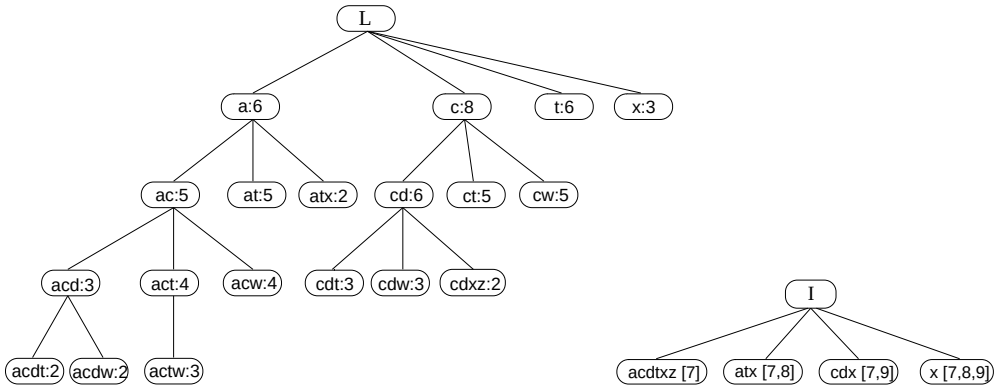


Figure 2.4: Complete lattice structures: L and I

2.3 Removal

The creation of the increment lattice (I) facilitates the removal of previous increments from L by alleviating the need to re-mine the increment dataset. This significantly reduces the removal process by reusing information. The maintained dataset D is first updated by removing the increment dataset (δ) from it. The subsequent updating of the maintained closed-set lattice (L) is similar to the merge stage of append, in that it is based upon the traversal of L and uses consistent ordering to reduce the search space.

For each $l \in L$, I is traversed until i lexicographically exceeds l or $i \supset l$. If $i \supset l$ then $i_{Tidlist}$ is removed from $l_{Tidlist}$, which may result in the subsequent declining and removal of l and its supersets from L. If l remains valid, its presence within the subsumption table is altered to reflect its new Tidlist and if l subsumes its parent, the parent is replaced by l in L.

2.4 Experimental Results

Some experimental results are presented in Figure 2.5, which compares processing time and lattice size between a naive algorithm and *MCL* as dataset density increases. These graphs are both from the same result set and are based upon artificial datasets, $|D^0| = 5K$ and $|\delta| = 0.5K$, in which density is manipulated by adjusting the number of elements from which the objects are generated. Figure 2.5(a) illustrates the efficiency of MCL over the naive re-mining of D^1, using Apriori, and also presents the time taken to construct the initial lattice L^0 using Charm. Figure 2.5(b) illustrates, from the same result set, the relative lattice size reduction of the maintained and increment lattices, L and I, over the valid elementset lattice, as density increases.

The contributions of MCL to incremental association mining are twofold. First, through the use of closed-sets a condensed representative lattice is maintained that facilitates efficient update (especially for dense datasets) through reduced processing. Second, the creation of a closed-set increment lattice allows insight to the increment's effect upon the maintained lattice and reduces the processing required for subsequent increment removal.

The preliminary results support the theory of closed incremental mining and its contribution to data mining, especially within dense datasets environments. However, further testing against other incremental association mining algorithms, such as FUP2 (Cheung et al. 1997) and ULI (Thomas, Bodagala, Alsabti & Ranka 1997), is required to discover the extent of these contributions.

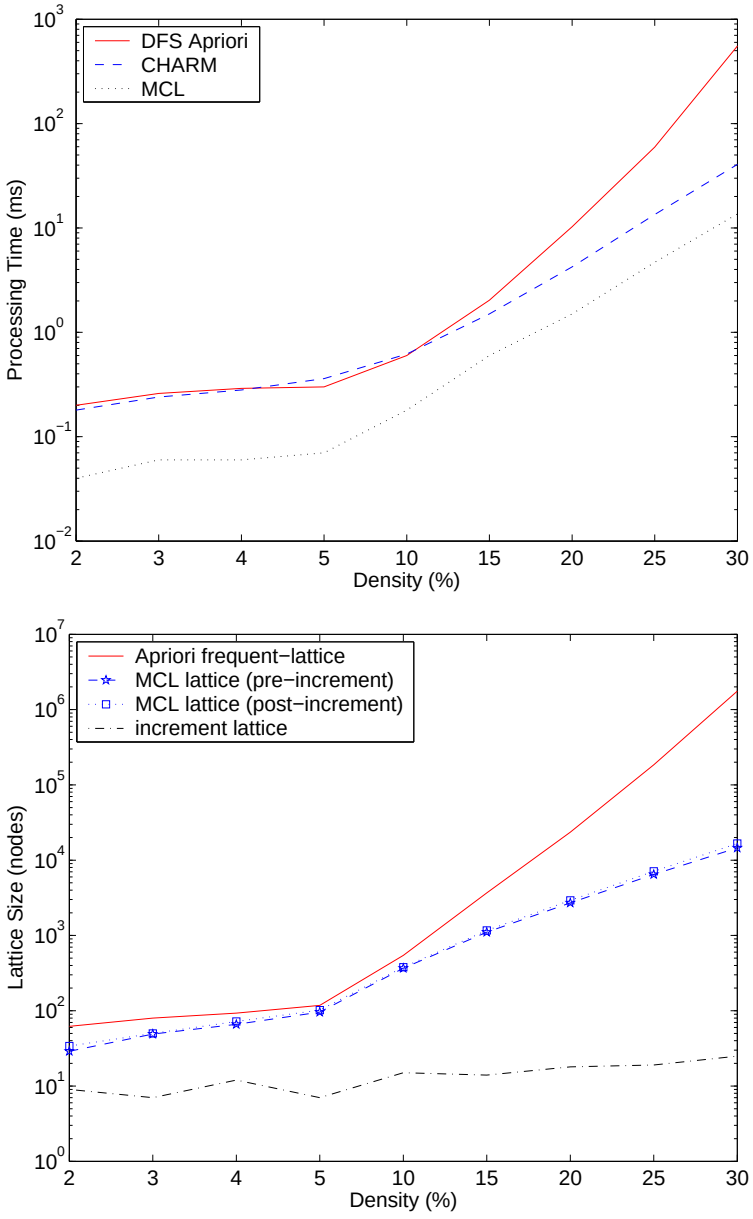


Figure 2.5: Experimental results (log_{10})

Part III

Rule Presentation

Thus I will again argue that visualization is necessary at all stages of the discovery process: at the front end for data awareness, understanding, massaging and audit; at the back end for presentation of results (either in confirmatory or presentation visualization); and in the middle stages for monitoring and understanding the computational elements, an area still under visualised

Georges Grinstein, 2002

Chapter 3

Rule Presentation Review

The user's acquisition of knowledge from a set of derived inferences, discovered during association mining analysis, requires an interface or means of communication between the computer and the user. This communication of information is at present restricted to visual and auditory perception. While there has been general research into the auditory presentation of data and the benefits of combining auditory with visual presentations (Barass, 1995), advances to date appear limited. This chapter focuses upon research into rule presentation techniques that facilitate user interpretation.

This chapter provides a review of presentation techniques used to facilitate the user's interpretation of association mining results. Section 3.1 discusses visual presentation in general, touching upon human perception and information theory. The next two sections (3.2 & 3.3) present a review of current association rule presentation techniques. Section 3.4 then presents a discussion of presentation interaction techniques in order to provide support for subsequent material that looks at incorporating guidance through interacting with rule presentation techniques.

3.1 Visual Presentation

There are two classes of visual presentation: textual and graphical. Textual presentation methods are simpler but more constrained. Graphical techniques are

more difficult to implement but show more promise, as they facilitate discovery through incorporating human perception in a less constrained manner.

Textual presentations are constrained to a set of well-defined primitives (characters, symbols and mathematical operators), which are interpreted by the user in a sequential manner at a fine-grained level of detail, with each primitive examined in turn. For example, reading is a sequential low-level interpretation of the symbols on a page. The benefit of this presentation style is that it is recognised and perceived in the same way by different users and is relatively quick and easy to produce. A drawback for textual presentations is that they are not conducive to the analysis of patterns, complex data or large data sets, all of which are key characteristics of useful data mining results. Table 3.1 shows the results of discovering inferences by performing association mining upon a small dataset. Textual presentation is satisfactory in this case as the result set is small and simple, and therefore the set of inferences can be easily interpreted through textual analysis. This form of analysis is impractical for typical association mining results, which are both large and complex.

Antecedent		Consequent	Confidence(%)
GingerBeer	$\Rightarrow$	Whisky	96
Whisky	$\Rightarrow$	GingerBeer	94
Lemonade	$\Rightarrow$	Whisky	95
Coffee	$\Rightarrow$	Tea	98
Cocoa	$\Rightarrow$	Coffee	95
Tea	$\Rightarrow$	Cocoa	93
Coffee	$\Rightarrow$	Tea, Cocoa	90

Table 3.1: Textual association presentation

Graphical methods or visualisations of mining results provide more powerful forms of presentation as they are not constrained to the pre-specified set of primitives, in the same way as textual presentation methods. Graphical presentations can take many different forms, as the underlying data can be mapped to many

different types of graphical primitives such as position, shape, colour, and size. Such a diversity leads to individual visualisations being able to present many dimensions of data in a concise manner by mapping data dimensions to differing graphical primitives. In contrast textual presentations of data dimensions are mapped to the same textual primitive type.

Human Perception and Information Theory (Miller 1956) indicates that graphical presentation facilitates the search for patterns by harnessing the capabilities of the human visual system to elicit information through visualisation, multi-dimensional perception, recoding, and relative judgement. Many experiments within the field of cognitive psychology have identified that regardless of sensory type (eg. sight, taste, and smell), humans can accurately perceive differences in the stimuli to a greater extent when many parameters of that stimuli are presented. For example, in experiments by Garner *et al.* (1956), participants were presented with a series of single dimension stimuli in the form of images, each showing a point at a different position on a line. Participants were asked to label each image either from a list of possibilities or with a number from 0 to 100 indicating where to the best of their judgement the point lay on the line. Results showed that on average humans could accurately perceive approximately 10 different placements. However in experiments where the visual stimulus was increased to two dimensions (Klemmer & Frick 1953) by the presentation of a point within a square, the level of perception rose to approximately 25 different placements. Multi-dimension perception results suggest that graphical presentations will greatly improve user perception. The relationship between dimensionality and perception has however been found to be asymptotic. Above ten or so dimensions, the addition of further dimensions does not improve perception (Miller 1956).

Recoding is the process of reorganising information into fewer chunks with more information within each chunk. This process is the means by which humans extend short-term memory (Miller 1956). The concept of recoding suggests that it is more difficult to perceive patterns within textual presentations because of the fine-grained sequential interpretation required. This is not conducive to pattern perception as the logical units remain small, resulting in the inability to understand the underlying structure of the result set. Visual presentations

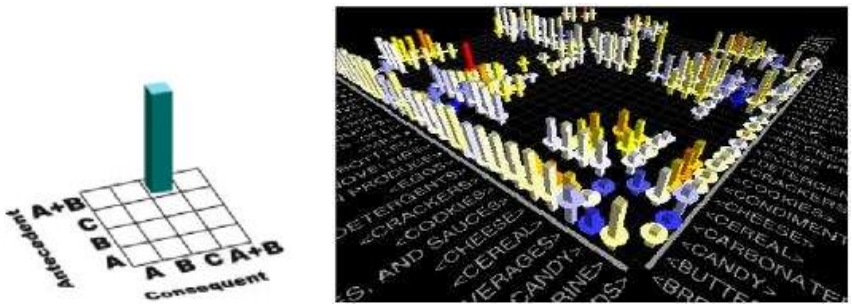
present a more contiguous representation of the data that can often be interpreted as a single logical unit, providing a conducive means by which the overall structure of the data set may be examined.

Weber's Law of Just Noticeable Differences (1834) states that *"The likelihood of detection [of a change] is proportional to the relative change, not the absolute change of a graphical attribute"* (Weber 1834). This law indicates that a user's perception will be superior when relative judgement instead of absolute measurement is made. For example, it is easier to perceive the change in a graphical object if its original form is displayed alongside the newly modified representation, because we can compare the difference or relative change between the two objects. It is more difficult to perceive changes when the original object is replaced by the new one because no comparison is available and reliance is instead placed upon the knowledge of the object's absolute measures.

Relative judgement is a graphical capability and is a major strength of graphical presentations as it allows users to obtain a holistic qualitative view of the result set, where relative differences between items can be recognised. A qualitative view is then used to focus attention, with subsequently more focused and quantitative analysis (absolute measurement) following. This process was dubbed the "Visual Information Seeking Mantra" by Shneiderman (1996) and is conducive to pattern discovery, allowing the user to analyse a picture at different levels.

Although more powerful and flexible than textual presentations, graphical presentations are more difficult to create and are open to subjective interpretation, whereas textual primitives have, in general, a more stable interpretation. Subjective interpretation is due to the abstraction of the underlying results into graphical primitives through defined mappings. This allows results to be presented in ways that facilitate perception of patterns and structures within the result set, however, if non-intuitive mappings are used then the perception of patterns will be less predictable.

The variance in subjective interpretations of a presentation can be reduced through good design. This includes ensuring that presentation styles reflect the data-mining task, and that the mapping between mining and graphical primi-



(a) 2D matrix (Wong, Whitney & Thomas 1999)

(b) SGI Mineset visualisation of association rules (D. Rathjens 1997)

Figure 3.1: Common matrix-based presentations

tives is intuitive, taking into consideration the user's objectives and facilitating interaction techniques. Clearly there is no single best presentation technique for a data-mining task as there are too many factors that depend upon both the user and the problem domain. The solution is therefore to create a flexible set of presentation formats for each mining task that can satisfactorily be applied to a wide range of problems.

To facilitate the interpretation of association mining results, various visualisation techniques have been created. Each of these methods requires two essential components: a graphical representation of the elements, and a technique that illustrates the relationships between these elements. The techniques fall into two classes: matrix-based and graph-based methods.

3.2 Matrix-based Visualisations

The basic design of a two dimensional matrix, maps the antecedent and consequent to the X and Y axes and, with inferences indicated by a graphical presence upon the intersecting cell (as illustrated in Figure 3.1(a)). This graphical presence can often encapsulate other information such as confidence through the use of primitives such as size and colour. Figure 3.1(b) from SGI's MINESET tool presents a complete three dimensional matrix based presentation (Rathjens 1997).

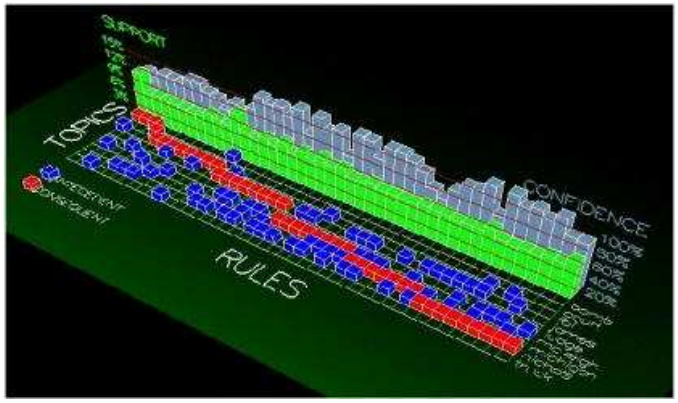


Figure 3.2: Rule vs item association matrix (Wong, Whitney & Thomas 1999)

This common presentation technique is useful when the inference set is comprised of a relatively small number of distinct antecedent and consequent values, so that given a particular presentation area (or real-estate) the entire inference set is viewable. Matrix-based techniques tend to quickly degenerate as the number of different antecedents and/or consequents in the inference set increases, resulting in the worst case, of an $O(n^2)$ rate of matrix area growth. This rapidly leads to large visualisations that are cumbersome, occluded and difficult to understand.

Wong *et al.* (1999) proposed a matrix presentation variation that attempts to minimise some of these matrix growth problems through the implementation of a rule-to-element matrix (shown in Figure 3.2) visualisation based upon the premise that an element can only occur once in an inference. This technique provides an improvement as matrix growth becomes linear, with respect to the number of inferences. Occlusion is also reduced by displaying associated quality heuristics, such as support and confidence as wall plates.

CrystalClear (Ong, Ong, Ng & Lim 2002) addresses the issue of scalability by mapping the quality heuristics of support and confidence to the axes instead antecedents and consequents. In this way the matrix size remains constant as it is not dependant upon the number of inferences but upon heuristic ranges, which may be grouped. Each matrix cell therefore relates to a group of inferences with common heuristics. Furthermore this approach allows for the easy identification of groups of heuristically similar rules that can then be displayed in a textual format for detailed analysis.

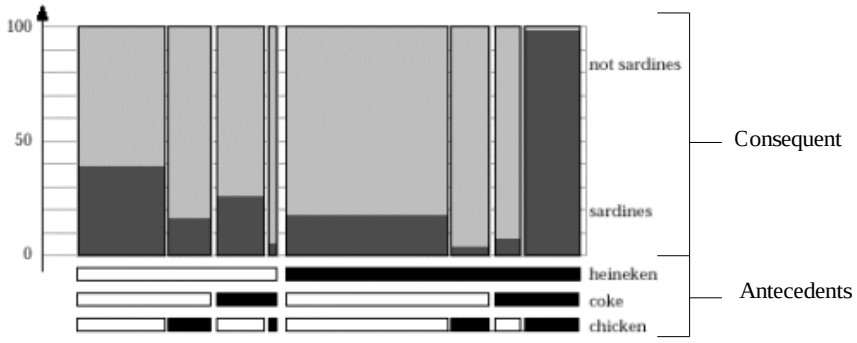


Figure 3.3: Interactive Mosaic Plot (Hofman, Siebes & Wilhelm 2000)

Hofman *et al.* (2000) created an alternative form of matrix visualisation - *Interactive Mosaic Plots* (Figure 3.3). This technique allows for the detailed investigation of inferences between a set of antecedents and a consequent. Within mosaic plots individual antecedents are represented as horizontal bars along the x-axis with the strength of an inference represented by the height of the vertical column above the specified antecedent permutation (inclusion denoted by black bar). Figure 3.3 illustrates the inferences between the antecedent set *heineken, coke, chicken* and the consequent *sardines*, the vertical columns indicate both the strength of the positive-inference (dark grey) and its negation *not sardines* (light grey).

Interactive mosaic plots allow the user to arbitrarily specify sets of antecedents and consequents. Such a technique is designed for focused interpretation where the set of attributes under consideration is small, as the number of elements increase the presentation technique becomes increasingly difficult to interpret.

Figure 3.3 indicates that a potential rule of interest may exist in the form of *heineken, coke, chicken* $\Rightarrow$ *sardines* as there is a significant difference between its confidence and that of all other permutations. The example highlights the contribution of mosaic plots, which although constrained in regard to information volume, provide a novel means of analysing the participants in detail.

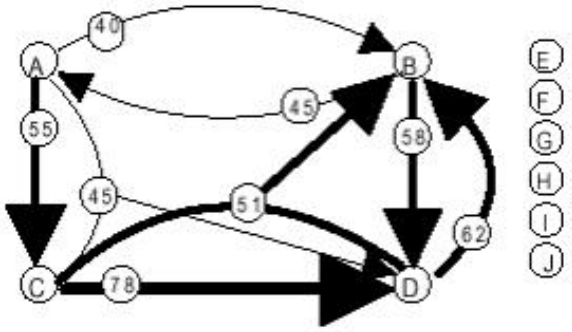


Figure 3.4: Rule Graph (Klemettinen, Mannila, Ronkainen & Verkano 1994)

3.3 Graph-based visualisations

Graph-based techniques present elements as nodes and inferences as links between nodes. Participant visualisations vary in respect to node placement techniques and in the use of graphical primitives to node and inference characteristics. This form of visualisation displays inferences in a more concise manner than matrix-based techniques, as it is based upon the number of elements within the dataset not the number of inferences derived, and therefore it generally achieves better scalability. However, as the number of elements and inferences increase, graph-based visualisations become increasing cluttered and hence more difficult to interpret.

Rule Graph (Klemettinen, Mannila, Ronkainen & Verkano 1994) is a comprehensive directed graph visualisation, with typical node and inference semantics where an inference's strength is represented by arc thickness and label. This technique also incorporates user interaction in the form of rule template specification, through which the user can create display filters to constrain the element set for which inferences are presented. This is illustrated in Figure 3.4 by elements E through J, which have been removed from the presentation and appear to the right. This allows the user to focus on inference subsets, facilitating visual interpretation. Similarly *Rainbows* (Hetzler, Harris, Havre & Whitney 1998) provide a variation in which specific inferences can be focused upon through edge animation.

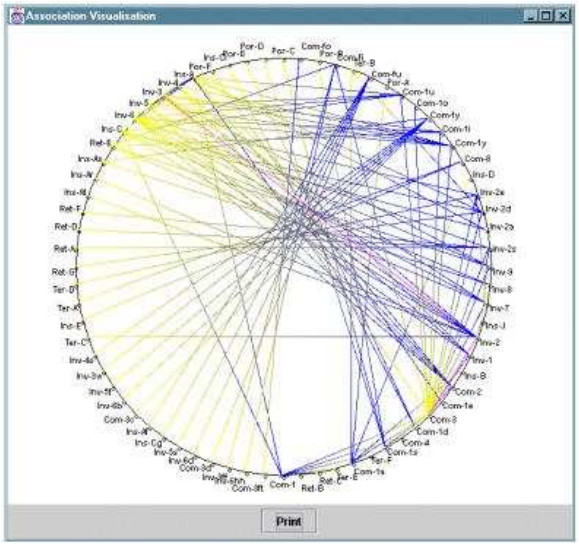
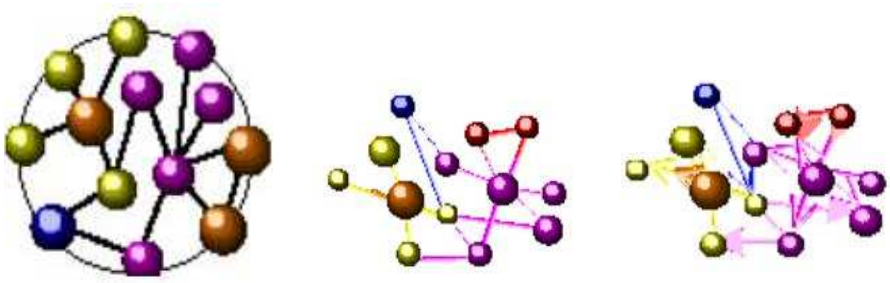


Figure 3.5: Circular association rule visualisation (Rainsford & Roddick 2000)

Rainsford and Roddick (2000) (Figure 3.5) developed a circular visualisation in which the elements are evenly spaced around the circumference and V_2 inferences are represented as chords, coloured with respect to inference direction. This type of visualisation is effective in providing a holistic presentation, concisely representing the entire inference set, effectively indicating general areas of interest. For example, from the figure it is easy to see the participation level of various elements within the entire inference set.

Directed Associated Visualisation (DAV) (Hao, Dayal, Hsu, Sprenger & Gross 2001), is a 3D visualisation technique that maps the elements and inferences to positions and vertices on a sphere, using weighted edges to indicate confidence and arrows for direction. DAV distributes elements equally on a spherical surface (Figure 3.6(a)). Based on physics principles of masses and springs, a support matrix is then created that relates the strength of the inferences in terms of spring tension. The spherical structure is then relaxed and a state of local minimum energy is reached (Figure 3.6(b)), resulting in each elements's relative position reflecting its inference participation. The direction and confidence of each vertex is then calculated (Figure 3.6(c)), and finally presented to the user. This technique also serves as the basis of the current Galicia project visualisation at the University of Montreal (Godin, Missaoui, Huchard & Napoli 2004).



(a) Initialisation (b) Relaxation (c) Direction

Figure 3.6: DAV Process (Hao, Dayal, Hsu, Sprenger & Gross 2000)

3.4 Presentation Interaction

Visualisation facilitates the perception of patterns and structure within data mining results. A static presentation in itself is often inadequate and human computer interaction (HCI) capabilities are required to allow effective exploration of the visualisation. This relates to the Visual Information Seeking Mantra which states that the initial view is qualitative and of an overview nature and through interaction the user can proceed to focus upon interesting sub-areas for more quantitative analysis Shneiderman (1996).

HCI occurs at many different levels as illustrated by the Layered Interaction Model (LIM) devised by Nielson *et al.* (1992) and shown in Table 3.2. This model identifies a sequence of interaction levels which build upon each other, illustrating that the *HCI* requirement can be broken into different levels of abstraction from the subjective goals of the interaction through to the physical I/O of the interaction. There is however a definite shift between the concept-based upper levels and the activity-based lower levels indicating a transition between *what is required* and *how it is done*. The mapping or transformation between the concept and activity levels of interaction is known as *direct manipulation* and occurs at the junction of the syntactic and semantic levels. This section discusses the exploration of presentations through *direct manipulation* and the use of views to facilitate understanding in large complex visualisations.

Level	Name	Exchanged Information	Example
7	Goal	Real World Concepts	Remove letter section
6	Task	Computer oriented actions	Delete 6 lines of text
5	Semantic	Specific operations	Delete selected lines
4	Syntax	Sentences of tokens	Click at left of first char., whilst holding down left mouse button, click to the right of the last character.
3	Lexical	Tokens (information units)	Click at left of first char.
2	Alphabet	Lexemes (primitive symbols)	Click at (200,150)
1	Physical	Hard I/O (movement,click)	Click

Table 3.2: Layered Interaction Model (Nielson 2001)

3.4.1 Direct manipulation

Direct manipulation can be defined as the mapping between the semantic and syntactic levels of the LIM. The objective in constructing such a mapping is to create as close a match as possible between the structure of how users think about a task and the activity used in solving it. This is achieved through pragmatic engineering, so as to maximise problem domain compatibility (John, Rosenbloom & Newell 1985). Pragmatics is a branch of syntax interpretation which deals with relationship of the syntax and their users. Direct manipulation design effects the interactive quality of the system, including error frequency, speed of task performance and user skill retention (Buxton 1986).

However, there is not always a single best set of interaction capabilities for a particular visualisation as users do not have the same mental model and even within a single user the mental model may differ depending upon the user's current goals. Therefore the optimal solution is an intuitive set of mappings that mimic real world activities. For example, the task of *moving* a file in a paper based office involves going to the relevant filing cabinet, picking the required

file and carrying it to its new location. Intuitively the same task on a computer system will follow the same pragmatics. Although not possible to incorporate this form of pragmatism within a text-based system, graphical interfaces provide this capability by representing data as graphical icons and encapsulating activities within interaction mappings. So the movement of a piece of data within the computer system will generally involve the selection of the relevant icon and the dragging of it to its new location. Importantly the syntax of operations in an interaction command should closely correspond to the semantic changes to the data and the screen representation should reflect these changes.

Graphical level interaction is based upon selection and navigation activities which are specified to the computer by the user through pointing devices. Selection is the designation of a point of interest within the graphical interface, signified through a terminal action such as clicking a mouse button or pressing a key. Navigation is the movement of the interest focus which is generally accomplished through a continuous activity such as moving the mouse or holding down specific keys. These primitives work together within different environments to allow the user the means by which any presentation may be explored in a detailed manner.

As there are an arbitrary number of semantic tasks that can be undertaken within a presentation, the overloading of an activity primitive such as selection is overcome by varying the graphical primitive specification. For example, the functionality required within a presentation environment may include the ability to delete items and to display item details in a pop-up dialog, both of which involve item selection. This overloading of the selection primitive is overcome by either varying the singular selection action for each semantic task (e.g. left click for deletion, right click for details dialog) or requiring a combination action either in sequence or parallel (e.g. left click whilst holding down 'D' to delete, left click then press 'I' to display dialog). Another technique used to combat overloading especially in navigation is indirect manipulation, whereby the presentation is manipulated through interaction with associated graphic artefacts, a common example is the inclusion of scrollbars within presentation environments to provide navigation at both the screen and document levels. However this incorporation of indirect manipulation techniques requires that the user's focus be drawn away

from the actual presentation and lessens the interactive experience (Koedinger 1992). The advent of mice with scroll-wheels have overcome this by providing an additional form of vertical navigation, hence providing a means for direct vertical navigation at both the screen and document levels.

Euclidean presentations provide another degree of presentation freedom commonly referred to as depth, given that the first and second degrees of presentation freedom are height and width. Satisfactory interaction with these presentations can be accomplished through the mapping of Euclidean graphical primitives onto regular input devices such the mouse and keyboard. However this increases input overloading and hence the variation of actions required to effectively explore the environment. Alternatively the utilisation of immersive presentation devices such as headsets and pointing devices which allow further degrees of freedom such as flying mice and gloves can increase the users experience of direct interaction and reduce the mapping overload upon input primitives.

The provision of a succinct set of direct manipulation mappings is critical for effective presentation exploration. The level of interactive functionality provided is important as not enough will constrain the exploration process and too much will result in 'overload' and interactive quality degradation. This degradation reflects the provision of too many functional alternatives, resulting in non-intuitive mappings and hence longer task times and less skill retention. Therefore good direct manipulation design involves the specification of a succinct intuitive set of mappings based upon selection and navigation primitives that facilitate the direct comprehensive exploration of a presentation. As a final complication, the correct level of interactive functionality is user dependant with the level of functionality increasing with user experience. Consideration should therefore be given to user specified de/activation of functionalities and visual devices depending upon their requirements.

The following subsections provide a discussion on view filtering and distortion, which are two common interaction based methods used to alleviate some of the problems incurred through the presentation of large complex datasets in co-ordinate space.

3.4.2 View Filtering

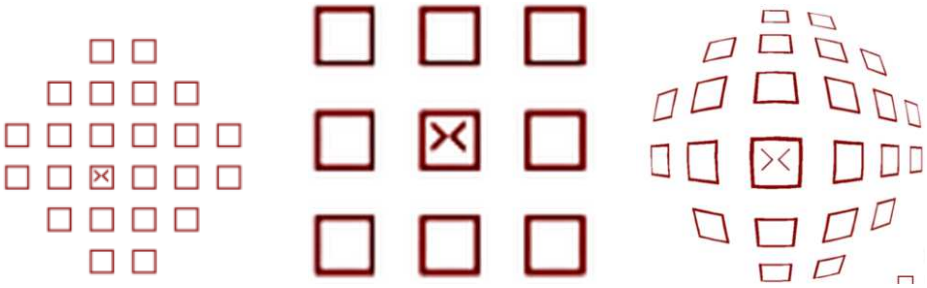
Typical inference sets are large and complex making them difficult to present to the user in their entirety, in a form conducive to user understanding. Large result sets produce cluttered presentations due to the volume of graphical objects required to represent the underlying elements and inferences. View filtering uses direct and indirect manipulation techniques to enhance user perception of the result set by constraining the presentation to a particular subset of interest (Klemettinen, Mannila & Toivonen 1997, Ribarsky, Katz, Jiang & Holland 1999, Wills 1998).

Presentation constraint uses both direct and indirect manipulation depending upon the presentation's characteristics and the nature of the constraint. For example, the exclusion of an element from a graph based presentation would likely involve the direct selection of the element from the presentation canvas, whereas an indirect technique such as the use of a drop-down exclusion list widget may be used in a matrix based presentation. In contrast, global constraint adjustment, such as the heuristic *confidence*, is generally specified through indirect widgets as they apply to the entire presentation and not specific graphical objects and hence cannot be directly manipulated. The specification of constraints is discussed in detail in Chapter 5.

3.4.3 View Distortion

Research into the computer-based distortion of visualisations has been an area of research for the past twenty years and has seen the development of many varied techniques including *Bifocal Displays* (Spence & Apperley 1982), *Fisheye Views* (Furnas 1986), *Perspective Walls* (MacKinley, Robertson & Card 1991) and *Frustrum displays* (Anderson, Smith & Zhang 1996). The objective of distortion is to provide, within a graphic environment, space for the magnification of an area of interest whilst providing context through compression of the rest of the image (Sheelagh, Carpendale, Cowperthwaite & Francis 1997). This allows the user to focus on a particular presentation area whilst still maintaining a holistic view of the presentation. Without distortion techniques, zooming upon an area of inter-

est can result in context loss, as graphics objects move outside the viewable area as illustrated in Figure 3.7.



(a) Original image (b) Zoom with context loss (c) Fisheye distortion

Figure 3.7: Graphical Fisheye Views (Sarkar & Brown, 1994)

Focus without context loss improves interpretation of the information presented as all relationships between the underlying elements are still evident whilst providing more detailed information about a particular element or set of elements and their immediate surroundings. Note that although distortion focus is generally at a single point (as shown in Figure 3.7(c)) it can have multiple foci, allowing the scrutiny of many areas at once without context loss.

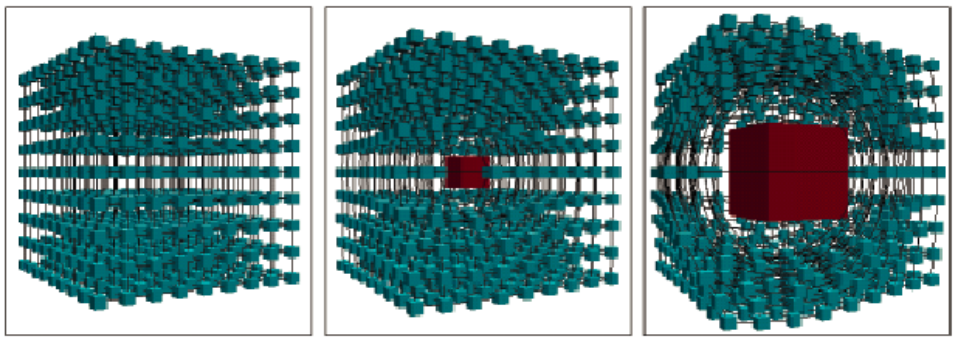


Figure 3.8: Removing occlusion through distortion (Sheelagh *et al.* 1997), left: original view of matrix, central: selection of required item of interest, right: visual access distortion.

Distortion within Euclidean space can additionally provide a solution to the problem of occlusion in which graphical objects are hidden by those graphical objects closer to the view point. This concept is described by Sheelagh *et al.* (1997) and illustrated in Figure 3.8. This set of images shows an initial dense matrix of similar objects and the internal object of interest, indicated by colouration and enlargement. As the object of interest is internal, regular manipulation of the presentation may not be adequate as the object will either still be occluded through *rotation* or context will be lost through *zooming*. The third image shows the use of Visual Access Distortion which clears the line of sight to the chosen focal region, providing an unobstructed focus.

Translucency can also be used to provide a solution to occlusion in Euclidean space, by causing the objects closer to the viewing point to become semi transparent. This technique can be applied to facilitate the presentation of inferences (as discussed in Chapter 4).

3.5 Summary

This chapter provides a review of association rule presentation techniques used to facilitate the user's interpretation of association mining results. The review discusses the perceptual advantage of graphical over textual presentation and identifies two classes of graphical association presentation 1) matrix-based and 2) graph-based, in which graph based techniques appear more powerful due to their superior presentation flexibility.

This survey (in conjunction with Chapter 1) serves as a foundation upon which further research into the hypothesis of guided association mining is based. To support this further research, this chapter presents a discussion upon presentation interaction techniques, which are required to provide a guided mining environment (Part III).

As a result of research undertaken in the construction of this review, the novel concept of incorporating hierarchical semantics within an association rule presentation was proposed. This technique, known as Concentric Association Rule Presentation ($CARV$) is presented in the next chapter (Chapter 4).

Chapter 4

CARV: Concentric Association Rule Visualiser

A typical mining run can produce many associations that need to be presented to the user in a manner that facilitates interpretation. Presentations have evolved from text to graphic visualisations enabling the better use of human perceptual capabilities. Researchers have devised several techniques for the visualisation of regular associations built upon matrix and graph based techniques. However our research has found no current techniques that allow for the effective visualisation of hierarchical inferences. The problem lies in finding a technique by which the hierarchy and inference contexts can be merged into a single effective presentation.

To this end we present a novel and useful technique, the *concentric association rule visualiser* (CARV), which allows the simultaneous visualisation of both hierarchical and association semantics. This chapter provides a review of hierarchical presentation techniques and then introduces CARV, discussing its design, functionality and implementation.

4.1 Visualisation of Hierarchies

In general, a visualisation's objective is to display data in a manner that facilitates user interpretation. Hence the visualisation of hierarchical semantics should match the user's concept of a hierarchical structure, a layered structure within

which each node has a parent and a number of children. This structure is known as a rooted tree as the structure emanates from a single root node, in comparison to free trees that do not encode any structure apart from their topology (Eades 1992). Rooted-tree visualisations can be thought of as belonging to either of two groups: graph-based in which the hierarchy is presented as a collection of nodes with connecting arcs representing the hierarchical semantics, and embedded in which the parent's graphical representation encompasses that of its children. This section provides an overview of the types of hierarchical visualisation.

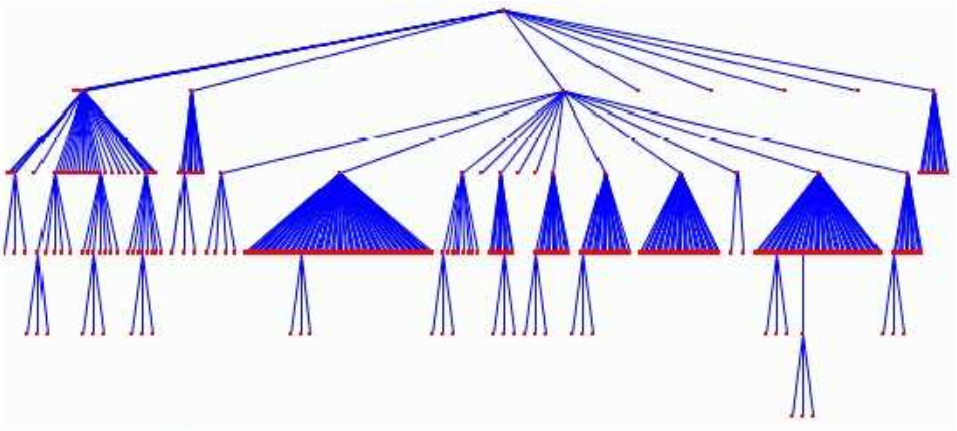


Figure 4.1: Classic Tree Visualisation (Herman, Melancon & Marshall 2000)

Graph-based visualisations use nodes and connecting arcs to represent hierarchical semantics, a set of rules for the effective placement of these nodes was developed by Wetherell and Shannon (1979). The classical tree visualisation positions child nodes under their parents as shown in Figure 4.1 (Herman, Melancon & Marshall 2000). Building upon this work, two significant visualisation techniques based upon a fractal approach were devised, the *radial* and *cone-tree* models.

The radial model (Eades 1992) positions nodes on concentric circles according to their depth in the tree. Each sub-tree is positioned over a sector of the circle and a *convexity constraint* forces the sub-tree wedge to remain convex ensuring that adjacent sub-trees do not overlap (Figure 4.2).The cone-tree model by Robertson

et al. (1991) differs in that each sub-tree is represented by a circle connected to the parent node (Figure 4.3). A 2D version of the cone-tree model can be obtained by projecting the model onto a plane (Carriere & Kazman 1995), termed a *balloon view* (Figure 4.4).

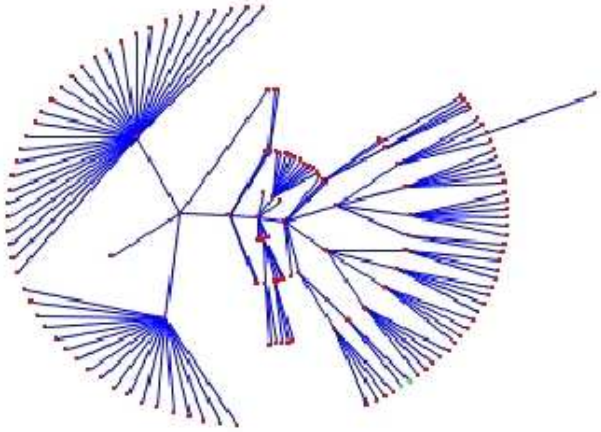


Figure 4.2: Radial visualisation (Eades 1992)

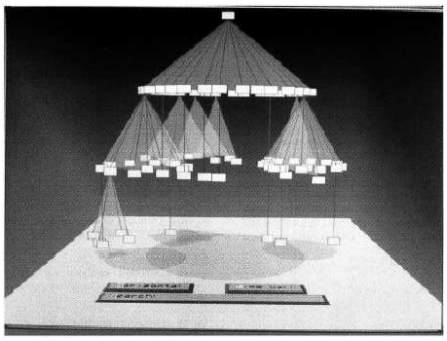


Figure 4.3: Cone-tree visualisation (Robertson, Mackinley & Card 1991)

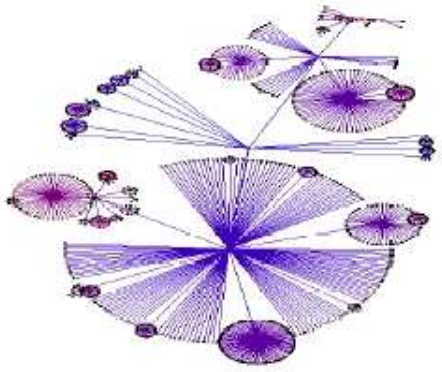


Figure 4.4: Balloon View (Carriere & Kazman 1995)

Embedded visualisations represent hierarchical semantics by encapsulating sub-trees within the parent's graphical representation in a manner similar to Venn diagrams. Examples include *Treemaps* (Johnson & Schneiderman 1991), *Bubble Trees* (Boardman 2000) and *Onion Graphs* (Sindre, Gulla & Jokstad 1993). Figure 4.5, a Treemap, illustrates the principle by recursively splitting the screen into

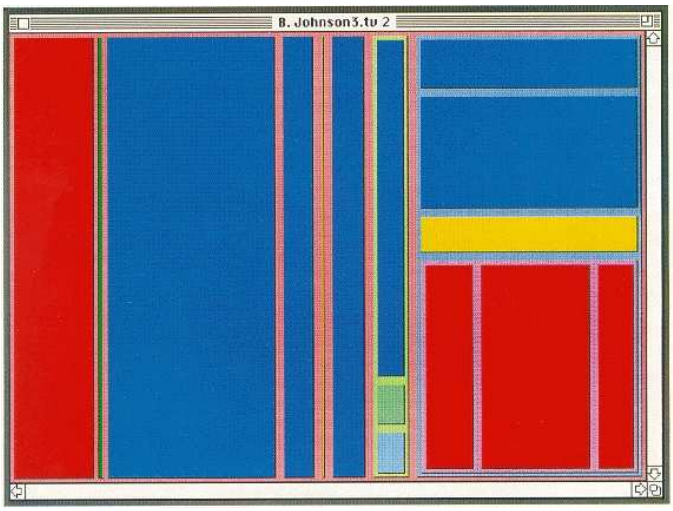


Figure 4.5: Treemap (Johnson & Schneiderman 1991)

rectangles of alternating orientation as the hierarchy is traversed. Onion Graphs are a circular version of this technique and Bubble Trees reduce clutter by only displaying three hierarchy levels simultaneously and requiring user interaction to traverse the tree.

The visualisation of hierarchical information in an effective manner has been an active field of research for the past two decades and there is an abundance of literature available. However research indicates that much of this work involves extensions and variations of the techniques summarised above, in order to better visualise larger datasets (Koike & Yoshihara 1993, van Wijk & van de Wetering 1999). This generally involves the incorporation of functionality such as navigation and focus + context concepts (Herman, Melancon, de Ruiter & Delest 2000, Kreuseler & Schuman 1999, Yee, Fisher, Dhamija & Hearst 2001).

4.2 Concentric Association Rule Visualisation

A hierarchical association visualisation requires the visualisation of elements and their inferences within a structure that captures hierarchical semantics. This requires the simultaneous visualisation of both the hierarchy and association structures in an effective and intuitive way. Background research into the visualisation of hierarchical and association semantics, summarised in Sections 1.3.3 and 4.1,

indicated the effective use of graph-based techniques to represent both types of semantics. While matrix-based and embedded techniques are useful within their respective fields they are more tightly constrained and cannot accommodate the introduction of other semantics.

In comparing the suitability of the different graph-based techniques, our focus was upon the aesthetic quality of the resulting visualisation. Scalability, which often seems be a key issue in data visualisation, is of less importance as the resulting visualisations will typically not be large. This is because although a typical dataset may contain thousands of different elements, the resulting presentation is significantly smaller, both in regard to the participant hierarchy and discovered inferences, through constraint inclusion. Which, in the case of *support* excludes those elements of little impact, reducing presentation content to the extent that it can be easily interpreted by the user. Herman *et al.* (2000) argue that from a cognitive perspective it makes little sense to display large amounts of data because understandability is reduced.

Much research has been undertaken in regard to graph aesthetics (Battista, Eades, Tamassia & Tollis 1999, Purchase 1998) and the main types of graph-based visualisation conform to these (see Section 4.1). By basing node positioning upon hierarchical semantics and using an existing hierarchical graph-based technique many aesthetic qualities are assured. The issue then becomes the preservation of these aesthetics when association semantics are introduced. The introduction of association semantics as additional edges will not effect those aesthetic qualities that relate to node placement, it may however have a detrimental effect upon presentation clarity through obscurement.

In relation to both the hierarchy and inferences, obscurement will result in information loss and therefore obscurement can have a significant impact upon the understandability of the visualisation. Hierarchy obscurement will occur in visualisations that rely upon edges as well as nodes to convey the hierarchical semantics. Of the three core graph-based models, the cone-tree model (Section 4.1) relies heavily upon edges in conveying hierarchy semantics. The other two models can effectively present hierarchy semantics solely from node placement.

The classic model, although the most effective in regard to minimising hierarchical obscurement because of its unidirectional layout, will result in excessive inference obscurement due to this same layout. For example from Figure 1, if there existed two inferences *Whiskey* ⇒ *Fruit Juice* and *Gin* ⇒ *Soft Drink*, the first would be overlain by the second and hence obscured. This form of obscurement is not a significant issue in the radial and cone-tree models because of their more flexible layout.

Based upon the primary need to minimise obscurement the radial model is the best suited technique as it is simple enough to convey the hierarchy without the use of edges and therefore association obscurement is minimised due to its circular layout. Additionally, previous work by Rainsford and Roddick (2000) (Section 1.3.3), has shown the effectiveness of using circular representations and as the visualisation is bounded by the hierarchy, the visualisation will be less obscured than one based upon the classic or cone-tree models. Figure 4.6 shows the visualisation using radial layout of the complete hierarchy used in Figure 1, in which the arcs indicate sibling elements.

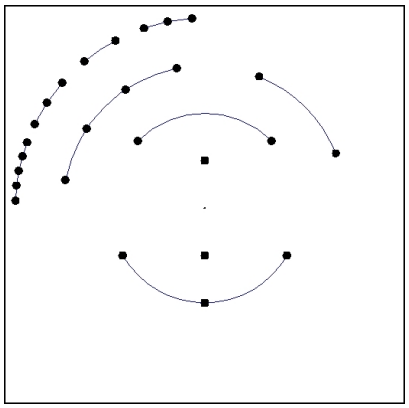


Figure 4.6: Radial visualisation of example hierarchy

The inference can then be simultaneously visualised using two techniques: direct method, in which relationships are drawn directly between the associated elements, and the trail method, where the relationships are drawn along the hierarchical path. For example, using the trail method and the hierarchy in Figure 1 the rule *Whiskey* ⇒ *Soft Drink* would be represented as a line that traverses the

hierarchy from *Whiskey* to *Soft Drink* through nodes *Alcohol*, *Drink* and *NonAlcohol*. These techniques are illustrated in Figure 4.8, based upon the hierarchical frame presented in figure 4.7, which shows the result of an association mining session using a test dataset. The direct method (Figure 4.8(a)) is useful in tasks that seek to identify general trends within the rule set, however the hierarchy is quickly obscured as the number of inferences increase. The trail method (Figure 4.8(b)) results in a less cluttered visualisation of the hierarchical semantics. However, due to inference overlay there is a high level of obscurement, thus resulting in an ineffective visualisation.

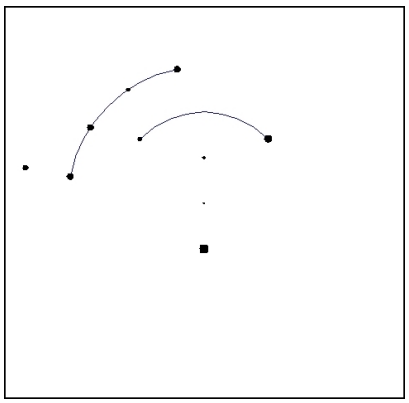
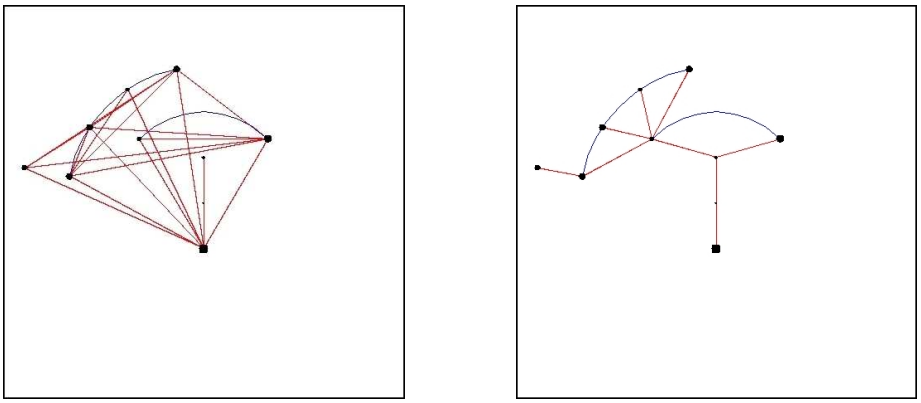


Figure 4.7: Hierarchical Frame.



(a) Direct method (b) Trail method

Figure 4.8: Incorporation of inferences upon a radial hierarchy framework using both direct and trail methods.

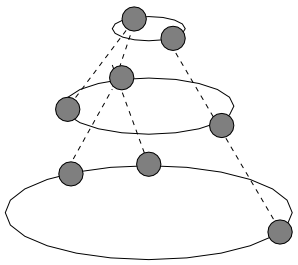


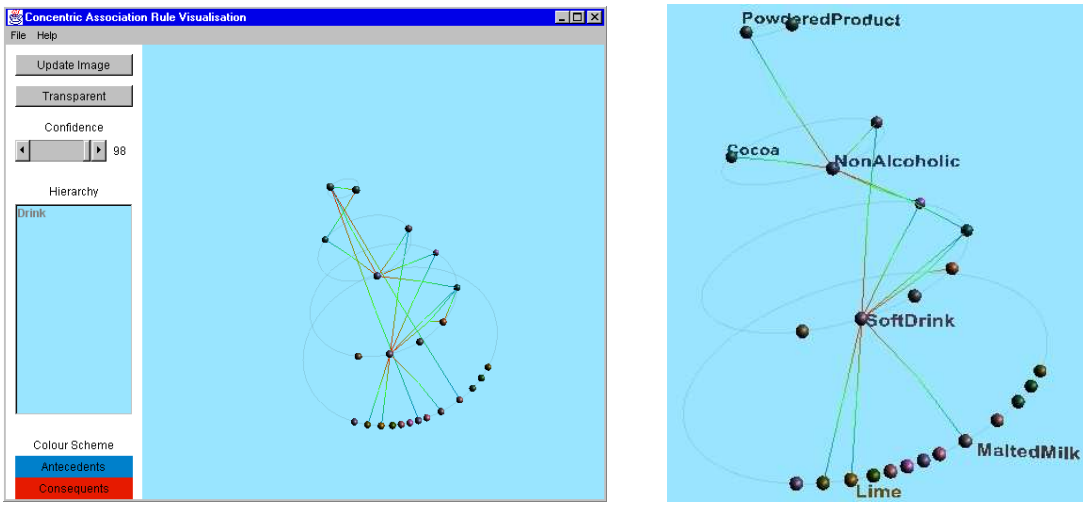
Figure 4.9: Conic model representation of item hierarchy

From Figure 4.8 it is apparent that the direct method is more effective in representing associations, as overlaying of associations is, in general, avoided, however this method leads to hierarchy obscurement. An effective solution, similar to that proposed by Baker *et al.* (2002) to clarify the visualisation of genetic regulatory networks, is to make the hierarchy more prominent by extruding the two dimensional radial model into the third dimension, effectively creating a transparent conical model (Figure 4.9). Within this model elements become more prominent as they lay upon the cone's surface, while both intra and inter-level inferences, within and across hierarchy levels respectively, lie within the cone and hence do not obscure the hierarchical representation.

Although the direct method avoids most overlays, they will occur where multiple associations with the same constituent elements exist. For example the inferences $Gin \Rightarrow SoftDrink$ and $SoftDrink \Rightarrow Gin$, which have different connotations, will be overlaid using the direct method. An effective solution to this problem is the introduction of an inference vertex. A vertex is an approximate midpoint of the constituent elements, where the vertex must be unique to ensure no overlaying occurs. This is accomplished by checking new vertices against a listing of existing vertices. If non-unique the new vertex is moved slightly in a random direction and rechecked.

The introduction of a vertex also provides a method for the effective visualisation of inferences in which the number of constituent elements is greater than two. This provides a capability not apparent in some current graph-based tech-

niques that only present V_2 associations, such as Rainsford & Roddick (2000) and Hao *et al.* (2001), presented in Chapter 3. Rule Graph (Klemettinen et al. 1994) although capable of visualising V_n inferences, does not present a generally applicable solution to the overlay problem.



(a) Interface with rotated model

(b) Labelling and transparency

Figure 4.10: CARV implementation snapshot

4.2.1 Implementation

The implementation of these concepts resulted in the development of the Concentric Association Rule Visualiser or *CARV* (illustrated in Figure 4.10(a)), in which the implementation of 3D environment functionality such as zooming, panning and model rotation provides a direct and effective inference model exploration. Associated functionalities of interest include labelling and focus, presented in Figure 4.10(b). Focus is achieved by using transparency to hide inferences of little interest. For example, within Figure 4.10(b) only the inferences in which the concepts *Soft Drink* and *Non Alcoholic* participate are apparent.

Figure 4.11 illustrates two important features of the visualiser: firstly the ability to represent non-hierarchical or regular association mining results, and the inclusion of vertices to eliminate overlay and visualise complex inferences.

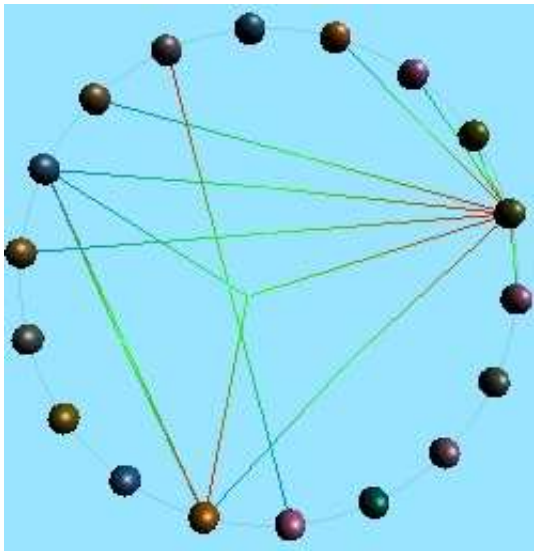
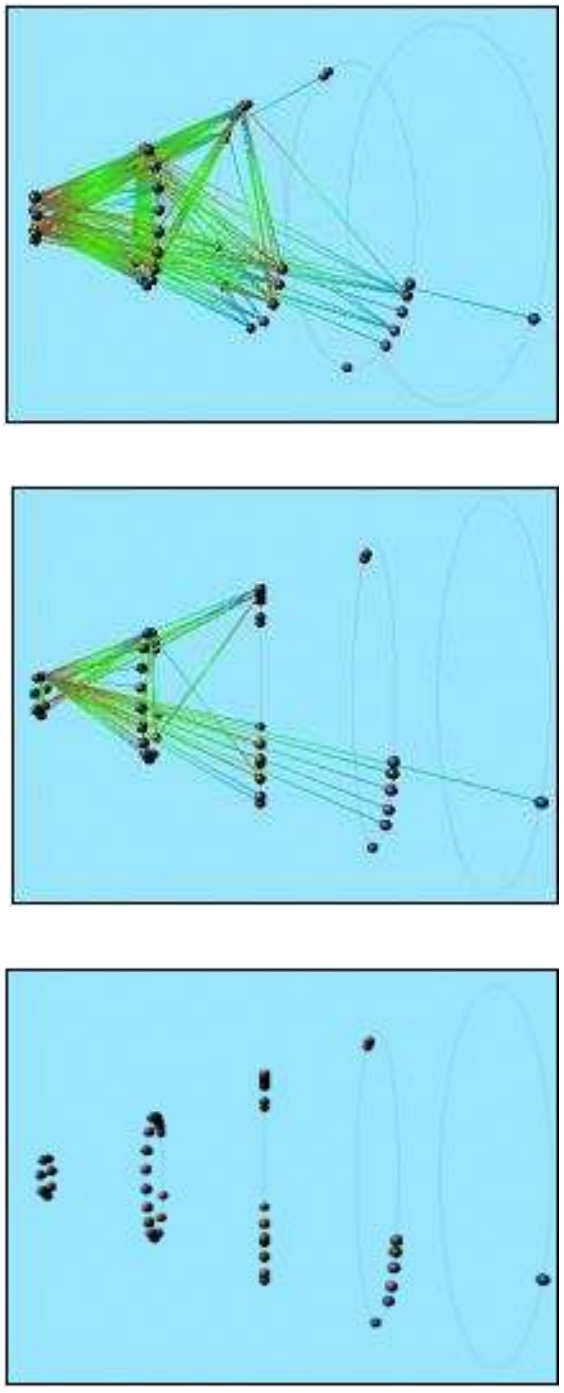


Figure 4.11: Non-hierarchical visualisation illustrating vertex inclusion

4.3 Dynamic Inference Presentation

The concept of CARV was developed to provide a solution to the visualisation of hierarchical inferences, however it has also been used to provide a base visualisation component for subsequent research into guided association mining. Where a guided association mining system requires in effect a *merging* of the analysis and presentation stages of the mining system, allowing the user to *see* the analysis process and provide feedback to it (presented in detail in Part III). To this end CARV has been extended to enable the dynamic presentation of inferences and to allow for interactive functionality, through direct manipulation, so that the user can insert guidance within the analysis process.

An example of the dynamic presentation capability is provided in Figure 4.12. The images illustrate the initial visualisation of valid elements (Figure 4.12(a)), the dynamic visualisation of inferences (Figure 4.12(b)), and the completed visualisation (Figure 4.12(c)). Further discussion and presentations of CARV in relation to the provision of guidance are provided in Chapter 8.



(a) Initial visualisation of itemsets (b) Intermediate visualisation stage (c) Final visualisation

Figure 4.12: Dynamic CARV

Part IV

Guided Association Mining

Anyone who has ever used a computer
knows that interactive tools are
indispensable.

Surajit Chauduri 2002

Chapter 5

Analysis Constraint Review

The unconstrained analysis of the dataset (D) results in the discovery of an inference set that grows exponentially as the number of dataset elements increases linearly, $|R| = 2^{|E|} - 1$. Analysis constraint, or the inclusion of constraints within the analysis process is fundamental to maintaining exploration tractability, given the typically large and complex nature of target datasets.

Furthermore the user specification of constraints provides a mechanism whereby the user's subjective knowledge about the domain and the task at hand can be incorporated within exploration. This results in an inference set that better reflects the user's focus, enabling the specification of characteristics of interest to the user, in essence providing a limited form of analysis synergy between the user and computer. Therefore user involvement can enhance the usefulness and performance of mining processes, by eliminating information that is of no interest to the user.

This chapter presents a discussion about constraint specification and its inclusion within association analysis. Section 5.1 presents the different types of constraints, while Section 5.2 provides a review of the different techniques by which analysis is currently constrained.

5.1 Constraint Classes

There are two classes of constraint: domain and functional, which serve as the building blocks for complex constraint specification, where domain constraints involve set comparisons and functional constraints are based upon mathematical functions and operators. Prior research by Ng *et al.* (1998) describes four classes of constraint, however this thesis argues that there are only two foundation classes: domain and functional, that definitively encapsulate the constraint class domain. From these all other constraint classes, including those proposed by Ng *et al.* (1998), are built through sub-classing and complexity.

Given two elementsets A and B that are subsets of the common domain E, then $A \lambda B$ represents the common domain constraint form, where λ is a logical operator, $\{=, \neq, \subset, \subseteq, \supset, \supseteq \in, \notin\}$. For example, given the partial hierarchy in Figure 1 and the generation of elementsets, Examples 1 and 2 respectively represent the constraint of valid elementsets to those containing *GingerBeer* and those not containing the abstract concept *Alcohol*, while Example 3 presents a domain constraint based upon excluding an element characteristic, *colour*.

Example 1 $A \supset \{GingerBeer\}$

Example 2 $A \not\supset \{Alcohol\}$

Example 3 $A.colour \notin \{red, blue\}$

Functional constraints, of the form $A\theta B$, relate to the boolean comparison of two operands, $\{=, \neq, <, \leq, >, \geq\}$, which are either constants or functions. Given x is a constant and *age* is a numerical element characteristic, Examples 4 - 6 present sample function constraints.

Foundation classes can then be built to describe more complex constraints, either positive or negative, including inference and conjunct constraints. Positive constraints specify *what the user wants* and hence the valid concepts, either elementsets or inferences, that satisfy the constraints. Negative constraints, represent *what the user already knows*, and therefore valid concepts are based upon constraint violation or unexpectedness.

Example 4 $freq(A) \geq x$

Example 5 $freq(A) \geq avg(freq(B) \in D)$

Example 6 $sum(S_{age}) \leq Sum(T_{age})$

Constraints are of most effect during elementset generation, in reducing exploration by applying the constraint to each candidate elementset. However inference constraints can also be applied during inference derivation, and enable the positive and negative constraint of either the antecedent or consequent. While more specific than elementset constraints, some inference constraints can be incorporated during elementset generation and again during inference derivation. This enables the inference constraint to aid an exploration constraint. If an inference is valid then its underlying elementset is also valid, given that the inference constraint is generically applied to each elementset irrespective of antecedent or consequent specification, which is handled during inference derivation.

For example, given an elementset A from which derived inferences take the form $A_a \Rightarrow A_c$, then Example 7 represents an inference constraint that can also be used to constrain exploration. Example 8 can only be included during inference generation as the comparison is between antecedent and consequent subsets.

Example 9 represents a conjunct constraint that has been constructed through the joining of multiple simple constraints. This constraint involving a conjunction of inference constraints to be incorporated during inference derivation. However the portion $S_c.type \in \{alcoholic\}$ can also be used to constrain exploration as it refers to the existence of a domain concept within all valid elementsets.

Example 7 $sum(A_{a.age}) \geq x$

Example 8 $sum(S_{a.price}) \leq sum(S_{c.price})$

Example 9 $size(S_a) > 2 \wedge S_{c.type} \in \{alcoholic\} \wedge freq(S_a) > x \wedge freq(S_c) > x$

An important constraint attribute is that of reflexivity, where many elementsets can be removed from consideration given a reflexively invalid elementset. This is based upon the principle of directed validity, where given the *monotonically* valid elementset A, all *supersets* of A are also valid (Definition 1). For example, consider the functional constraint *count(A)*$> x$, where x is a constant, then if A is valid the all supersets of A will also be valid due to the static value of x. The opposite is true for *non-monotonically* reflexive constraints, such as *support*, whereby given the *non-monotonically* valid elementset A, then all *subsets* of A are also valid (Definition 2). For example, given the common functional constraint σ, then if $\sigma(A)$ is valid all subsets must be valid. Therefore given a top-down lattice exploration incorporating the non-monotonic constraint *support*, valid V_κ can be derived from $V_{\kappa-1}$ as all subsets of a valid elementset must also be valid (see Chapter 1).

While all domain constraints exhibit reflexivity as they relate to the concept existence, not all functional constraints are reflexive and are therefore not useful in regard to exploration reduction. For example, consider the mathematical function *average*, as it is not directional within a set domain, ever increasing or decreasing, then it is not reflexive. Given $avg(A)x$ it cannot be predicted, based upon the introduction or removal of an element, whether x will increase or decrease.

Definition 1 (monotonic) $\forall\, A, B \mid (A \subseteq B \subseteq E \ \wedge V(A)) \Rightarrow V(B)$

Definition 2 (anti-monotonic) $\forall\, A, B \mid (B \subseteq A \subseteq E \ \wedge V(A)) \Rightarrow V(B)$

The use of uncertainty (or fuzziness) in the definition of functional constraints has been proposed by Liu *et al.* (1999), based upon the principle of fuzzy linguistic variables (Zimmermann 1996). An element's characteristic is associated with a defined set of *concepts*, each of which represents a subset of the charateristic's domain. For example, given the element characteristic *price* with a range of $r = [0.05, 200]$ and an applicable set of concepts $c(price) = \{cheap, reasonable, expensive\}$. Then each concept is mapped to r denoting a degree of membership, $m(concept)$. Example 10 presents a definition of $m(expensive)$. Based upon this, fuzzy constraints, both positive and negative, can be specified in which an inference's conformity to a fuzzy constraint is

specified by its membership within that constraint. Example 11 presents a fuzzy constraint based upon the specification of fuzzy concepts for *price* and *colour*.

Example 10 $m(expensive) = \{(u, \mu_{expensive}(u)) \mid u \in [0.05, 200]\}$

$$\mu_{expensive}(U) = \begin{cases} 1 & : \quad u \in [120, 200] \\ \frac{u-50}{70} & : \quad u \in [50, 120] \\ 0 & : \quad u \in [0.05, 50] \end{cases}$$

Example 11 $S.price > cheap \wedge S.colour = dark$

Constraints are often referred to as Measures Of Interest (MOI) within the literature and are commonly classified as being either objective or subjective. Where objective measures are dependent upon statistics and the underlying dataset, while subjective measures depend upon user beliefs in respect to what is of interest to them (Silberschatz & Tuzhilin 1996). This classification however appears flawed in its use of the term objective. A Measure Of Interest, by its definition, cannot be objective, as it embodies a user's belief and therefore is always subjective. For example, the inclusion of the "objective" MOI *support*, is based upon the statistical validation of elementsets according to a specified threshold, which is specified by the user based upon their beliefs. Therefore *support* is ultimately subjective.

Measures Of Interest is an important, large area of research, however this work focuses upon constraint inclusion within analysis and not upon specific constraints or MOI's. For detailed reviews see Silberschatz & Tuzhilin (1996), Hilderman & Hamilton (1999), Padmanabhan & Tuzhilin (2000), Tan *et al.* (2002) and Wang *et al*(2003).

5.2 Constraint Inclusion

Constraints can target elements, elementsets or inferences and are incorporated at the earliest effective point within the mining cycle, given the constraint's target, to maximise the constraints effect upon analysis reduction. For example, prior to

analysis, only constraints targeting elements or "raw data" can be incorporated, as elementsets and inferences do not yet exist. While constraints targeting elementsets are incorporated during exploration, and inference based constraints are included during derivation. However some inference based constraints can also be included during exploration in a more general form to reduce exploration, as discussed in Section 5.1.

The seminal works in constraint analysis were published in the early 1990's with researchers proposing techniques to constrain the number of generated inferences presented to the user, through the removal of redundant inferences and template specification, thereby reducing cognitive load (Han, Cai & Cerone 1992, Toivonen, Klemettinen, Ronkainen, Hatonen & Mannila 1995, Klemettinen, Mannila & Toivonen 1996). However, the most notable contribution of this period was the introduction of reflexive constraints within analysis to significantly reduce exploration, the most common being *support*, (Agrawal et al. 1993), where *support* measured an elementset's dataset presence providing a quality heuristic that resulted in the discovery of only the frequent elementsets within D, given a user specified threshold, $minsup$.

The following subsections (5.2.1 ... 5.2.3) provide a review of constrained mining techniques separated into the stage of constraint inclusion. Dataset Constraint (Section 5.2.1) presents the inclusion of constraints upon the dataset before analysis. Exploration constraint (Section 5.2.2) discusses the inclusion of constraints *within* analysis and Inference Derivation Constraint (Section 5.2.3) discusses the incorporation of constraints during the derivation of inferences.

5.2.1 Dataset Constraint

Constraints included prior to analysis focus upon the removal of data, or elements, from D. In its simplest form this is achieved through the selection of the dataset to be mined, or the construction of the analysis dataset D, through querying of source database(s) via SQL. Most dataset constraint research has pragmatically focused on the extension of standard SQL for its use within a data mining environment.

Meo *et al.* (1996), propose an SQL operator *Mine Rule* that allows the SQL specification of dataset constraints to be incorporated before analysis. However, the join operations used are inefficient. This operator was subsequently extended to allow for the specification of inference constraints, that were incorporated during inference derivation (Yen & Chen 1997). Similar SQL extensions have also been proposed that enable the embedding of constraints within a data mining query language (Han, Fu, Wang, Koperski & Zaiane 1996, Imielinski & Virmani 1999).

Ng *et al.* (1998) proposed the concept of succinctness to identify forms of constraint that can be used to identify and subsequently constrain the exploration space before analysis begins. Succinctness, as proposed by Ng *et al.* (1998) applies to single-variable constraints only, where the constraint can be expressed as a relational algebra selection predicate, $\sigma_p(E)$. Given a number of succinct sets $A_1, \ldots, A_k \subset E$, a constraint is succinct if it can be expressed in terms of the strict powerset of these succinct sets using union and minus operators.

For example, the constraint $\{Alcohol, SoftDrink\} \subseteq A_{Type}$ is succinct as it can be reduced to a simple selection on individual elements. In more detail, the constraint specifies that valid elementsets must contain at least one element of type *Alcohol* and one of type *Softdrink*. For example, given the succinct sets $A_1 \ldots A_3$ below, the constraint is succinct as its applicable search space, V, can be expressed as in Example 12, where 2^{A_1} represents the powerset of E_1.

Therefore if a constraint is succinct, the valid exploration space in respect to this constraint can be identified before analysis, allowing the generation of all valid elementsets satisfying the constraint without exploration being undertaken.

Example 12 $V = 2^E - 2^{A_1} - 2^{A_2} - 2^{A_3} - 2^{A_1 \cap A_3} - 2^{A_2 \cap A_3}$

$$A_1 = \sigma_{type='Alcohol'}(E)$$

$$A_2 = \sigma_{type='SoftDrink'}(E)$$

$$A_3 = \sigma_{type\neq'Alcohol' \vee type\neq'SoftDrink'}(E)$$

Subsequent research, by Lakshmanan *et al.* (1999), extended the concept of succinctness to two-variable constraints, by introducing the concept of *quasi-succinctness*. Whereby a quasi-succinct two-variable constraint can be reduced to two one-variable constraints, upon which the succinct pruning techniques can be applied. Furthermore the concept of succinctness and quasi-succinctness were applied to the efficient computation of correlated sets, extending Brin *et al.* (1997).

Goethals & Bussche (2000) propose dataset constraint given a set of domain based inference constraints. This is accomplished through the generation of a set filter (S) based upon a user specified rule filter, R^f, of the form $Ante \Rightarrow Cons$ where each element of $Ante$ and $Cons$ relates to an element's inclusion or exclusion within valid inferences. This results in positive and negative element filters, S_{pos} and S_{neg} respectively, that are used to reduce D to only those objects that are supersets of S_{pos} and from which all elements in S_n are removed, resulting in D_s. Association mining upon D_s is then conducted, resulting in the discovery of all valid elementsets V, with respect to S. In effect, each elementset represents the elementset $X \cup S_{pos}$, where X is an arbitrary elementset, and hence no invalid candidates are generated.

Subsequent inference derivation ensures that all valid rules satisfying R^f are generated, where the rules antecedent (consequent) is a superset of R_A (R_B) and does not contain any R_B (R_A). To calculate rule confidence an additional scan of D is required to calculate the *support* for all rule participating subsets that are disjoint to S_{pos}. This results in the valid set of rules for the defined rule constraint.

Xia *et al.* (2002) propose the use of Bayesian networks to efficiently embed user beliefs within a compact but expressive structure. Furthermore the authors propose three techniques by which the set of objects within D that satisfy the user's criterion can be efficiently selected for subsequent mining.

5.2.2 Analysis Constraint

The inclusion of constraints within analysis is significant, reducing exploration and discovered inference quantity, ultimately improving result quality. This enables a limited form of synergy to be incorporated during analysis and is the

forerunner to guided analysis. Applicable constraints target elementset generation and inference derivation through the inclusion of both functional and domain based constraints. Agrawal's seminal work that presented the Apriori algorithm, (Agrawal et al. 1993), introduced the use of a functional constraint within analysis, namely *support*, while domain based constraints were introduced into analysis by Srikant *et al.* (1997).

Srikant *et al.* (1997) introduced the inclusion of complex domain constraints within analysis where a domain constraint is of the form $X_1 \wedge X_2 \wedge ... \wedge X_n$ where each disjunct X_i of the form $x_{i1} \vee x_{i2} \vee ... \vee x_{im}$ specifies a set of elements, where each element x_{ij} is either positively or negatively constrained. When a concept hierarchy is available, an element can also be contextually constrained in regard to the hierarchy. For example, based upon Figure 1, $(Gin \vee Coffee) \wedge descendants(Softdrink)$ is a valid complex domain constraint.

However it was found that the implementation required the subsequent generation of subsets during inference derivation, which although invalid in relation to the domain constraint, were required to derive inferences, as they had a valid superset. This work was extended by enabling the specification of domain based inference constraints and optimising exploration by incorporating them before analysis through dataset pruning (Goethals & Van den Bussche 2000).

Liu *et al.* (1999) propose the concept of applying multiple constraints upon an element characteristic, by specifying the group of elements upon which a particular constraint is valid. The implementation is simple, with the constraint being attached, in this case *support* to each element. During exploration valid elementsets are those whose *support* exceeds the lowest participant element's threshold, instead of a global threshold. This work was subsequently extended through the notion of *support* constraints, that allow direct *support* specification for elementsets that can subsequently be treated as a single concept (Wang, He & Han. 2000).

Readt & Kramer (2001) propose a technique that naively incorporates reflexive constraints by incorporating anti-monotone constraints within analysis and subsequently re-analysing the output, incorporating any monotone constraints. Dualminer, (Bucila, Gehrke, Kifer & White 2002), provides an optimisation by

incorporating both monotone and anti-monotone constraints in each level-wise scan of D, by applying the monotone constraints to the elementset complements. For example, given $E = \{a, b, c, d\}$, the complement of a is $\{\bar{a}\} = \{b, c, d\}$, resulting in the simultaneous reduction of the search space from both directions at once. This concept has been subsequently adapted for closed-set mining in MIN-Ex (Boulicaut & Jeudy 2001).

Pei *et al.* (2001) extend functional constraint research by identifying and proposing optimisations for a constraint sub-class, referred to as *convertible constraints*. The authors refer to the class members as *tough* as they are neither reflexive nor succinct, including $sum(i)$ and $mean(i)$. The optimisation is based upon the theory that tough constraints often become reflexive in the presence of certain element orderings and by ordering the set of elements in regard to the characteristic in question, either in descending or ascending order, the constraint can be dealt with in a reflexive manner. Due to its focus upon element ordering, FP-growth (Han et al. 2000) provides a conducive environment for convertible constraint inclusion, not possible in classic level-wise algorithms due to ordering irrelevance.

Wang *et al.* (2003) propose a novel Divide-and-Approximate approach for pushing functional constraints into the analysis of Iceberg-cubes. This differs from previous constrained analysis research as Iceberg-cubes deal with tuple-based instead of element based mining. Therefore the element order approach to function constraints taken by Pei *et al.* (2001) cannot be applied, as no such tuple-ordering exists. The suggested approach optimises processing by pushing the functional constraint so that only *likely* cells are examined. Through *approximators*, which are approximations of the function constraint, the algorithm converges upon the constraint using a divide and conquer strategy.

5.2.3 Post-Analysis Constraint

Given that analysis results in a discovered inference, a number of techniques have been proposed that manipulate this inference set to improve the subsequent presentation quality by reducing quantity and increasing interestingness. This is achieved before actual presentation by implementing rules (as discussed below)

or during presentation to facilitate interpretation through the use of interactive views or graphical interaction (Xiao & Dunham 2001, Wills 1998, Chu & Wong 1998, Klemettinen et al. 1997, Ribarsky et al. 1999).

Shah *et al.* (1999) propose a set of *pruning rules* to increase the interestingness of the inference set by eliminating redundant inferences. These rules, presented below, introduce two concepts: subsumption and weak rules. Where subsumption refers to relative element occurrence. For example, given that element a subsumes element b then b has a high co-occurrence with a, or $a \Rightarrow b\,\gamma(x\%)$, where x is high. Weak rules are those whose validity may be questioned due to the presence of alternative causes.

- Given $a \Rightarrow C$ and $a, b \Rightarrow C$ occur with similar confidence, then $a, b \Rightarrow C$ is redundant as b is not a significant contributor or cause of the inference.
- Given $a \Rightarrow C$ and $b \Rightarrow C$ occur with similar confidence, then $b \Rightarrow C$ is redundant, if b subsumes a but a does not subsume b.
- Given $a \Rightarrow C$ and $b \Rightarrow C$ occur with similar confidence, then they are *weak* rules if a and b subsumes each other.
- Given $A \Rightarrow c$ and $A \Rightarrow c, d$, then $A \Rightarrow c$ is redundant as it is represented within $A \Rightarrow c, d$.
- Given $A \Rightarrow c$ and $A \Rightarrow d$ and c subsumes d then $A \Rightarrow c$ is redundant as the subsumed consequent is logically stronger.

Liu *et al.* (2000) propose the filtering of discovered inferences to remove expected inferences through the iterative specification of three constraint levels: general impressions, reasonably precise concepts and precise knowledge. Each of these in an increasing order of precision specify a class of expected inference, the participants of which can be removed from consideration. General Impressions (GI) represent vague user intuition about a relationship between a set of elements or higher level concepts. The precision is increased in Reasonably Precise Concepts (RPC) with the specification of antecedent and consequent classes, while Precise Knowledge (PK) allows the user to specify a specific inference and its strength. The constraints are then applied to each discovered inference and assigned a conformance weight, which is the degree to which the inference conforms to the specified constraint, providing a measure of unexpectedness. This work

was subsequently extended through the inclusion of fuzzy constraints (Liu, Hsu, Mun & Lee 1999).

Saha (1999) proposes a technique by which large families of inferences can be quickly eliminated through the user classification of *seed rules*, where a seed rule is a simple inference that forms the basis of many other existent inferences. Through a user classification of seed rules, a knowledge base containing user beliefs is developed this allows the elimination of families of inferences deemed expected, based upon the classification of their seed rules. By iteratively presenting the largest seed rules for the user to classify in regard to their interestingness, families of uninteresting rules can be eliminated from consideration where the *largest* seed rules refer to those seed rules that represent the largest families of inferences. The author presents evidence indicating that, over five iterations of classification up to half of the inferences can be eliminated from consideration.

5.3 Summary

This chapter provides a review of techniques used to constrain the exploration of a search space lattice. The discussion presents constraints as being derived from two classes: 1) functional or 2) domain based constraints and discusses the stage of constraint inclusion within the knowledge discovery process.

This discussion identifies the different types of constraint that need to be incorporated within a guided mining environment, information subsequently used in the development of a guided mining environment and proof-of concept tool (Chapter 7 & 8). This discussion is continued in the next chapter (Chapter 6) by discussing techniques that allow for the refinement of constraints during a knowledge discovery session.

Chapter 6

Constraint Refinement Review

The inclusion of constraints within data analysis is fundamental to maintaining exploration tractability. As the nature of a dataset's inferences are initially unknown, initial constraint specification, is at best based upon an educated guess. Therefore as inferences, or patterns, are elicited from the dataset and presented to the user, the user's knowledge base evolves in respect to understanding the patterns within the dataset. As a result the user can subsequently refine constraints, based upon this new knowledge, to focus upon areas of interest and thereby improve the quality of results.

The typical inclusion of refinement results in an iterative analysis process involves the successive refinement of constraints, until the user is satisfied with the quality of inferences generated. However a new area of research, and the focus of this thesis, is the dynamic refinement of constraints *during* the analysis process.

Dynamic constraint adjustment therefore provides a mechanism allowing the user to steer subsequent analysis into areas of exploration interest. To accommodate such functionality, an interactive analysis environment is required that provides users with the dynamic presentation of intermediate results. This gives the users the opportunity to react to the current state of analysis and through constraint adjustment guide subsequent processing. The inclusion of guidance also leads to a greater level of user trust in the mining results, as the user is more involved with the process (Section I.3) and hence better understands the process leading to the results, thereby reducing the unknown processing element.

This chapter provides a review of the current state of constraint refinement within knowledge discovery and provides a foundation for subsequent chapter contributions. The first section (Section 6.1) presents several techniques proposed to facilitate iterative analysis through the use of previously generated information. The remaining sections review the current techniques that enable user guidance within the field of knowledge discovery, leading into a detailed discussion on this thesis's contribution to guided analysis in Chapter 7. To this extent, the following review looks at guided analysis not only in regard to association mining but also within the knowledge discovery process (Section 6.2) and the main facets of exploratory mining: clustering (Section cim:gc), classification (Section cim:gcl) and association mining (Section cim:ga).

6.1 Iterative constraint refinement

Nag *et al.* (1999) propose the use of a knowledge cache to reuse previously discovered inferences during subsequent analysis, effectively reducing analysis iteration time. While several caching strategies are proposed the only constraint incorporated is *support*, while solutions for the inclusion of domain based constraints are not presented. This work was extended by using condensed representations, such as closed sets, to reduce cache size and improve processing in highly correlated datasets (Jeudy & Boulicaut 2002). Other iterative optimisations are based upon the storage of previous results (Raghavan & Hafez 2000, Ortega, Chakrabarti & Mehrotra 2003).

Cong & Liu (2002) propose using the boundary of the valid elementset lattice to summarise the useful information from the previous analysis instance. Given the inclusion of *support* the boundary provides the set of maximal valid elementsets. Subsequent refinement of *support* therefore results in either a tightening or relaxation of the lattice boundary. If tightened then the new valid elementsets can be easily derived from the existing boundary set, while if relaxed, all previous results are valid and subsequent analysis can begin at the lattice boundary. This lattice boundary technique is constrained to simple reflexive constraints due to the use of the lattice boundary and the requirement for global constraint non-monotonicity.

Bolton and Adams (2003) propose a refinement technique based upon an evolving baseline model (or background model) that represents what the user *expects* in the data. By iteratively analysing and appending uninteresting inferences to the baseline model, the process approaches an optimal set of interesting inferences, as the baseline model of expected inferences becomes complete. Elicited inferences of interest can also be appended to the baseline model to avoid repeat discoveries. Any initial baseline model proposed by the user is rarely complete or correct and requires subsequent iterative analysis and refinement to become so. The baseline framework is not dependent upon a single methodology, the authors suggesting its use in both modelling and algorithm pattern discovery techniques, however, the examples presented focus upon statistical modelling techniques.

A similar knowledge refinement strategy is presented by Padmanabhan & Tuzhilin (2002) in which unexpected inferences are reconciled with the prior specified domain knowledge base. Two reconciliation algorithms are proposed that extend previous work upon the discovery of unexpected patterns (Padmanabhan & Tuzhilin 2000), through the inclusion of a set of refinement properties, these are outlined below.

- **Convergence** of a belief system to a fixed point. After a finite number of iterations no unexpected patterns are discovered in relation to a refined system of beliefs.
- **Consistency** of all beliefs at any iteration of the belief refinement process.
- **Path Independence** ensures that the order in which the selected patterns are incorporated into the belief system does not affect the refined system of beliefs
- **Minimality** requires that the refinement strategy creates a minimal set of beliefs.
- **Monotonicity** guarantees that once an unexpected pattern is incorporated into the belief system and becomes expected, it will not subsequently reappear as unexpected.

Goethals and Bussche (2000) propose an *online filtering* technique that attempts to optimise integrated filtering by reusing previous analysis results, where

integrated filtering is the inclusion of the constraints in the generation of valid elementsets. Goethals and Bussche (2000) also discuss the trade-off between post-process and integrated filtering. They show that given an iterative integrated environment, there comes a point at which naive analysis and subsequent post-process filtering becomes more efficient than iterative integrated filtering, even with reuse, due to the extra processing involved for each new query in an integrated environment.

6.2 Guided Knowledge Discovery

Guidance within the knowledge discovery process provides a framework mechanisms to facilitate or control the mining session. This guidance occurs at a coarse level facilitating the selection of tools (collection, pre-processing, mining and presentation) and the piping of data between them, however it does not address the incorporation of guidance within the mining process.

Wrobel *et al.* (1996) propose an architecture allowing the guidance of hypothesis exploration, incorporating parallelism to allow different subtasks (hypothesis explorations) to be carried out upon separate processors. The system is based upon a Search Space Manager (SSM) that persistently stores the search state of different hypothesis. Through SSM the user can analyse current subtasks, instigating new subtasks based upon existing hypothesis refinement. Furthermore, through the maintenance of a hypothesis history any previously investigated hypothesis can be refined and re-processed, in essence providing backtracking functionality within the knowledge discovery framework.

Aide (Amant & Cohen 1997), is an assistant based architecture in the related field of Exploratory Data Analysis (EDA) that incrementally explores a dataset (D) guided by user directives and its own evaluation of the data. The area of EDA provides statistical tools through which patterns in data may be extracted and examined in detail. *Aide* facilitates EDA exploration by representing the EDA search space as a plan of operations, which, based upon intermediate data analysis results, guides EDA operation selection to select algorithms that discover more extensive descriptions of the underlying data.

User inclusion within the system is at an operational level. During processing the user can override the next operation selected in the plan, thereby directing further processing and enabling the inclusion of domain-knowledge through operation selection. Based upon user intervention, the system subsequently searches through its plan-base to propose and execute subsequent operations based upon the user's guidance and the specified task at hand. Similar frameworks are proposed within knowledge discovery Engels *et al.* (1997) and Jensen *et al.* (1999) that assist the user in decomposing a knowledge discovery task by proposing suitable tools within each stage, which may be overridden, while handling subtask dependencies.

Extending this work Livingston *et al.* (2001a) propose an autonomous knowledge discovery system that decides upon the next task to perform based upon an agenda and justification based framework. The authors propose that an autonomous system improves inference quality through the removal of human error, the use of multiple strategies and multi-hypothesis analysis. The system is comprised of an initial agenda of tasks, specified by the user, that are prioritised according to their *plausibility*, where a task's plausibility is calculated using a function of the *strength* of a *reason* for performing the task and an estimate of the *interest* of the elements or elementsets that the task operates on. This is presented in Example 13, given that T is the task being considered and e is an elementset within the domain of T.

Example 13 $Plausibility(T) = (\Sigma Strength(Reason_T)) * (\Sigma Interest(e_T))$

The reasons for undertaking a specific task is based upon qualitative representations of the *support* and fit of that task to the knowledge discovery task at hand, providing a means by which the framework can reason about task appropriateness. During the processing of tasks, new tasks can be appended to the agenda and new reasons for existing agenda tasks can be discovered, altering plausibility and hence agenda ordering. The result is the autonomous processing of the agenda according to a set of user defined heuristics.

The heuristics used to control agenda processing and result interestingness are specified before analysis (Livingston, Rosenberg & Buchanan 2001*b*). Given

that all tasks use a generic set of interestingness heuristics, the result set cannot be optimal for all user defined tasks, as what is interesting can significantly change between tasks. Although the user may propose good general measures, they cannot be optimal for each task and therefore the processing and level of result interestingness suffer. Furthermore the automatic insertion of new tasks, based upon the initially specified heuristics and statistical interpretation, cannot accurately account for what the user requires and therefore irrelevant processing may occur.

This autonomous framework is potentially beneficial for the general exploration of a hypothesis, as it considers many possibilities and does not require the presence of a user for lengthy periods of time. However it is not an efficient means of discovering an optimal set of interesting results as the heuristics incorporated are general and cannot be optimal for all tasks.

6.3 Guided Clustering

The Mitsubishi Electronic Research Laboratory (MERL) has investigated the use of interaction to solve optimisation problems in the fields of capacitated vehicle routing with time windows Anderson *et al.* (2000) and network partitioning Lesh *et al.* (2000). This work centres upon the *Human Guided Simple Search* or *HuGSS* paradigm that improves the effectiveness of a relatively simple search algorithm by allowing users to steer the search process interactively. The interactive capabilities provided by this include the ability to escape local minima through manual editing and the ability to focus the search into areas of promise.

The problem in capacitated vehicle routing involves the optimisation of goods delivery to a group of customers with the least amount of trucks, whilst minimising the distance travelled by each truck. *HuGSS* addresses this problem by using either a greedy or steep-descent clustering algorithm to determine the number of trucks and the routes they should take. The user specifies the number of steps in a search invocation, which effectively controls how many automated allocation moves the computer can make before presenting a set of intermediate results to the user. This allows the user an opportunity to insert a guidance primitive for the next invocation of automated allocation.

(a) Initial Solution

(b) User movement of customer from route A to B

(c) Moving customer result in initial solution

(d) Re-optimisation based upon new constraints

Figure 6.1: Repercussions of user-movement (Rabejij 2001)

The user guides the route allocation process, escaping local minima by manually assigning customers to routes, effectively changing the element's cluster. This automatically invokes a route optimisation algorithm upon the effected routes only, ensuring that under these new constraints they still provide the best possible solution as illustrated in Figure 6.1. The user can also refine clustering constraints such as customer priority and the number of clusters before invocating another sequence of automated route allocations. Figure 6.2 shows a snapshot of the completed routing allocation.

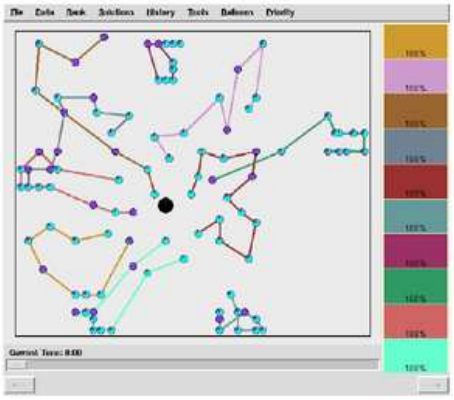


Figure 6.2: Routing interface snapshot (Anderson *et al.* 2000)

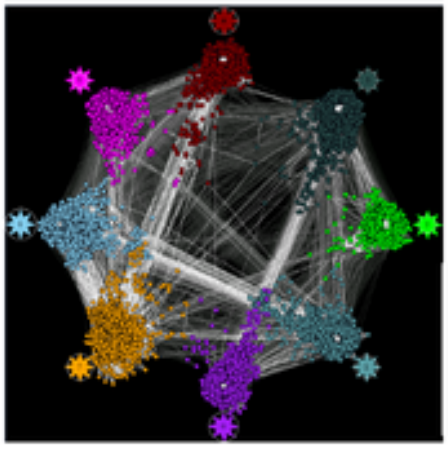


Figure 6.3: Network Partitioning presentation (Lesh *et al.* 2000)

The MERL group has also applied the *HuGSS* paradigm to the area of *k-way* network partitioning, an NP-hard problem arising in VLSI design and elsewhere. This required the development of a different set of presentation techniques to visualise the required relevant aspects (as shown in Figure 6.3). This consequently

led to the development of a new set of interaction mappings which effectively provide the same types of interactive guidance as the route discovery application (Lesh, Marks & Patrignani 2000). This research by MERL is the most significant published work in this field to date. Although other work such as Nascimento & Eades (2001) allows the user to interactively change the cluster number and size, the critical element in MERL's research is the ability to guide cluster membership.

Aggarwal (2001) proposes an interactive clustering architecture that identifies clusters within high dimensional space through interactive subspace analysis. A need for interactivity is motivated by the apparent variations that may exist between subspaces in regard to noise, cluster density and the difficulty that static measures have in effectively distinguishing clusters across subspaces. By incorporating the user in subspace selection and cluster specification, less subspaces are analysed and the quality of subspace clustering is increased. The process combines active subspace exploration with user interaction to reduce the number of possibilities considered during subspace exploration. For each selected subspace, clusters are algorithmically proposed and presented to the user for validation and possible manipulation. Once subspace analysis is complete, the higher dimensional clusters are automatically derived from the subspace clusters identified. A similar technique, presented by Hinneburg *et al.* (1999a), varies in respect to metrics and subspace selection. The work is further extended to provide similar optimisations for high dimensional nearest neighbour search in Aggarwal (2002).

6.4 Guided Classification

Classification involves both descriptive (or exploratory) and predictive processing stages where guidance is incorporated into the descriptive aspect of constructing the decision tree model. The construction of decision trees lends itself to guidance inclusion as they are iterative, applying the same process to each step in the decision tree's construction. This provides consistent feedback points, where given intermediate presentations the user can guide subsequent construction.

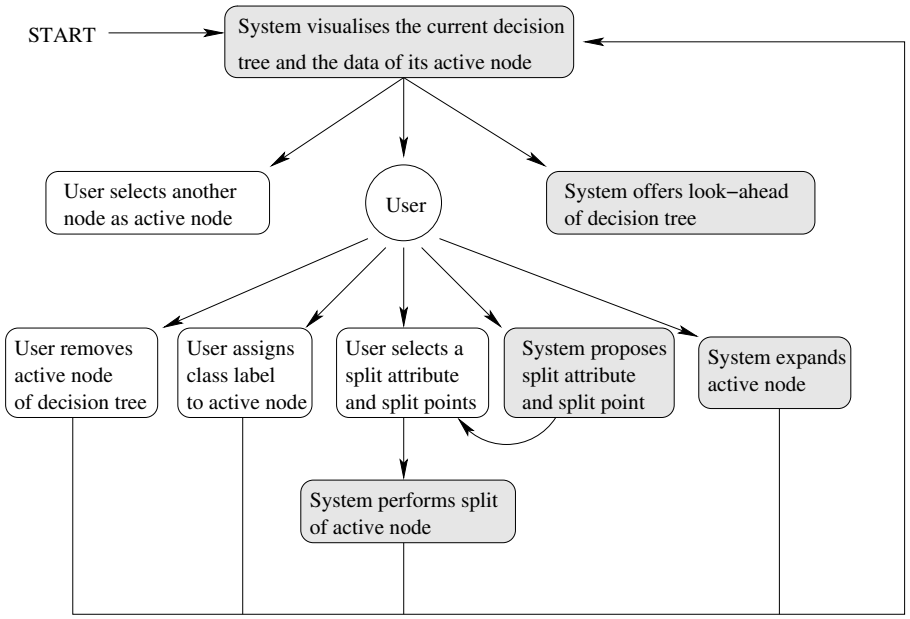


Figure 6.4: Perception-based classification process (Ankerst *et al.* 2000)

Perception-Based Classification (Ankerst, Ester & Kriegel 2000) uses a cooperation based framework, allowing the user to guide and automate construction as necessary. The interactive process as illustrated in Figure 6.4 indicates the automated steps (*shaded*) and user interaction. It can also be seen that the user has five options at each intermediate presentation, which can be classed into local and global interaction with respect to their area of effect.

Local interaction includes *split*, *propose split* and *look-ahead*. The first, *split*, involves manual interaction while the others instigate automated processing, either to propose the next split, *propose split*, or provide a *look-ahead* to the result of specific splits upon further construction. The global interactions effect global decision tree structure and consist of active node removal and labelling or active node expansion. The manual tasks of node removal and labelling are orthogonal in that removal undoes prior processing. This allows for other splits to be explored, while class labelling completes processing of that particular subtree. Active node expansion instigates the automatic classification of that particular subtree. Processing is complete when either all leaf nodes have had class labels associated with them or when the user is satisfied with result quality, classification accuracy and tree size. Through this level of interaction the authors claim that a

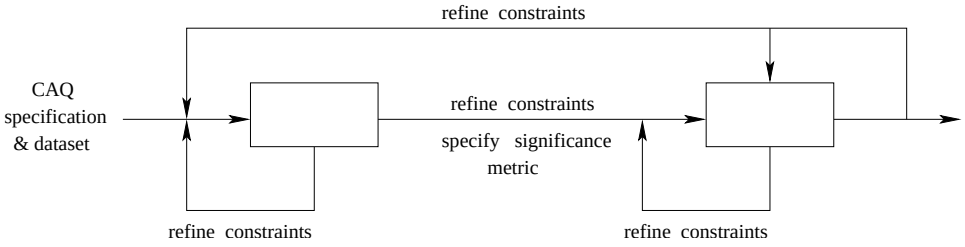


Figure 6.5: CAP process (Ng *et al.* 1998)

better classification solution is achievable through user specification of splits and the ability to backtrack, when a subtree's classification is suboptimal.

A similar technique (CIAD) is proposed by Poulet (2002) that appears less flexible in its interaction capabilities and uses interactive scatter-plot matrices as the cooperative interface.

6.5 Guided Association Mining

Ng *et al.* (1998) propose an open architecture and a rich set of constraints (Section 5.1) that enable coarse-grained user involvement within the association mining process. The proposed architecture enables constraint refinement, within analysis, between elementset derivation and inference derivation through the inclusion of user feedback. Given analysis based upon an initial constraint set, the user is presented with the set of valid elementsets. If the user is satisfied with the quality of V they can instigate inference derivation, or refine constraints and re-generate V. The architecture also enables the inclusion of inference constraints during inference derivation and their possible refinement. Given an unsatisfactory inference set the user is able to refine either elementset or inference constraints and re-instigate the respective analysis stage (illustrated in Figure 6.5). This architecture has been embedded within On-Line Analytical Mining that includes OLAP functionality to focus analysis upon data warehouse subsets (Han, Lakshmanan & Ng 1999).

Brin and Page (1998) proposed Dynamic Data Mining (DDM) that attempts to produce more interesting inferences by foregoing traditional *support* based algorithms that use a single deterministic run and instead use a technique that

incrementally explores more of the search space. This is accomplished through the inclusion of two new heuristics: *Weight*, which can be dynamically refined and the *Heavy Edge Property* which guides the exploration process. Instead of the mining process being a single deterministic run, producing a well defined set of elementsets and subsequently inferences, DDM invokes an analysis process that iteratively generates valid elementsets (V) of improving quality. Based upon the Dynamic Itemset Counting algorithm (DIC) (Brin, Motwani, Ullman & Tsur 1997), DDM takes advantage of intermediate counts to form an estimate of an elementset's occurrence or *weight*, resulting in an intermediate presentation. The user is then able to dynamically adjust the individual element *weights*, in effect prioritising them. Such user interaction is indirect and textual, with refinement occurring through an associated text box, whereby the user adjusts global analysis parameters and individual element weights via a simple language. This technique importantly enables a simple type of user focus through element prioritisation that facilitates the improvement of resul t quality. This incorporation of user focus is extended in GAM (presented in Chapter 8).

Hidber (1999) proposed *CARMA* that allows the dynamic adjustment of *support* and *confidence* thresholds during analysis. The algorithm continuously constructs a candidate elementset lattice during an initial scan of a dataset (D), each member of which has a deterministic lower and upper *support* bound. Upon completion the lattice is comprised of a superset of all elementsets in relation to some *minsup*, depending upon *support* adjustment during analysis. A second scan is then used to remove all elementsets from the lattice whose upper support bound is below the last user specified threshold and to update the precise frequency of all remaining elementsets.

Given V_{e-1} as the lattice representing the discovered elementsets in objects up to $e-1$, V_e is constructed by appending e using three steps: increment, insert and prune (presented on the next page). An intermediate presentation is made after each object's inclusion within V enabling opportunity for the dynamic adjustment of *support* threshold. This results in the development of a support sequence, used to determine the upper support bound on subsequent elementsets.

- **Increment** all count's of $v \in V | v \in O_e$ are incremented.
- **Insert** all s into V, given that $s \subset O_e$ and $s \notin V_{e-1}$ and s is valid in respect to the current support bounds. The lower bound is defined by the count of s, while the upper bound is based upon the function *maximised*, which is based upon the current and previous *support* thresholds defined by the user.
- **Prune** all v, where $|v| \geq 2$ and the upper support bound is less than the current user defined threshold.

6.6 Summary

This chapter provides a discussion and review upon constraint refinement within knowledge discovery sessions, which can be classed into iterative and interactive refinement. The literature review conducted indicates (through a lack of published material) the novelty of this promising area of research.

This discussion provides a basis for justification of our contributions in the area of guided association mining and guided knowledge discovery in general. The following two chapters present this thesis's contributions in this regard. The next chapter (Chapter 7) presents a generic guided knowledge discovery framework and Chapter 8 presents a guided association mining system (GAM) built upon this framework. Furthermore, a discussion is provided within the conclusion of this thesis (Part IV), that compares this thesis's contributions against the current interactive techniques presented within the chapter.

Chapter 7

Guided Association Mining Architecture

The effective inclusion of guidance within analysis requires an extension to the current accepted knowledge discovery architecture presented in Figure I.2. The provision of guidance requires a dynamic association between the analysis and presentation stages, not possible given the current architecture, in which the stages are independent and sequential. This chapter presents a guided knowledge discovery architecture, that although targeted at association mining, could be adapted to other exploratory tasks such as clustering or classification.

Within the current architecture the user is able to interact with the knowledge discovery system between stages, enabling user selection of appropriate tools, parameter adjustment and possibly input data adjustment. User interaction within specific stages is tool specific, however the user, given current technologies, is generally integrated within collection, pre-processing and presentation tools. In contrast, the analysis stage is currently an automated and opaque process, a "black box", that explores a dataset according to a specified constraint set, presenting upon completion a set of valid inferences.

Association mining is an exploratory task, whereby the user is in effect *fishing for interesting information* without an established hypothesis as to what the *catch of the day* will be. The initial constraint specification is therefore at best an educated guess, based upon the user's current understanding of the domain

including: the dataset, the domain, past experience and the current task. Because of the analysis "black box", the construction of a quality result set becomes an iterative process, requiring successive analysis and presentation iterations until an inference set of satisfactory quality is obtained.

It is therefore the proposal of this thesis that by opening the process and allowing the dynamic refinement of constraints during analysis, that a higher quality of inferences will be obtained through the maintenance of user-computer synergy. To effect this the analysis process must become transparent and interactive rather than opaque and automated.

By making analysis transparent, through the dynamic presentation of the evolving inference model, the user's knowledge base can also evolve during the analysis, enabling the discovery of interesting inferences before analysis is complete. By also including premature termination functionality that allows the user to prematurely terminate analysis, analysis can be reduced once all the inferences of interest have been discovered. Such an architecture will be referred to as a *dynamic architecture* throughout this chapter.

Although an optimisation over the classic architecture, through the use of transparency and premature termination, a dynamic architecture still uses batch processing in the sense that it is automated and statically constrained. By further extending the architecture and providing feedback points through which the user can dynamically adjust constraints during analysis, the user can guide search space exploration. The level of synergy that this interaction maintains is determined by the level of possible feedback granularity within the analysis. Where the term "possible" relates to the goal of dynamic feedback, so that the user should be able to interrupt automated analysis as required, where the lag between interrupt and interaction reflects the level of interaction granularity.

The inclusion of guidance therefore requires the tight coupling of the previously independent analysis and presentation stages. This functionality could be provided at the tool level through the provision of a component providing both analysis and presentation functionality. However, it is a goal of this thesis to present a guided association mining architectural design, through which guidance enabled analysis and presentation components can be "plugged in".

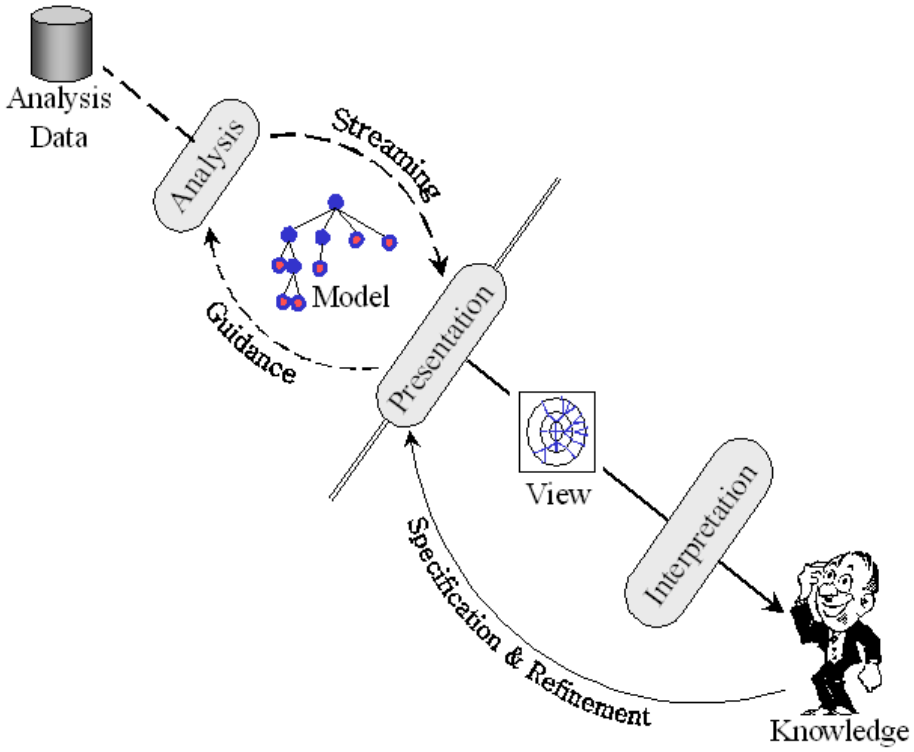


Figure 7.1: Guided Knowledge Discovery Process

Such architectural extension is achieved by strengthening the coupling between the analysis and presentation stages, providing real-time streaming of analysis results to the presentation tools or *views*. The user can then interpret and guide further analysis through view interaction, as shown in Figure 7.1. Although this could be achieved through the merging of the analysis and presentation stages, this would require specialised guidance tools. A more promising approach is the maintenance of stage independence and the increase of coupling through the specification of an interface to manage interactive functionality between the two stages, facilitating "pluggable" components.

Figures 7.2 and 7.3 present an example that highlights the difference between classic and guided analysis, through illustration of their respective analysis exploration space. These models are represented as trees, in which the node colour represents its function.

Figure 7.2 presents classic analysis in which the terminal results are presented upon completion. Figure 7.3 contains three images illustrating inference model

evolution during analysis. The images represent initial, intermediate and complete inference model presentations, where the intermediate presentation typifies a dynamic presentation architecture. The number of intermediate presentations within a system is arbitrary, dependent upon the underlying analysis algorithm and the inference model.

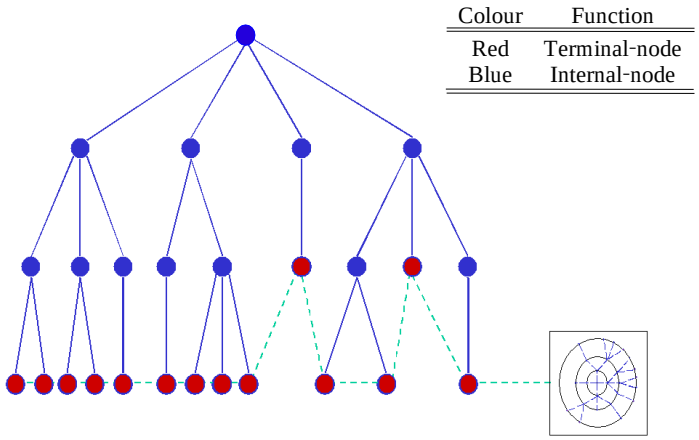
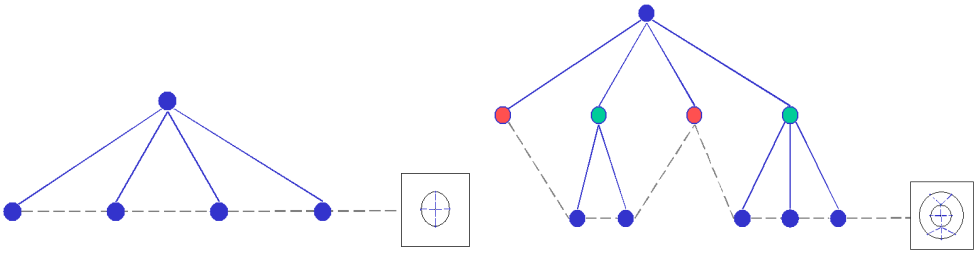


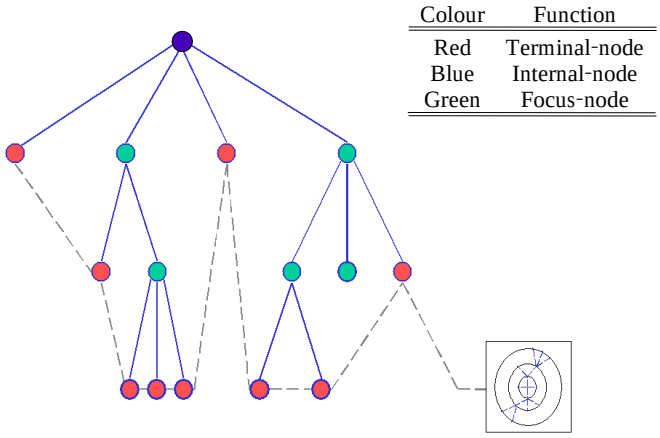
Figure 7.2: Classic Analysis

Figure 7.3(a), presents an initial model, based upon the valid dataset elements (E) given a set of initial constraints, before any combinatorics are considered. Through interaction with this initial presentation the user can focus subsequent exploration. This is apparent in Figure 7.3(b), an intermediate presentation, as exploration has only been undertaken in specified areas of focus. By guiding analysis through intermediate presentation interaction, the final model, Figure 7.3(c) is constructed.

By comparing a presentation of the complete models in Figure 7.4, some intended advantages of guided analysis are apparent. The presentation tool used within the figure is a two-dimensional pre-cursor to CARV designed during candidature, that clearly illustrates the example model differences for this example. Through the ability of the user to focus exploration, only a portion of the entire search space has been explored, resulting in reduced analysis time and a reduced result set. More importantly however is that through the maintenance



(a) Initial presentation (b) Intermediate presentation

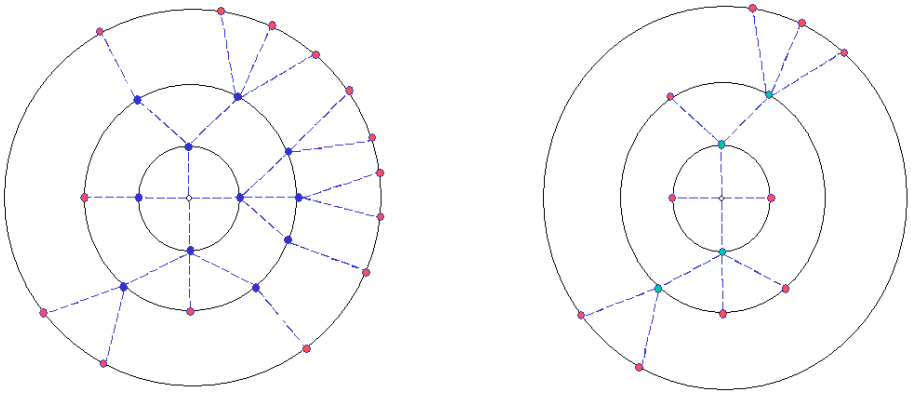


(c) Completed presentation

Figure 7.3: Guided Analysis

of user-computer synergy, the resultant inferences are of higher quality (or of greater interest), further optimising mining by reducing or alleviating iterative refinement. Making analysis transparent and allowing user guidance improves knowledge discovery by reducing processing time and improving result quality.

This chapter discusses guided mining and its advantages over current system architectures. The following section discusses constraint adjustment in three different architectures: classic, dynamic presentation and guided. Chapter 8 proposes a guided association mining system and process architecture.



(a) Batch model (b) Guided model

Figure 7.4: Comparison of batch vs guided model

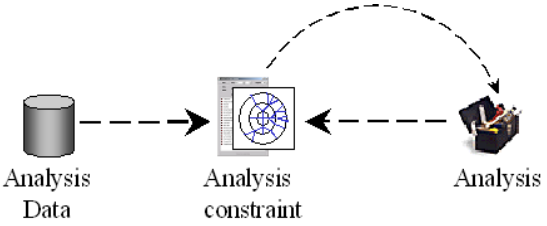
7.1 Analysis Interaction

Analysis interaction, or constraint specification relevant to the analysis stage within classic architectures, occurs either before or after analysis due to its automated processing. A guided architecture will enable interaction within the analysis process itself, through the dynamic refinement of constraints. The difference between the two interaction models is presented in Figure 7.5. In practice, post-analysis interaction or constraint refinement is integrated with the presentation tool, enabling filtering of the resultant inference model, while pre-analysis interaction allows the specification of constraints and their refinement upon subsequent analysis iteration. Within a guided architecture a single interaction point is evident due to the merging of the analysis and presentation stages.

As previously stated, analysis interaction provides mechanisms through which constraints upon the analysis process can be specified and refined. Previous discussion (Chapter 5) presented two classes of association mining constraint: functional and domain based with functional constraints focusing on algorithm heuristics, and domain constraints on concepts such as elements and elementsets.

In addition to these algorithmic constraints, the inclusion of analysis transparency enables the inclusion of process constraints. That is, given that algorithmic constraints refer to exploration extent, process constraints deal with the nature of the exploration, including automation extent.

(a) Classic Architecture



(b) Guided Architecture

Figure 7.5: Stages of analysis constraint refinement

The following sections (Section 7.1.1 . . . 7.1.3) discuss the extent of interaction available for both algorithmic and process constraints within the three architectures: classic, dynamic and guided. Finally a summarisation of guided interaction is presented ion Section 7.2.

7.1.1 Algorithmic Interaction

Algorithmic interaction encapsulates domain and functional constraint refinement. Domain based constraints enable the user to effect a concept's (either elements or elementsets) characteristics and participation within analysis. Functional constraints allow the user to incorporate quality heuristics based upon mathematical operators or functions to improve the quality of discovered inferences. There are three forms of domain constraint interaction: filtering, adjustment and focus, while functional constraint interaction takes the single form of adjustment (presented on the next page).

Algorithmic constraint refinement is similar within both classic and dynamic architectures as they are both automated. While dynamic architectures incorporate transparency and process interruption, the extent of interaction is constrained by the automated analysis, thereby reducing dynamic system interaction capability to that of classic systems. The following discussion therefore implicitly

groups these two architectures together, unless otherwise specified. The advantage of dynamic systems over classic systems, in regard to interaction, lies in their ability to incorporate process constraints.

- **Domain Filtering** allows the positive or negative constraint of concept participation within analysis.
- **Domain Adjustment** refers to concept characteristic manipulation. For example, the adjustment of a concept's weight in a quantitative analysis algorithm.
- **Domain Focus** is concerned with concept prioritisation, by which the user can alter the order of processing to explore a particular concept before, or perhaps after, other concepts.
- **Functional Adjustment** refers to the manipulation of statistical thresholds incorporated within analysis.

The nominated architectures can potentially incorporate all forms of algorithmic constraint refinement, however it is the effect of this refinement that differs between classic and guided architectures. Due to automated analysis, classic refinement results in complete model reconstruction, the exception being constraint tightening, which can require model re-analysis only, while the interactive, rather then iterative, nature of a guided architecture enables refinement to be incorporated within the same analysis process. This is however, dependent upon the effect of the interaction upon the search space and the user goals.

The discussion of algorithmic constraint refinement through user interaction can be broken into three classes: exploration refinement, prioritisation and concept adjustment.

Exploration Refinement

Exploration refinement encapsulates both concept filtering and heuristic adjustment, arguably the most common forms of refinement, as they effect the extent of the search space. The subsequent processing required to ensure inference model accuracy in regard to the refined constraints is therefore dependent upon the refinement's effect on the search space.

Given the refinement of a reflexive constraint, the result will be a global tightening or relaxation of the lattice, allowing for model update optimisations without requiring a re-generation of the model. However if the constraint is not reflexive, then its effect upon the lattice cannot be predicted. Therefore as its effect is not directional, inference model reconstruction to the current exploration extent will be required to ensure accuracy in regard to the refined constraint.

While all domain constraints are by their participatory nature, reflexive, not all functional constraints are. The inclusion of non-reflexive constraints within analysis is infrequent as they are of little use. It is therefore assumed for this discussion, that all exploratory constraints incorporated are reflexive. Furthermore, given complex constraint refinement (in which multiple constraints are refined) global reflexivity can still be ensured by treating each constraint independently.

Given a classic architecture lattice relaxation requires re-analysis due to the possible introduction of new valid concepts. Lattice tightening however may not require re-analysis, dependant upon the global reflexivity of the incorporated constraints, as all pertinent information is contained within the current lattice (inference model).

However, this leads to the point that tightening is generally unwarranted within classic architectures, because the information is already present in the inference model. By providing filtering and adjustment functionality within the presentation tool, the same effect can be achieved without any re-analysis. This is not the case for relaxation as new information will result from lattice relaxation and therefore update analysis is required.

Dynamic architectures provide an advantage in this case over classic architectures, as the user is able to prematurely terminate and re-instigate analysis based upon a refined set of constraints. Therefore lattice tightening may become feasible depending upon how early in analysis the tightening occurs and the extent of its effect upon exploration. The larger the exploration effect, the later its inclusion is warranted. Given an optimal algorithm, tightening would then require analysis of the partially generated inference model, removing concepts as necessary, then automated analysis with dynamic presentation would proceed, based upon the tightened partial model.

Based upon this explanation of dynamic tightening, heuristic adjustment and positive concept filtering may be warranted, as their effect upon exploration can be significant. However, although further investigation is required, it appears unlikely that refinement based upon negative concept filtering or concept exclusion is warranted due to the limited effect they have upon exploration.

Furthermore while tightening is generally based upon the user's evolving knowledge base, relaxation cannot be, as concepts that have not yet been presented cannot be incorporated. The decision to relax is based therefore not upon semantic factors resulting from presented inferences but upon concerns that inferences of interest may have been missed.

While the above discussion considers the complex case of updating the existing model to reflect refinement before proceeding with further analysis, it is possible to reduce update processing by enabling a flexible re-start process constraint. This constraint restricts the update of the existing model by indicating where the refinement is to restart. For example, given a tightening of *support* during the BFT construction of the fourth level of an inference model using classic candidate generation analysis, refinement can be incorporated from the current or any previous point of model construction, in this case constrained to specification of levels within the lattice. If included from the current analysis state then no model updating is required and analysis resumes from the current state. In contrast, the user could request inclusion from level-two, whereby level-two downwards would be refined before analysis recommenced. While this feature is practical for heuristic refinement such as *support*, domain relaxation must always update the entire current model, due to its exclusion at the top-level.

Prioritisation

The usefulness of concept focus or prioritisation within a classic architecture is arguable, as because of automated analysis, it can be reduced to pre-analysis concept filtering. Focus can be considered as temporal concept filtering, as the analysis is temporarily constrained to the positive filtering of a concept. There is no significant difference, in a classic architecture, between focus and two analysis iterations, with the concept first positively and then negatively constrained. Dy-

namic architectures through the use of intermediate presentations can accomplish this in a single analysis iteration. However the main limitation of both systems is that focus must be defined before analysis begins and cannot be dynamically incorporated as in a guided architecture.

Focus is pragmatically specific to guided architectures as through dynamic refinement the user has the ability to focus upon different concepts as the analysis proceeds. Although of little use within automated analysis architectures, it is the ability within a guided system to dynamically change focus as new areas of interest are revealed to the user that are significant. While focus can still be seen theoretically as a subclass of concept filtering, it is singled out because of implementation differences, but more importantly due to its significant contribution toward validating guided architectures.

Concept Alteration

Concept alteration is valid where analysis is based upon concept characteristics through the inclusion of functional constraints upon characteristics of interest. Included for the purpose of completeness, this form of interaction will result in functional constraint refinement. The difference is that the adjustment target is not a functional constraint but a concept specific parameter, used within a functional constraint during analysis. For example an analysis algorithm may use user specified weights to influence concept importance, therefore the user can change the weighting of a concept to effect subsequent exploration (as in Fule & Roddick 2004). The functional inclusion of characteristic refinement within analysis will be as for functional constraint refinement, which is presented in the next section.

7.1.2 Process Interaction

Constraints on the analysis process allow the user to control the extent and pace of automated processing. Although not possible in classic systems due to its "black box" nature, process constraints can be incorporated within both dynamic and guided systems, as they do not effect exploration extent, but the manner in

which exploration is automated. The purpose of process constraint is to facilitate user interpretation of intermediate results and allow fine level control of further analysis. Process constraints fall into four classes: interrupt, pace, step and restart (presented below).

- **Interrupt** allows the user to pause analysis, giving the user an opportunity to inspect current inference model state. Given a guided architecture, this allows the user to instigate refinement without analysis overrunning the target of their refinement before implementation.
- **Pace** allows the user to control analysis speed.
- **Step** allows the user to incorporate an automatic interrupt after a specific amount of analysis.
- **Restart** provides the ability to re-instigate analysis from a previous model state. By default the process restarts from the current inference model state.

While pace, step and interrupt are valid for both applicable architectures, restart is not valid within dynamic architectures as the same model will always be built due to static constraint inclusion.

7.1.3 Guided Architecture Interaction

A guided architecture providing Dynamic Constraint Refinement (DCR) in a transparent analysis environment enables interaction capabilities previously impossible due to architectural constraints. This is apparent through features such as focus and restart that are particular to an interactive environment, and the ability to implement constraint refinement with less overheads than previous architectures. A summary of interactive functionality viable within the different architectures is presented in Table 7.1.

Guidance provides a richer set of functions than presented in Table 7.1 - it provides an interactive analysis environment through which emergent capabilities based upon the described functionality become obvious. Consider the scenario

Constraint Class	Interaction Task	Classic		Dynamic		Guidance
		Pre	Post	Pre	Post	
Domain	concept filtering	✓	✓	✓	✓	✓
	concept adjustment	✓		✓		✓
	concept focus	✓		✓		✓
Functional	heuristic adjustment	✓		✓		✓
	heuristic filtering		✓		✓	
Process	pace adjustment			✓		✓
	step adjustment			✓		✓
	flexible restart					✓
	analysis interrupt			✓		✓

Table 7.1: Interaction type summarisation

for both classic and guided architectures, presented on the next page, in which the power of guidance over automated analysis architectures in facilitating the production of quality results during a single analysis iteration is illustrated. Given that quality is measured by the number of interesting inferences over the total number of inferences.

The result of including guidance within analysis is the production of a quality inference set in less time than classic architectures, as guidance reduces processing overheads in relation to the refinement of constraints. This is the advantage of guided association mining in a nutshell. All types of interaction or guidance can be mimicked in a classic architecture, but due to the overheads incurred they become infeasible. This is due to automated analysis and the requirement for analysis iteration to implement constraint refinement.

There are two disadvantages to the implementation of a guided architecture: processing complexity and scalability. The additional processing complexity, due to the inclusion of guidance functionality within the mining architecture will slow analysis where no iteration is required. However, given the goal of obtaining a quality inference set and the dynamic constraint of search space exploration, analysis will often be faster within a guided architecture.

- **Classic Architecture:** The user, based upon an educated guess, constrains the analysis too tightly in order to reduce analysis, and therefore may miss inferences of interest. The only way to avoid this possibility within classic architectures is to start with a set of minimum constraints and if warranted refine and tighten, through re-analysis. Initial analysis is therefore relatively slow, discovering many inferences of little interest, resulting in a cluttered presentation. Following this, if warranted the user can tighten constraints and re-analyse, however as previously discussed, this can be achieved through post-analysis interaction.
- **Guided Architecture:** The user starts analysis with a minimum set of constraints and as analysis proceeds, the user is able through a combination of both process and algorithmic interaction to progressively tighten constraints as either uninteresting concepts are presented or the user realises that a functional constraint is too tight, thereby resulting in a greater quality of inference set in a single analysis iteration. If the user during analysis realises that a heuristic was too tight and relaxation is required, backtracking is enabled through the use of flexible re-start.

Scalability limitations relate, not to analysis, but to use of quality presentation tools, i.e. the ability of the presentation to effectively facilitate user interpretation. As guided presentation tools are dynamic, it is advantageous for the actual presentation to be directly viewable. In other words the model presented should fit be directly viewable. Therefore as the presentation area (or real estate) is constrained, it becomes increasingly difficult to effectively present elements and their inferences, as the number of elements, and to a lesser extent inferences, increase.

This limitation is difficult to overcome – it is not viable to effectively represent ten thousand elements, let alone show the inferences between them. However, as discussed in Chapter 3, there is little point in presenting such large quantities of information to the user as the inherent cognitive load inhibits interpretation. Although resulting in scalability limitation, it is due to an inherent visualisation problem, which although not limiting the proposed architecture, does constrain its practical application.

7.2 Guidance Architecture

The inclusion of guidance within the knowledge discovery framework requires an extension to the current architectural model as presented in Figure I.2. Figure 7.1 preliminarily introduced the architectural extension required by introducing bi-directional data flow between analysis and presentation: *streaming* and *guidance*. This section presents the proposed generic architectural extension that enables guidance inclusion within the knowledge discovery framework. The discussion first presents the component level architectural extension and then discusses process flow based upon the guidance functionality presented in the previous section.

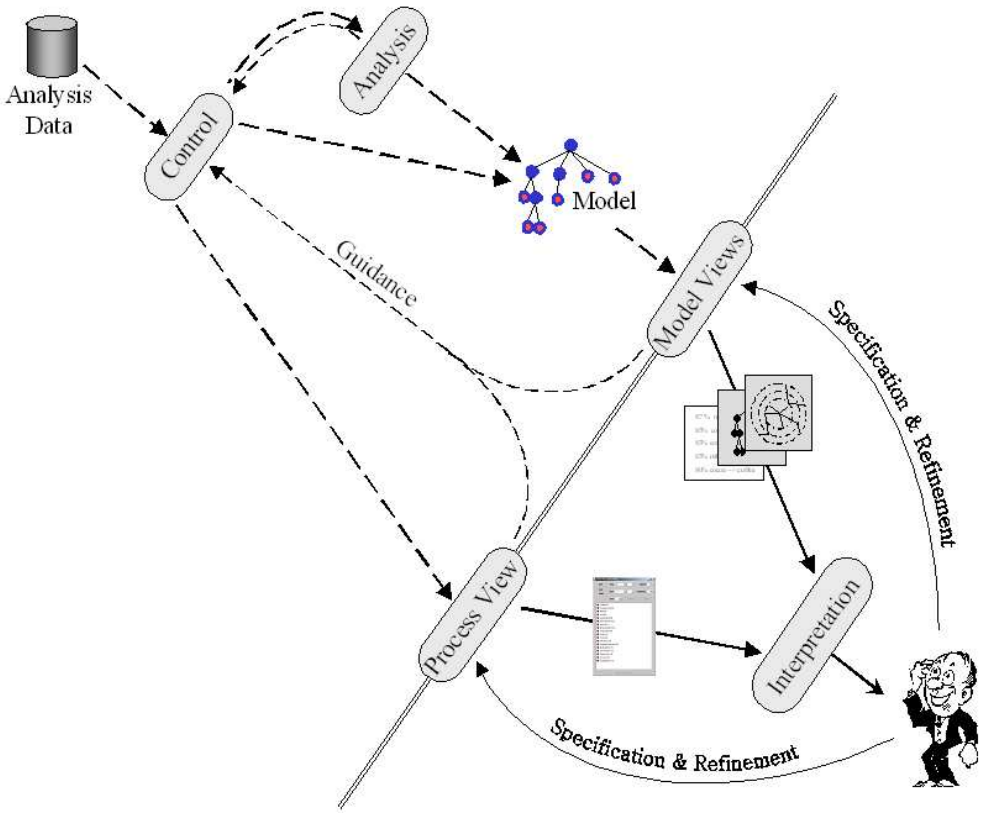


Figure 7.6: Guided association mining system architecture

7.2.1 System Architecture

The proposed system architecture is presented in Figure 7.6, which shows an expanded view of Figure 7.1. The classic architectural components of analysis and interpretation can be seen. The presentation is split into two distinct components providing views of the model and of the process state. The double-line and solid/dashed process flows indicate the boundary between user and system. The straddling of this boundary by the process and model view components indicates their functionality as an interface between user and system. The separation between streaming and guidance flow are indicated through thick and thin lines respectively.

The guidance extension is based upon the well established Model View Controller (MVC) pattern proposed by Reenskaug (in Krasner & Pope 1988), which maps the traditional Input Process Output (IPO) pattern into the GUI realm. This typifies the inclusion of guidance within the knowledge discovery framework as IPO maps directly to classic automated analysis. Therefore the guidance architecture is based upon the triad of component classes defined within MVC, where *model*, maps to analysis, *view*, maps to presentation and *controller*, a new component within the knowledge discovery architecture, is the means by which the user interacts with analysis. In effect, the controller is responsible for mapping user interaction to application response within analysis and the model. The controller maintains functional modularity within the architecture and assumes the central role for directing analysis and incorporating guidance.

The separation of the presentation stage into process and model views is due to their different sources. The model view presents the current model state and provides interactive capabilities that allow the user, through both direct and indirect interaction, to specify both domain and functional constraint refinement. In contrast, the process view provides the same functionality for process constraints. While the model views are derived from the current inference model, the process view is maintained by the controller, which allocates analysis work and therefore controls the state of analysis (see Section 7.2.2). While there is generally a single process view, many model views may exist, each presenting different aspects of the inference model.

Through view interaction the user refines constraints, which are then collected and interpreted by the controller and subsequently used to direct amendments to the model and analysis. This is discussed in more generic detail in process flow and and a specific implementation example is provided in Chapter 8.

7.2.2 Process Flow

This section discusses the process flow model used within the proposed guidance architecture. The model is generic in the sense that it does not make assumptions about the nature of the tools used.

Figure 7.7 is an expanded view of the system side of the architectural extension (Figure 7.6) showing the process flow between architectural components. Within this view of the system the process and *model views* are simple sinks and the system sources are through *data input* and *guidance*. The *model* provides a single transient point, acting as a sink for *control* and *analysis* component output, and a source for multiple model views. Although represented as a single component, model can maintain multiple data structures depending upon analysis implementation. The analysis and control components form the process core, with control providing the mechanisms to direct analysis.

The model's process flow is separated into classic and guidance represented by solid and dashed lines respectively, where the classic process flow is typified by a lack of guidance. In fact the proposed guidance model still allows, in the manner of backwards compatibility, the ability to mimic a dynamic or even classic mining architecture, with respect to automation and transparency. Control is the central flow component, responsible for analysis invocation and controlling subsequent guidance and its influence upon analysis. The following discussion assumes a guided association mining session involving a single process view and multiple model views.

Through control, the guidance system is initialised, based upon the input dataset and default parameters. This involves initialisation of the model, analysis and default view components (including a process and model view) and any other required views. Although no analysis has been undertaken, depending upon the

nature of the model and the default views, an initial presentation can be made to the user. This enables a final opportunity to perform pre-processing tasks, such as element deletion and initial element focus.

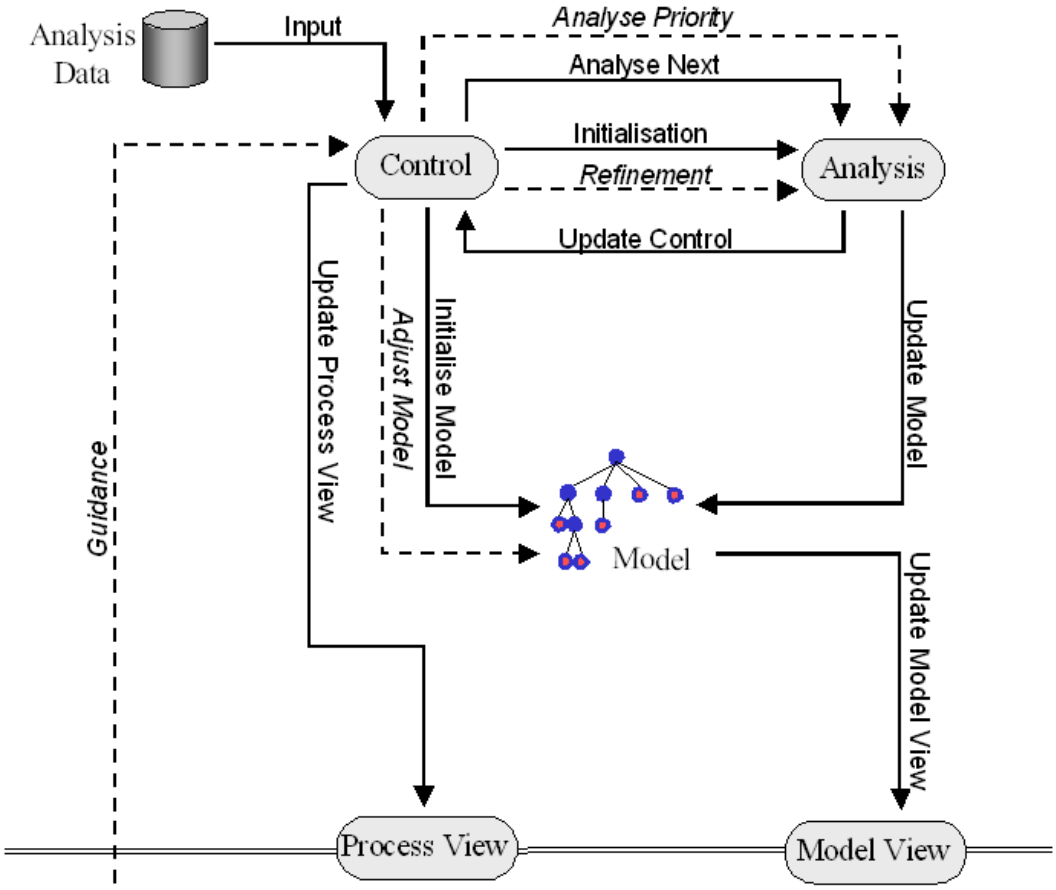


Figure 7.7: Guided association mining process architecture

The model and process views dictate the types of guidance allowed through the presence of both direct and indirect manipulation features. In general, the model views provide an effective means through direct manipulation to perform domain constraint refinement, while functional refinement, which generally has a global effect and is implicitly presented within the model views, is indirectly manipulated through the process view.

Analysis execution can be constrained (as described in Table 7.1) in regard to extent as represented by the flow, *analyse next*. Upon invocation, analysis is automated to the extent specified by the process constraints, which by default should be "null", allowing analysis to naturally conclude, unless interrupted by the user. As analysis proceeds it updates the model and control components with relevant information. By updating model, the MVC architecture cascades any model changes to the relevant model views and by updating control the current analysis state is presented within the *process view*. It is during this update of control that user guidance, specified through the views, is inserted within subsequent analysis processing (also referred to as control check). If no user interaction has occurred since the last control check and no extent constraint has been reached, the next step of analysis is performed. The granularity of guidance inclusion is therefore based upon analysis steps, the smaller the steps the finer the possible guidance granularity.

Although a fine level of guidance is useful, there is a processing overhead, albeit small. As a result, the step size is dictated by the analysis algorithm implemented, because analysis must be in a stable state, in regard to both process and underlying data structures, before regular analysis can be interrupted to perform constraint refinement. An analysis step is therefore an autonomous piece of processing. For example, in Apriori, analysis reaches a stable state at the end of processing a level of candidates (e.g.C_κ), few analysis steps exist within a particular analysis iteration. Therefore step size is large providing limited opportunities to insert guidance, making Apriori a poor choice for guided analysis.

In fact, given the association mining analysis review in Chapter 1, both the candidate generation/merge and pattern growth families of inference analysis are not good choices. Merge is a poor choice because the generally small number of evident stable states leads to limited opportunity for guidance insertion, while pattern-growth algorithms are a poor choice as their use of complex hyper-structures results in complex model views that are difficult to understand.

The best apparent choice for guidance analysis algorithms are from the candidate generation/extension genre. These algorithms are generally based upon simple data structures, such as prefix-trees, facilitating the creation of simple ef-

fective model views. Furthermore this genre results in many stable analysis states, as they are based upon valid elementset extension with a set of single elements. Therefore a stable state exists after the processing of each valid elementset or even after the consideration of each possible extension.

Therefore given no user interaction and no constrained extent, analysis mimics a dynamic architecture, running to completion and dynamically displaying model evolution. A classic architecture can also be mimicked by not initiating a model view until after analysis has completed, upon which it will be populated with the completed model for post-analysis user interpretation.

Given view interaction, guidance may result depending upon the interaction type and the guidance functionality incorporated. For example, navigation or simple selection interaction may not have a mapping to guidance functionality, where as an MOI adjustment or element exclusion will. The types of possible view interactions will be dependant upon the guidance functionality incorporated. For example, given the inclusion of *support* within analysis, if no functional constraint refinement is enabled for *support* then although adjustable before analysis, its related widget, given indirect manipulation, should be disabled during analysis.

The guidance details are passed to the control component where they await deployment depending upon their nature. This can include the updating of the current model and other relevant data structures to a specified extent and adjusting analysis heuristics. Upon completion of refinement deployment, analysis proceeds based upon these updated structures and heuristics. If concept focus has been specified, the prioritised elementsets are analysed first, before returning to regular analysis.

The guidance architecture uses an interrupt-based guidance methodology, in which the analysis process is paused while the user refines the constraints. Once complete the user then re-starts the analysis process. The alternative to this is a blackboard method of inclusion where instead of interrupting analysis, the guidance instructions are written to a virtual blackboard. These instructions are then automatically incorporated at specific points during analysis. Both methods achieve the same result and both could be used, given the guidance architecture specified. However the interrupt method provides more control, allowing time

for the user to interpret the current model state and refine constraints, while in using a blackboard method the analysis continues during interpretation and as such the analysis may overrun the required refinement before it is incorporated.

The manner in which guidance functionality is incorporated within a guidance architecture is implementation dependant, to which end the Guided Association Mining system (GAM) was developed. This system is presented in the next chapter as a proof of concept, presenting a realisation of the guided association mining architecture.

Chapter 8

GAM: Guided Association Mining

This chapter presents a proof of concept tool for the guidance architecture described in the previous chapter. Guided Association Mining or GAM provides an effective guided mining environment, incorporating the dynamic refinement of both concept and heuristic based constraints. To this end, the following discussion focuses upon the techniques used to maintain accurate inference model evolution in the face of dynamic constraint refinement.

The chapter first discusses the role of the three main system components: analysis, presentation and control in detail. A detailed discussion of incorporated guidance functionality is then presented, with the chapter concluding with a short discussion.

8.1 Analysis

As discussed in the previous chapter, the genre of analysis algorithm best suited to the inclusion of guidance is candidate generation/extension due to its fine-grained stability and use of simple data structures. The analysis algorithm used within GAM is based upon concepts introduced in Tree Projection (Agrawal et al. 1999), see Section 1.2.1, a robust extension algorithm that uses tidLists and dataset projection. Furthermore Tree Projection constructs a prefix-tree inference model, or exploration model, that is conducive to effective presentations. This is in contrast to Eclat and Clique that use more complex data structures. The analysis

algorithm implemented within GAM differs significantly from Tree Projection to facilitate guidance through the use of BFT and the storage of additional extension set information, both provide optimisations in guidance environments, which are not critical to analytical success.

Breadth First model evolution was implemented as it follows the visualisation seeking mantra (see Section 3.1), presenting an evolving overview that progressively focuses, on a global scale, upon more complex elementsets. This enables the identification of broad areas of user interest during early analysis, whereas DFT evolution explores the full extent of a single concept, providing a focused exploration technique that is less conducive to general guidance. However, upon invocation of prioritisation (or concept focus) model exploration of the specified concept is undertaken using DFT, in which case the concept in question is explored to its full extent. The storage of additional information within the node extension steps is a trade-off that has been made against memory usage to improve processing upon user refinement.

The following subsections present the the underlying data structures, how they are initialised and finally classic algorithm execution, without refinement.

8.1.1 Data Structures

There are two main underlying data structures of GAM analysis: the *queue* and the *prefix-tree*. The queue, is a support structure containing the sequence of tree nodes, or elementsets, to be processed to maintain BFT evolution. The prefix-tree is the evolving elementset structure used during analysis to represent the explored search space it is also referred to as the inference or exploration model within the guidance architecture. This is because analysis is based upon elementset exploration, from which inferences are derived, therefore it is to this elementset structure that guidance is applicable, while model views may present either the elementset structure or the derived inferences, the model closest relating to the actual exploration will be one based upon elementsets not inferences. Furthermore for each elementset there are multiple inferences, therefore the elementset based model view will provide a less cluttered presentation of the current exploration state.

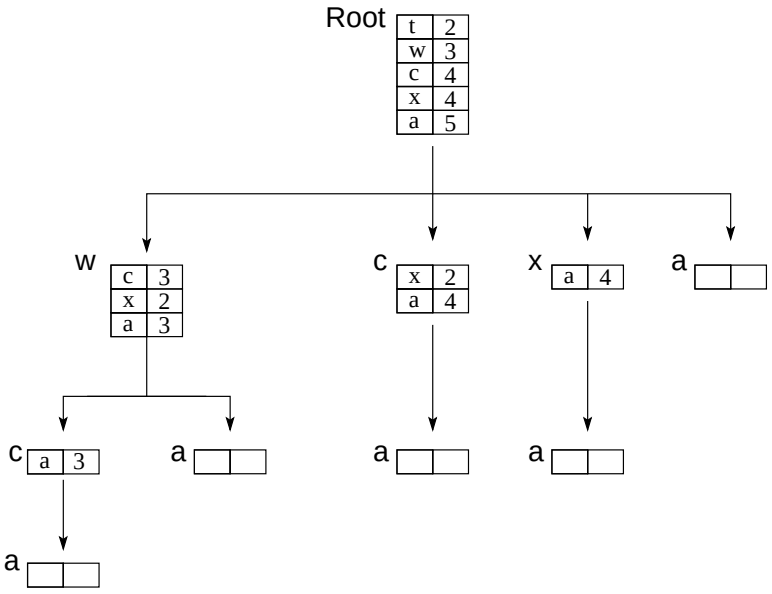


Figure 8.1: GAM prefix-tree (model)

Figure 8.1 presents a complete GAM prefix-tree highlighting the information content of each node, which consists of two parts: the valid elementset, i, and its extension set, i^{ext}, where the extension set consists of those valid parental extensions, or siblings, that occur after the current node's element within the parent's extension set. Also the extension is maintained in increasing order of element presence, maintaining a consistent ordering throughout the tree, which facilitates certain refinement updates. Furthermore, each elementset within the GAM tree consists of two parts, its label and its tidList.

GAM analysis is similar to Tree Projection, as exploration proceeds by merging each extension member with the current elementset and if valid, a child node is created. However in order to facilitate refinement, especially functional refinement, additional information is stored for each concept. A detailed example follows for clarification.

Given a current elementset a, its tidList a^t and it extension set a^e, then the extension of a involves merging a with each $i \in a^e$ resulting in the candidate set C^a. Each $c \in C^a$ is then validated against any relevant constraints, success resulting in c being appended as a child of a.

To facilitate subsequent refinement, GAM stores all candidate elementsets C^a within the relevant extension set a, as shown in Figure 8.1. This results in a trade-off between storage and relaxation refinement optimisation. By storing all C within the relevant extension sets, the update of relaxation refinement is optimised. Candidate storage becomes useful if relaxation is required, which is often the case, as discussed in the scenario presented in Section 7.1.3. A detailed description of how this storage is an advantage is presented in the discussion on heuristic relaxation, Section 8.4.3.

8.1.2 Analysis Initialisation

Initialisation prepares for subsequent analysis by populating the model and queue structures, denoted N and Q respectively, which are reflected within initial model views. The structures are populated from a scan of D, transforming the dataset and incorporating any initial constraints, in effect populating N and Q with valid elements (V_1). The transformation stage maps elements to an ordered numeric representation and the dataset is vertically organised for intersection based accrual. The numeric mapping represents the element's relative frequency within the element set providing pruning optimisation as the least frequent elements are processed first within the prefix-tree. Given that analysis is based upon subsequent element extension, then the largest candidate extension sets are processed first, but as the related elements are the least frequent, the resultant valid extension set is optimally small. This facilitates model balance, as the later extension, although based upon smaller candidate sets, are larger due to increased element frequency.

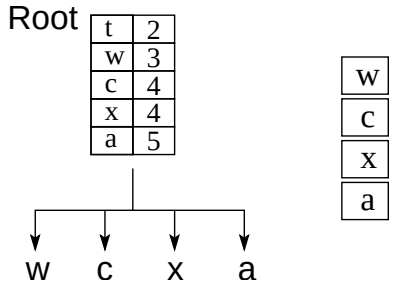


Figure 8.2: Initial GAM structures (model and queue)

The first model level N_1 is populated with V_1, as is Q, however the extension set for N_1 nodes, contains the ordered set of all elements irrespective of frequency stored at $\Re^{ext}$. This storing of all candidate extensions, as previously discussed, provides optimisation upon relaxation refinement. The resultant initial structures are presented in Figure 8.2 based upon the initial state of Figure 8.1.

Through provision of this initial state, the users can interact with the initial model views, manipulating constraints before analysis begins.

8.1.3 Classic Processing

Classic processing refers to dataset analysis using GAM without the inclusion of guidance, so that a classic architecture analysis is mimicked. Given an initialised state as described above, Algorithm 8.1 presents an outline of the subsequent analysis process, in which the inclusion of guidance is alluded to at lines #5 and #13.

Algorithm 8.1 GAM Analysis

```
Analysis(queue Q)
 1: while Q ≠ ∅ do
 2:    node n = Q.pop()
 3:    set X = n.getExtSet()
 4:    for all x ∈ X do
 5:       Control Check B
 6:       node c = merge(n,x)
 7:       n.addExt(c)
 8:       if validExt(c) then
 9:          n.insertChild(c)
10:          Q.push(c)
11:       end if
12:    end for
13:    Control Check A
14: end while
```

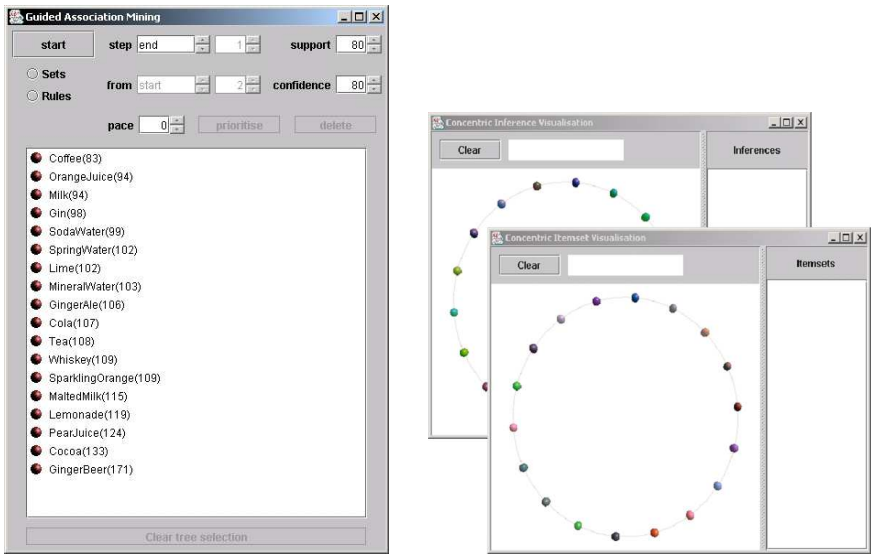
Analysis proceeds by processing each queue member until it is empty. Each $n \in Q$ is extended by merging it with the valid extensions that occur after n, in the extension set of n's parent. Given that extension sets store all candidates and are ordered in regard to elementset frequency, processing is optimised by iterating backwards over this set, stopping when the current extension element is reached. For example, the extension of c, in Figure 8.2, proceeds through the consideration of extensions a and x respectively, stopping at c. Furthermore, although all candidate extensions are stored, infrequent extensions are never considered during processing as they are positioned before the first valid extension and are therefore never reached.

The merge operation results in the creation of a candidate elementset (c) that is appended to n^{ext}. If it is valid in respect to specified constraints, c is inserted into the model as a child of n and pushed onto Q, leading to its eventual extension. This process results in the constrained breadth first exploration of the search space defined by the element set E.

The *Control Check* points with the algorithm illustrate typical stability points at which constraint refinement can be incorporated, where *Control Check A* and *Control Check B* respectively refer to coarse and fine level guidance. *Control Check A* allows guidance to be incorporated after the analysis of each elementset and *Control Check B* provides for finer guidance, allowing its inclusion after each considered extension. A trade-off is evident between granularity of guidance and processing overheads incurred due to analysis interruption. GAM implements the coarser of the two approaches.

8.2 Presentation

The GAM system provides a process view and multiple model views required to facilitate effective guidance, as discussed in Chapter 7. Upon invocation GAM provides a default view incorporating both a process and model view, shown in Figure 8.3(a), while two optional model views presenting a different model aspect and a view of the derived inferences are also available, Figure 8.3(b). Through the use of the multiple model views, various aspects of the model can



(a) Default View (b) Optional Model Views

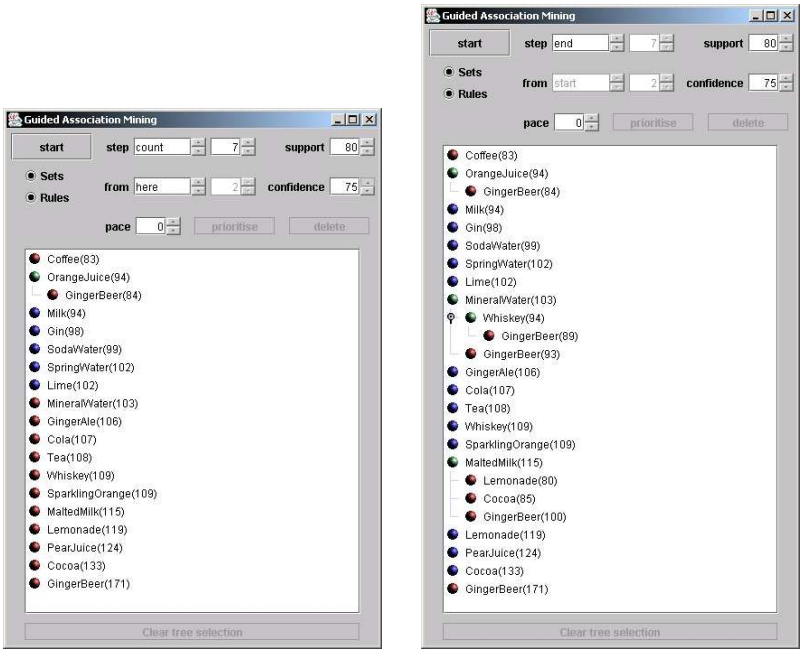
Figure 8.3: Initialised GAM Views

be presented, facilitating user perception as the different views can elicit different patterns. These views or interfaces in association with dynamic presentation open the "black box" providing a transparent analysis system, or dynamic architecture, providing the user interface support for guided analysis.

Figure 8.3 presents an initialised GAM session using the example dataset presented in Figure 1. The model views present V_1, given $\sigma(80\%)$, with the default providing a tree based presentation, while the optional model views: concentric elementset visualiser and concentric inference visual, are based upon CARV (Ceglar et al. 2004). The following subsections discuss these views in detail, including model interface interaction and model view linking.

8.2.1 Default View

The default view is comprised of two main areas: control panel and tree view. The control panel provides indirect guidance through a set of widgets, to refine process and functional constraints. The tree view provides a view of the elementset model, which through direct manipulation enables domain constraint refinement. The tree view presents the model structure in a prefix tree form, where each node is



(a) Intermediate Model (b) Complete Model

Figure 8.4: GAM default view

represented by the valid extension element and the elementset's presence. The tree view evolves as analysis proceeds, to incorporate discovered valid elementsets.

Figure 8.4 illustrates the evolving tree view during analysis. Given the initialised view in Figure 8.3(a), Figures 8.4(a) and 8.4(b) present an intermediate and complete elementset model. Where a leaf node is *red*, an internal node is *blue* and an expanded node is *green*. Within these images it can be seen that paths represent elementsets. For example, in Image 8.4(a) the concept *GingerBeer(84)* being a child of *OrangeJuice(94)*, represents the valid elementset *OrangeJuice,GingerBeer* and specifies its *support* as being 84%.

8.2.2 Inference View

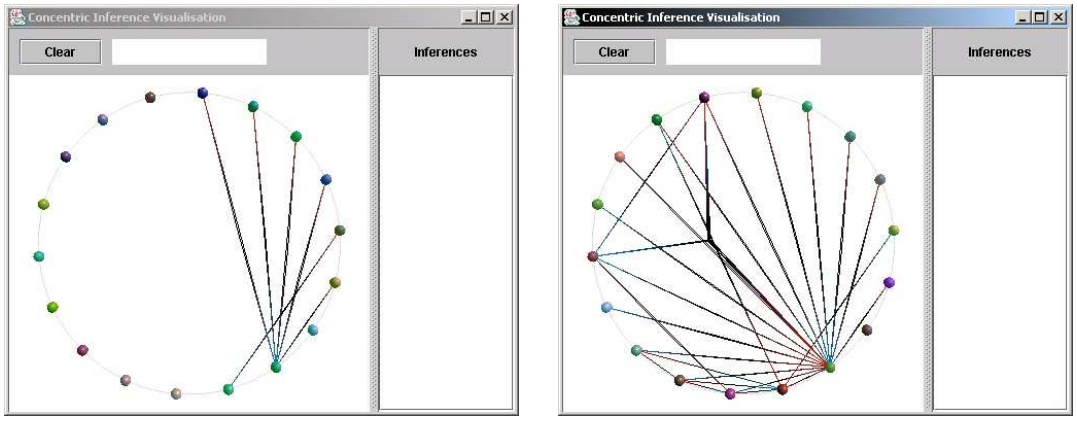
The inference view is an optional view that presents the derived inferences as they are discovered, if this view has not been instigated then the inferences are not derived. This implementation choice optimises analysis by only requiring inference derivation if they are being presented to the user. It may be that it is

the elementsets rather than the inferences that are of interest and need not be calculated. As previously mentioned, it is the constraint of elementset exploration that is critical to quality, not the derivation of inferences.

Given that the inference view is instigated during analysis, the inferences from the current model extent are generated and presented before further analysis. The subsequent discovery of valid elementsets then results in dynamic inference derivation and presentation.

The Inference view is comprised of a graphical and textual component. The graphical component is an instance of CARV and given that no hierarchy is defined it initially appears as a disc upon the circumference of which the valid elements are arrayed. As analysis progresses, the inference view evolves, presenting the discovered inferences as shown in Figure 8.5.

The textual component enables a more detailed analysis of the resultant inferences by presenting textually the inferences selected within the graphical view. For example, Figure 8.6, shows the selection of *GingerBeer* and *MineralWater* and the subsequent textual presentation of the inferences within which *both* these elementsets participate. Selection is through direct manipulation and is represented graphically by "boxing" the relevant spheres.



(a) Intermediate Model (b) Complete Model

Figure 8.5: GAM inference views

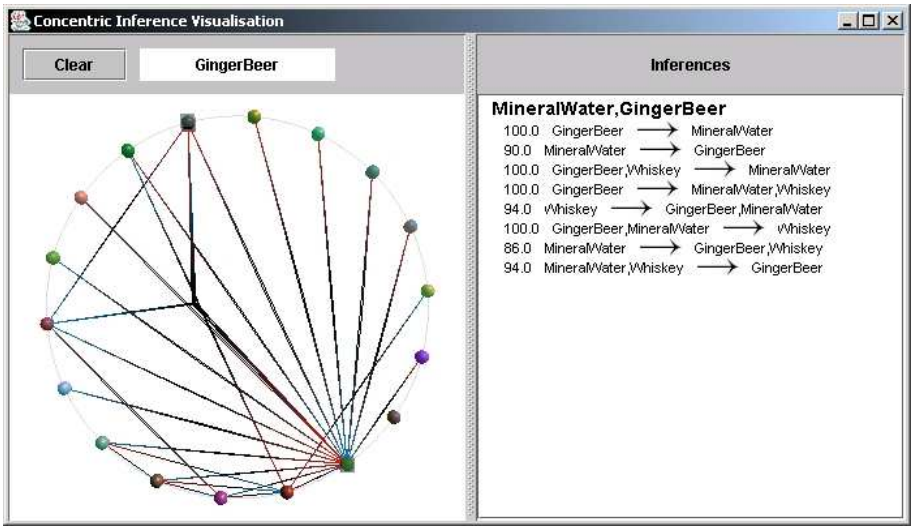
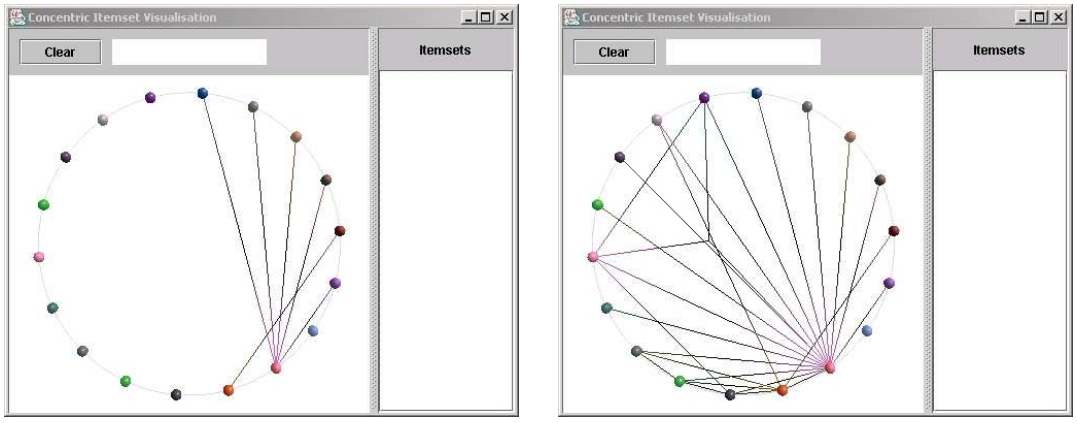


Figure 8.6: GAM inference view selection

8.2.3 Elementset View

The elementset view, shown in Figure 8.7, is derived from CARV and is similar in principle to the inference view. However, this view presents an alternative to the elementset model and conveys different model information to the user, potentially improving interpretation.

The graphical and textual components of the elementset view work in the same manner as the inference view. However the graphical view now presents valid elementsets instead of inferences. Therefore within the presentation, relationships between elements now represent valid elementsets. This CARV extension is significant as the two optional views, although different are presented in a common manner and reduce cognitive load. This can be seen by comparing Figure 8.6 and Figure 8.8, which present concurrent snapshots of the views and highlight through linked selection (see section 8.2.4) the derivation of inferences from valid elementsets. It can also be seen from this that a single elementset generally results in many inferences and hence the elementset view is more scalable providing a less cluttered presentation.



(a) Intermediate Model (b) Complete Model

Figure 8.7: GAM initialised elementset views

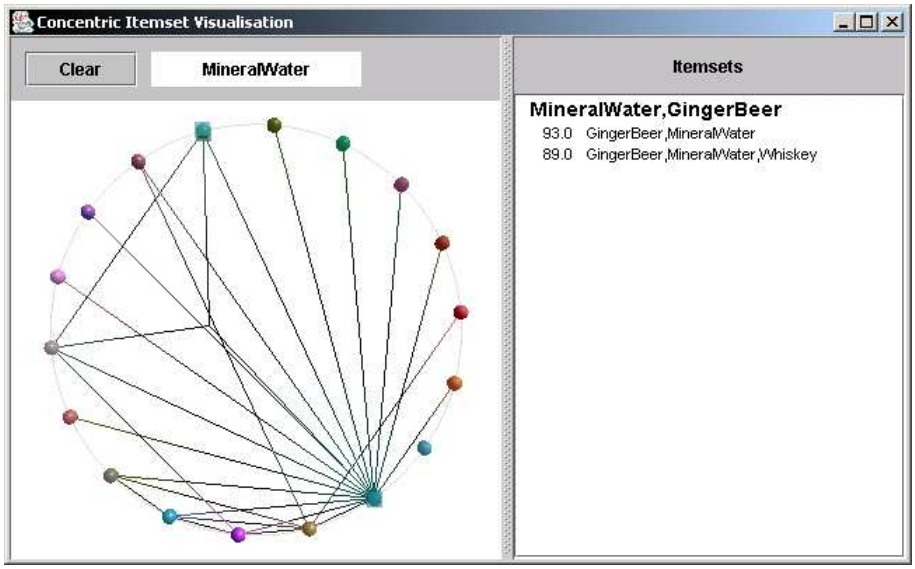
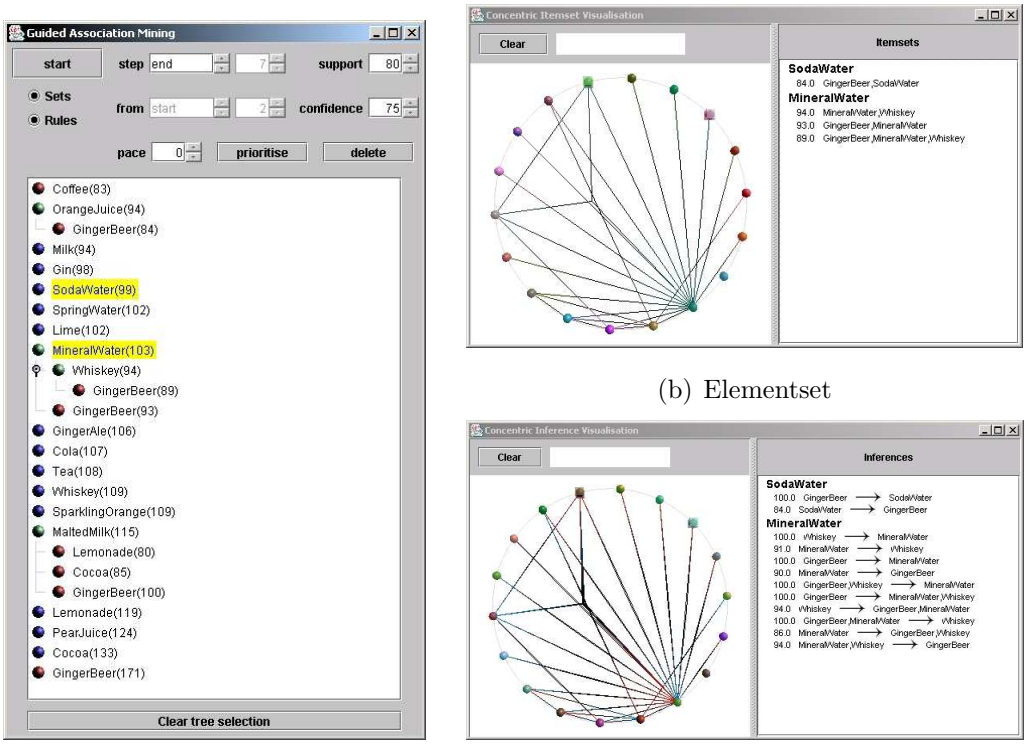


Figure 8.8: GAM itemset view selection

8.2.4 Model view management

GAM ensures that all model views maintain a consistent state, by implementing view management within the control component of the architecture. Therefore given model effecting view interaction, such as deletion and selection, control effectively ensures that all other instigated model views are updated. For example, given the deletion of an element through direct elementset interaction, the element is also removed from the default tree view and the inference view, while element selection within inference view results in not only the textual display of inferences in its associated panel, but also the selection and textual display of participant elementsets within the elementset view and its selection within tree view. However not all interaction requires global update. Consider the expansion of a path within tree view or the rotation of the elementset or inference models. An example of model view management is provided in Figure 8.2.4.



(a) Default

(b) Elementset

(c) Inference

Figure 8.9: GAM model view management

8.3 Control

The main function of the control component is to provide an interface between the user and the analysis process, enabling analysis synergy. To effectively provide this the control component also maintains system consistency, by ensuring that all aspects of the guided analysis session maintain a consistent state.

The main components of control are presented in Figure 8.10 along with their interaction with external components and data structures. Within this figure the classic analysis is signified through the use of bold process flows. The external components and data structures used are derived from Figure 7.7, except for the *Excluded* array based structure used to facilitate concept exclusion.

Given an initialised system state, all subsequent activity is triggered through user interaction, including constraint refinement, analysis and the instigation of optional model views. The nature of the interaction is determined by the *Interpreter* and the relevant processing is undertaken to comply with the request and ensure resultant inference model accuracy. The actual types of interaction and their interpretation are presented in the next section.

At a higher concept level, all interaction is received by the *Interpreter*. Process constraint adjustments are sent to *Extent*, while concept and heuristic constraint adjustments are sent to *Adjustment* and *Analysis*, depending upon task and the current analysis state.

While most process flows within Figure 8.10 are generic, a few relate to particular types of interaction included within GAM, these are analysis priority, select update and append (presented below).

- **Analyse Priority** Focused concept analysis is distinguished due to the different traversal strategy used.
- **Select Update** Maintain consistency between multiple views.
- **Append** Maintains a list of excluded elementsets.

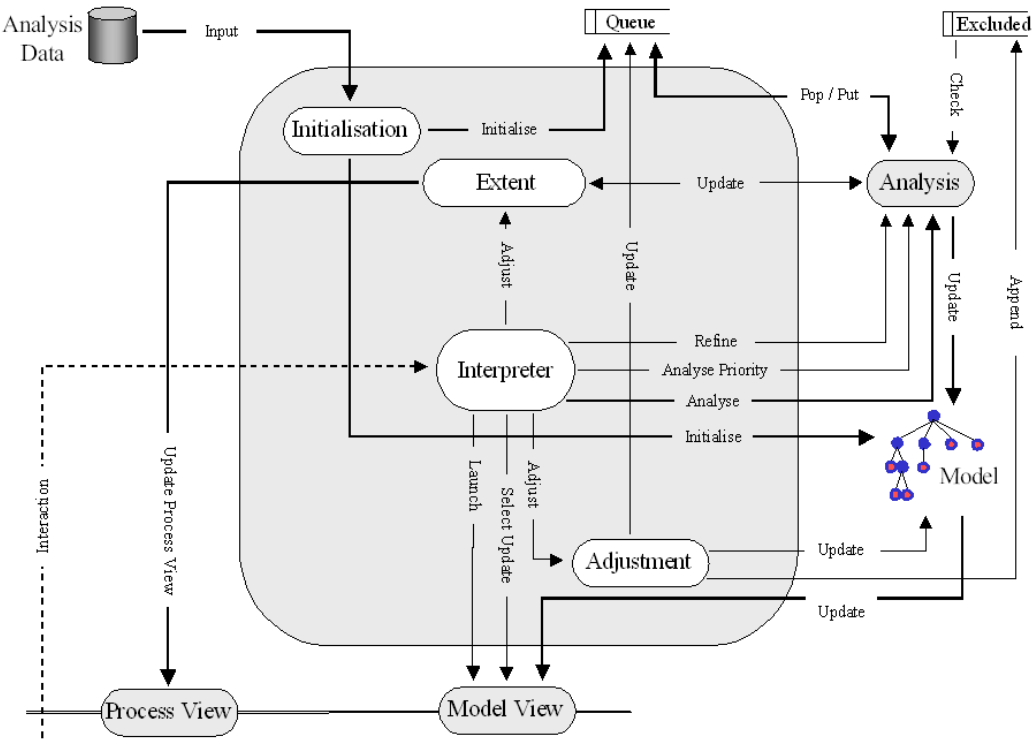


Figure 8.10: Control component

8.4 Guidance

This section presents the techniques by which guidance functionality has been incorporated within GAM, based upon the guided architecture interaction discussed in Section 7.1.3. Given the summarisation of required guidance functionality presented in Table 7.1, GAM includes all but *concept adjustment*, as it is not supported by the dataset used. However its exclusion does not detract from the tool or the underlying theory for two reasons. Firstly, it is not a common task. Secondly, it's inclusion, given a relevant dataset, is replicated by functionality included within GAM, namely concept selection, focus and flexible restart. Once the elementset is selected, the attribute is adjusted, then re-analysis of the elementsets in which it participates occurs in a prioritised fashion, using restart to begin analysis at the elementset in question.

The inclusion of process constraints due to their task independency is exhaustive. In relation to heuristic refinement, GAM allows the relaxation and tight-

ening of *support* and *confidence*, which typify heuristic constraints. While not exhaustive, the technique used to incorporate *support* refinement can be applied generically to all reflexive heuristic constraints. If the heuristic constraint is not reflexive then re-analysis from the initial model state is required to accurately represent the refinement. Concept refinement is incorporated through prioritisation and exclusion, while other forms such as templates have not been considered, a similar methodology to that of exclusion can be used (see Section 8.4.2).

The following subsections detail the inclusion of guidance within GAM. Figure 8.11 provides a snapshot of the elementset visualisation based upon the analysis of the dataset with support $\sigma(80\%)$. This figure is used to highlight the effect of refinement in the following subsections.

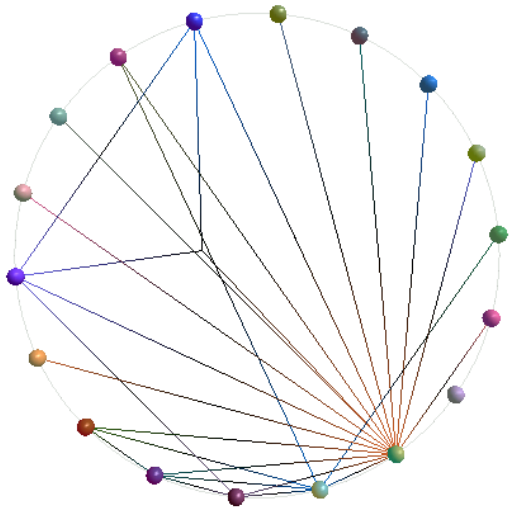


Figure 8.11: Complete elementset model at $\sigma(80)$

8.4.1 Process Constraints

Process constraints do not effect the nature of search space exploration but enables the user to control the automated analysis in terms of extent, pace, and point of invocation. Four classes of process constraint are incorporated within GAM (see Figure 8.10), the classes of *pace*, *interrupt* and *extent* are handled within *Extent*, while *restart* effects the model and associated data structures so that analysis, resumes from a previous point.

As previously shown (Algorithm 8.1) the analysis algorithm specifies two *Control Check* candidates, of which at least one must be implemented to incorporate guidance. This updates *Control* with information regarding current analysis state and ultimately provides the opportunity for guidance inclusion.

Through *Control Check* the process constraints of extent, interrupt and pace are included. Given that *Control Check* updates the current automated extent, then if the user has specified a constrained *extent* its state is updated, if the *extent* is reached analysis stops, passing control to the user. *Interrupt* forces analysis to stop upon the next *Control Check*. Finally *pace* allows the user to vary analysis speed, pausing for a user specified period during *Control Check*. The implementation of the extent constraint within GAM can take the forms of end, count, level and siblings (presented below).

- **End** (Default) No constraint.
- **Count** Analyse x number of elementsets.
- **Level** Automated analysis until a specified model depth,or level, has been reached.
- **Siblings** Automated analysis for siblings of current elementset.

The final process constraint is *restart* that allows the user to resume automated analysis from previous stages of model evolution. By default, analysis will resume from the current state, however given model views, the user, through interpretation of the current analysis state, may want to refine and resume from an earlier point of analysis. For example, during analysis the user realises that support is too low, producing too many valid elementsets, and tightening is required. The user is able then to *interrupt* analysis, refine *support* and resume analysis from a previous state such as initial state or third level.

Flexible restart is instigated through a call to the *Adjustment* component within *Control* that controls the update of the underlying data structures to reflect refinement, be it concept exclusion or restart. The implementation of restart requires model adjustment and a rebuild of the queue to reflect the model adjustment.

The queue is first cleared and the model is then traversed in DFT removing those elementsets below the specified restart level and adding the new leaf elementsets to the queue. For example, given the completed tree in Figure 8.1 and the constraint of restart within GAM to model levels, the user can resume analysis from level 2 (initial) or level 3. Given the specification of a level 3 restart, Figure 8.12 presents the adjusted model and queue.

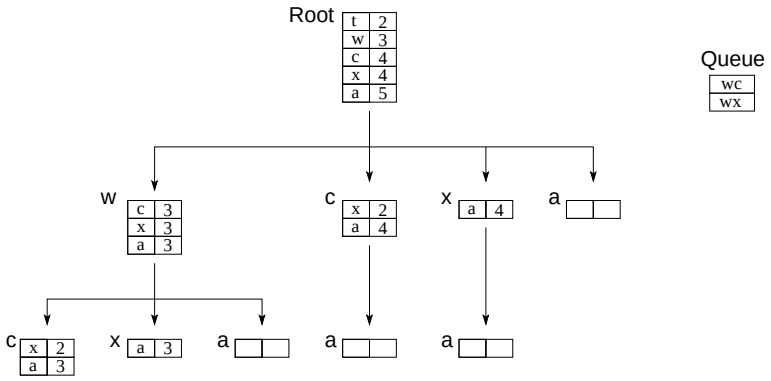


Figure 8.12: GAM restart

8.4.2 Domain Guidance

Domain Guidance refers to the refinement of domain based constraints including, concept filtering, concept adjustment and concept focus. Within GAM, concept guidance is represented through the implementation of elementset focus and exclusion, while concept adjustment is excluded for reasons previously discussed. The following subsections detail the techniques used to incorporate domain based refinement within GAM.

Concept focus is an important guidance function allowing the prioritised exploration of an elementset. In effect, it is a positive constraint that enables the user to generate all valid elementsets, and hence inferences, in which a specified elementset participates before any others.

Concept filtering, which can define both positive and negative domain constraints, is represented through the implementation of concept exclusion, which allows users to remove elementsets from further consideration. While there are

many types of concept filtering, exclusion is indicative of these and the methodology introduced within GAM can be easily adapted for generic use.

Concept Focus

The ability to dynamically prioritise the exploration of an elementset is a significant feature of guided analysis, as it allows the user to iteratively specify narrow areas of interest, exploring them before any other areas. This is implemented within GAM through the invocation of a DFT exploration of the focus concept.

However, as the entire extent of elementset participation is to be explored, model ordering used during regular analysis does not apply. For example, given an ordered set of valid elements a, b, c, d, e, f and a focus concept de then using GAM analysis, $de^{ext} = \{f\}$. However consideration must also be given to potential elementsets derived from extensions a, b, c, which will not be considered given regular analysis of de.

Algorithm 8.2 GAM priority analysis

```
Priority Analysis(queue Q, node n)
 1: node m = new (n)
 2: m^ext = validElements()
 3: Queue P = new Queue(m)
 4: while P ≠ ∅ do
 5:    node m = P.pop()
 6:    set X = m.getExtSet()
 7:    for all x ∈ X do
 8:       node c = merge(m,x)
 9:       m.addExt(c)
10:       if validExt(c) then
11:          insert(c)
12:          P.push(c)
13:       end if
14:    end for
15: end while
16: Q.remove(n)
```

Prioritisation creates a copy of the focus concept, referred to as a facilitator, within which ext is comprised of all valid elements. Facilitator analysis is then

undertaken in regular DFT fashion, using a separate queue (focus-queue) and inserting discovered valid concepts in their correct positions within the model. This, in effect, undertakes DFT exploration of the focus concept, but uses regular exploration techniques.

Therefore, given the discovery of the valid elementset *bde*, it is inserted within the model as a child of *b*. Finally the focus concept is removed from the regular queue avoiding subsequent replicated analysis. This prioritised analysis is shown in Algorithm 8.2 and an example is presented in Figure 8.13, showing the result of prioritising the concept $SodaWater$ at $\sigma(60\%)$, based upon the default concept model in Figure 8.11.

Anomalies can be caused during insertion of focus concepts into the model if intermediate concepts have not yet been discovered. For example, given the above case, *bde* is inserted as a child of *b* as *bd* does not yet exist. Assuming the inclusion of only non-monotonic constraints there are two possible solutions: to deal with it on the fly when *bde* is inserted, or to reorganise the subtree of *b* upon discovery of *bd*.

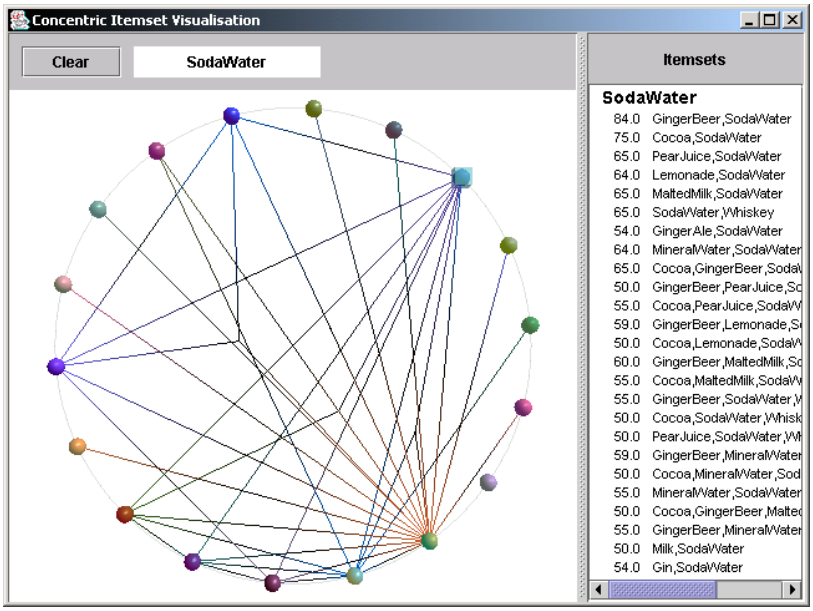


Figure 8.13: Focus analysis of $SodaWater$, $\sigma(60)$

The optimal method appears to be the creation of intermediate concepts on the fly to alleviate the need for *ad-hoc* reorganisation during subsequent analy-

sis. Finally some subsequent replicated analysis is unavoidable as regular analysis encounters those concepts discovered during focus. For example, given the insertion of *bde* during prioritisation, regular analysis will want to explore the subtree, which has already been undertaken. This is overcome by not appending elementsets to the focus-queue if they already exist within the model, unless heuristic refinement has occurred.

Concept Exclusion

Concept exclusion enables the user to remove elementsets from further consideration resulting in exploration restriction. Its inclusion therefore requires model adjustment and given the nature of GAM analysis, storage for reference during subsequent analysis. The model is adjusted by performing a partial DFT and removing all nodes within which the excluded elementset participates, given that if a node is excluded so are all its descendants. The excluded concept is then stored in the *Excluded* data structure, which is checked upon generation of new concepts. If the excluded elementset is a participant it is excluded.

The apparent inefficient management of excluded concepts is required as GAM analysis cannot exclude, using its regular underlying structures, the generation of excluded concept supersets during regular analysis. Therefore these concepts must be stored and checked against all discovered valid elementsets. Figure 8.14 shows the result, within the elementset view, of excluding two elements *SodaWater* and *Whisky* from the default model presented in Figure 8.11.

However, simple element exclusion can be achieved without requiring the subsequent step of storage and analysis checking. This is achieved by scanning the model and removing all instances of the element from *ext* sets as well as the nodes within which it participates. As the element does not exist in any form no elementsets can be derived from it and hence the need for storage and checking is alleviated.

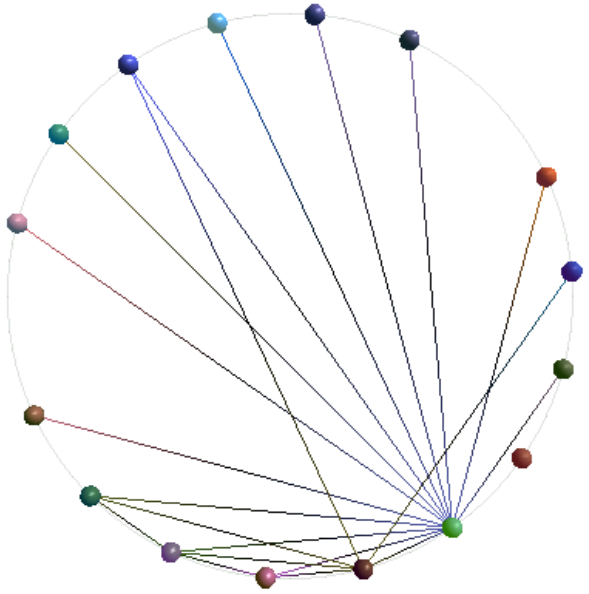


Figure 8.14: Exclusion of *SodaWater* and *Whisky* from the default model

However elementset or complex concept exclusion cannot be achieved in the same single step manner as it may result in extraneous removal, therefore the second storage and checking step is required. This is because by removing all elements of an excluded elementset from the model, concepts that share common elements but are not supersets of the specified complex concept will also be excluded. For example, given the exclusion of ce and the valid concept b such that $b^{ext} = \{c, e, f, g\}$, the elements c and e cannot be removed from b^{ext} as it will ultimately result in the exclusion of concepts bc and be, which are still valid. Therefore the only way to effectively exclude a complex concept is to perform the second step that maintains a list of excluded concepts against which all subsequent discoveries are checked.

GAM, as currently implemented, does not provide mechanisms to enable the inclusion of a prior excluded concept. However this could be achieved by implementing a variation of concept focus, by which the inclusion concept is explored, using DFT analysis, to the current model state, inserting valid concepts into the model.

8.4.3 Heuristic Guidance

Heuristic guidance relates to the dynamic refinement of functional constraints. This is presented in GAM through enabling refinement of the common heuristic *support* and *confidence*. Both of these constraints are non-monotonic in principle, however while *support* is included during exploration, *confidence* is used only during inference derivation and as such is presented separately. Furthermore, through cooperation with *restart*, the refined heuristic can be incorporated from previous model states, not just from the current state. In the case of heuristic relaxation the storage of invalid candidates is used to facilitate an accurate model update.

Typically heuristic relaxation will be incorporated from the initial state, ensuring that all valid concepts are discovered. In contrast, heuristic tightening may well be incorporated from the current state, as all valid concepts under the refined heuristic already exist within the current model and as previously discussed, can be excluded through post-analysis filtering.

Although the heuristic refinement techniques presented are applied to a specific MOI, the principles involved are generic and easily adapted to the refinement of any reflexive heuristic constraint. The only consideration is that the reflexive nature of the constraint can influence the traversal method required in order to incorporate pruning optimisations. If an incorporated constraint is non-reflexive, directional based optimisations cannot be incorporated. Given this, the proposed refinement technique has two stages: model update and heuristic adjustment.

Although heuristic refinement can be considered a two part process, the refinement complexity lies in the accurate updating of the current model. The following two sections present a technique for accurate model update for tightening and relaxation respectively, which considers the case of complete model update, or the inclusion of refinement from the initial state. This provides the most complex scenario, as inclusion from a current state does not require model update, only heuristic adjustment, for subsequent analysis, while inclusion from an internal model level, specified through restart, requires a constrained model update that only updates nodes below the specified level. The final subsection discusses the refinement of the inference constraint *confidence*.

Heuristic Tightening

The tightening of a reflexive heuristic results in exploration reduction, or the production of a smaller elementset lattice. In general, tightening is a two part process, involving model update and heuristic adjustment, however the extent to which the model is updated is dependant upon restart (as discussed above) which partially effects the extent of lattice reduction.

Given the requirement of a complete model update, the model is traversed, removing all sub-trees rooted at now invalid elementsets, due to the non-monotonic nature of *support*. To avoid subsequent invalid exploration, leaf nodes of removed sub-trees are removed from the queue. There are two methods of performing this given an invalid elementset. Firstly, the queue can be scanned, removing all elementsets that are supersets of the invalid concept. Secondly, given that the queue contains pointers to model nodes, by traversing over the invalid sub-tree, each node can be removed from the queue. While the first choice requires only partial model traversal, the second requires full traversal, including invalid sub-trees. The most efficient choice is dependant upon both queue and invalid sub-tree size, at present the second method has been implemented within GAM. Once the traversal is complete the model and queue contain valid concepts only and further analysis can be undertaken with the refined heuristic value. The update traversal is presented in Algorithm 8.3, where l represents the level of refinement inclusion specified by restart.

Within GAM, although the invalid elementsets are removed from the model, the top-level invalid elementset still exists as an invalid candidate extension of its parent. For example, given the invalidation of ce, it is removed from the model, however it is still represented as an invalid candidate extension within c^{ext}, eg $c^{\{e,g,i\}}$. As previously discussed, the storage of generated candidates irrespective of heuristic validity is used to facilitate subsequent relaxation. Although this only happens for the invalid sub-tree root, due to the subsequent parental existence, which cannot be said of its descendants.

The effect of heuristic tightening is shown through Figures 8.15 and 8.16. Figure 8.15 presents the elementset model generated with a *support* of 60, while Figure 8.16 illustrates the effect of tightening *support* from 60 to 80 after the 2^{nd}

Algorithm 8.3 GAM heuristic tightening

Tighten(queue Q, node n, support s, level l)
1: **for all** $m \mid m \in n$.children **do**
2: **if** $\sigma(m) \leq s \ \wedge \ m_{level} \geq l$ **then**
3: QueueRemove(Q,m)
4: n.remove(m)
5: continue
6: **end if**
7: Tighten(Q,m,s,l)
8: **end for**

QueueRemove(queue Q, node n)
1: **for all** $m \mid m \in n$.children **do**
2: Q.remove(m)
3: QueueRemove(Q,m)
4: **end for**

level exploration of *SodaWater*. To highlight the tightening effect upon model exploration, Figure 8.16(b) presents the result of tightening from the current analysis state. The figure therefore presents the result of analysis continuation from the current state, shown in Figure 8.16(a), without model update, with $\sigma(80)$. The resultant model is therefore a combination of the two models explored at $\sigma(60)$ and $\sigma(80)$ as shown in Figures 8.15 and 8.11 respectively.

Heuristic Relaxation

The relaxation of a reflexive heuristic results in expanding exploration, or a larger resultant elementset lattice. Like tightening, relaxation is a two part process, involving model update and heuristic adjustment, however the extent to which the model is updated is dependant upon restart, as discussed above, which effects the extent of lattice extension. Furthermore, this process is facilitated within GAM through the storage of previously calculated candidates irrespective of heuristic validity. This optimises relaxation by reducing the processing required to update the current model.

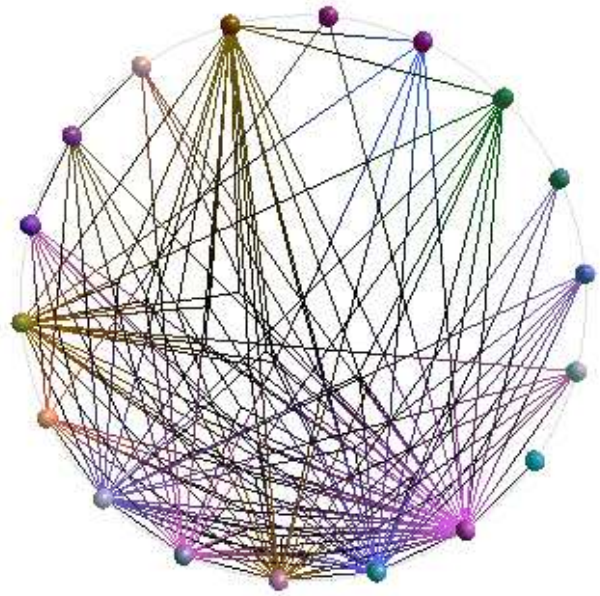
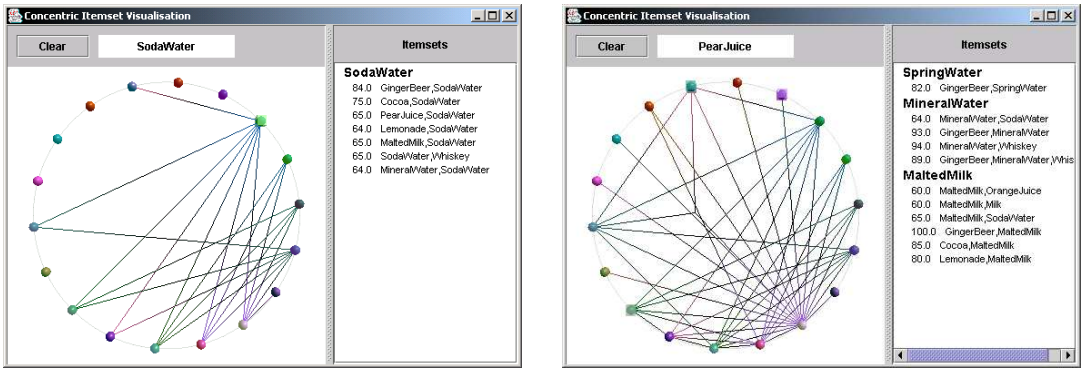


Figure 8.15: Complete elementset model at $\sigma(60)$



(a) $\sigma(60)$ to Sodawater (b) Resume from current state, with $\sigma(80)$

Figure 8.16: Heuristic tightening on *SodaWater* from $\sigma(60)$ to $\sigma(80)$

Model relaxation is undertaken in DFT, involving a two stage process for each node. Firstly, existing *ext* members are re-validated, appending to the model those candidate extensions that have become valid through the relaxation. Secondly, given the validity of previously invalid elementsets, new candidate extensions are generated for a node based upon the newly valid parental extensions.

Given the specification of model refinement, restart from previous state, then a full model traversal is required to accurately update the model. Algorithm 8.4 presents the recursive technique implemented to relax the current model, where s and $\bar{s}$ respectively represent the new and old support values, l represents the model level associated with the current model state and r represents, restart value, or the level from which relaxation is to be included within the model.

Algorithm 8.4 GAM heuristic relaxation

Relax(queue Q**, node** n**, int** s**, int** $\bar{s}$**, int** l**, int** r**)**

```
 1: if n^level ≥ r then
 2:    for all m | m ∈ n^ext ∧ s̄ > σ(m) ≥ s do
 3:       Append(Q,n,m)
 4:    end for
 5:    if n.parent != null then
 6:       for all x | x ∈ n^siblings − n^ext do
 7:          node c = merge(n,x)
 8:          n.extInsert(c)
 9:          if σ(c) ≥ s then Append(Q,n,c)
10:       end for
11:    end if
12: end if
13: if n^level ≤ l then
14:    for all k | k ∈ n.children do Relax(Q,k,s,s̄,l)
15: end if
```

The Algorithm 8.4 is comprised of two main parts: re-validate and discovery given a valid portion of model relaxation, specified through r and l. Revalidation iterates over a node's invalid candidates, appending those that become valid, while discovery generates new extension candidates for nodes given the validation of new siblings during parental relaxation. These new candidates are inserted into the node's extension set and if valid they are also appended.

The *Append* process, not explicitly shown, relates to elementset insertion, the actual nature of which depends upon its position within the model. Given that the model being relaxed is generally not complete, relaxation is constrained to the current model extent. Therefore if the appended elementset will be inserted at the current model level, it is also inserted into the queue. For the same reason l

is included within the algorithm to restrict update traversal to the current model extent. Therefore the result will be a relaxation to the current model extent, with those areas, or elementsets, to be further explored existing within the queue.

This process is further explained through relaxation of Figure 8.1 from a *support* threshold of 3 to 2, the result of which is presented in Figure 8.17.

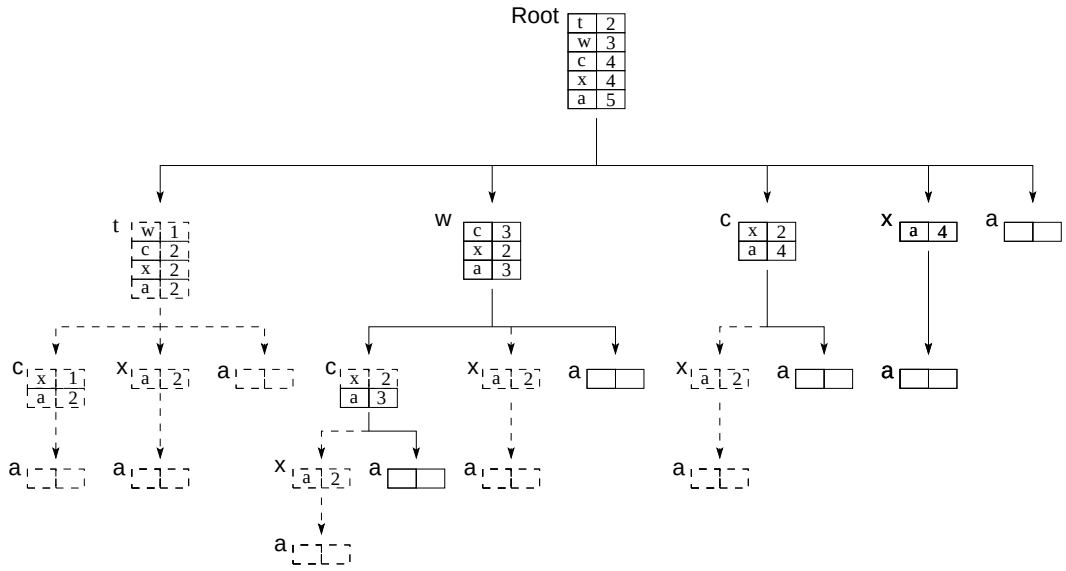
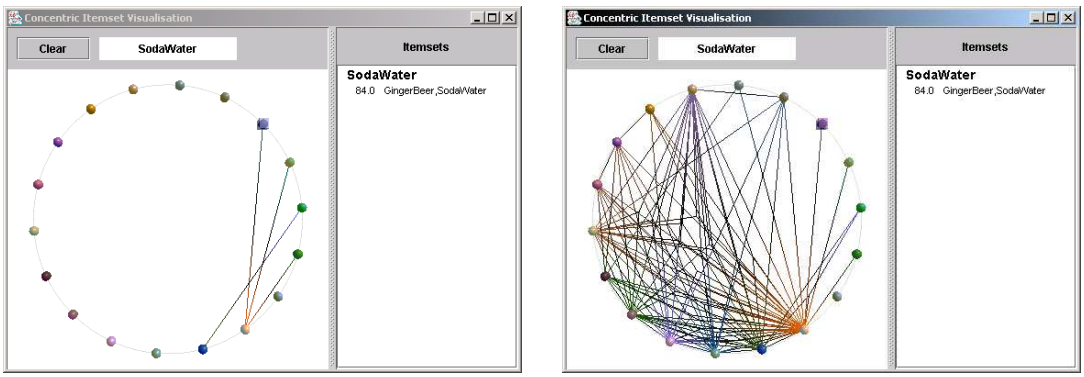


Figure 8.17: Prefix-tree model relaxation

Although the model presented in Figure 8.1 is complete, the application of relaxation is still appropriate. Figure 8.17 presents the completed relaxed model in which new artefacts are signified by dashed extensions and candidate sets. Using the process described in Algorithm 8.4, relaxation first analyses $\Re$, discovering that element t is now valid, resulting in its addition to the model. Subsequently using DFT, t is explored to the current model depth, resulting in the insertion of elementsets tca and txa to the queue.

During the subsequent analysis of w, wx becomes valid and is appended. The validation of wx results in the consideration of x as an extension of wcx, which is found to be valid and appended to the model and queue for further exploration, due to its depth within the model. Once traversal is complete the model accurately represents the refinement up to the current analysis state. Subsequent analysis results in the complete relaxed model shown in Figure 8.17.

The effect of relaxation is also presented in Figure 8.18, an inverse presentation to that shown in heuristic tightening. This figure shows the effect of analysis at $\sigma(80)$ to *Sodawater* and then relaxing the *support* heuristic to $\sigma(60)$ for subsequent analysis. Similarly to heuristic tightening, Figure 8.18(b) presents the result of tightening from the current analysis state.



(a) $\sigma(80)$ to Sodawater (b) Resume from current state, with $\sigma(60)$

Figure 8.18: Heuristic relaxation on *SodaWater* from $\sigma(80)$ to $\sigma(60)$

Inference Heuristic Refinement

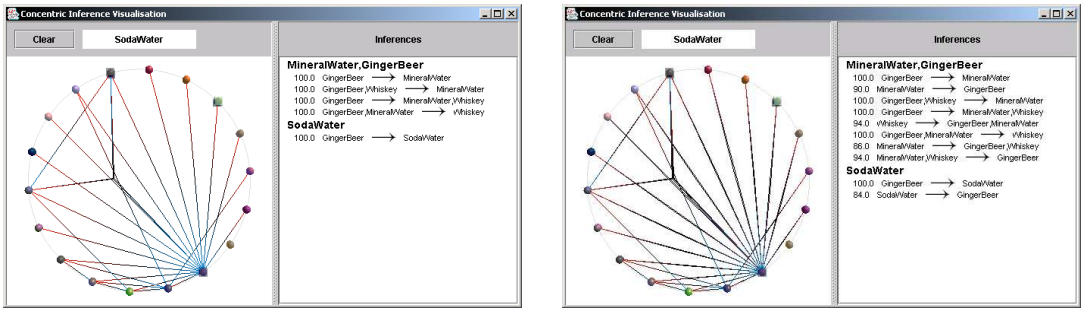
Inference derivation is typically constrained through inclusion of the *confidence* heuristic, improving inference quality by incorporating a strength measure. GAM enables the dynamic refinement of *confidence* during exploration when the optional Inference Visualisation tool is active, by deriving and presenting inferences from elementsets as they are discovered during exploration.

In a similar manner to *support*, once an inference is generated it is stored irrespective of subsequent heuristic validity. Therefore the inference list contains all inferences valid for the lowest confidence threshold specified. This facilitates subsequent refinement as, so long as the confidence threshold exceeds the previous lowest value, a traversal of the elementset model is not required to generate an accurate inference set. All the required inferences have been generated.

Confidence refinement is, again, a two part process requiring model update and heuristic adjustment for subsequent analysis. However in the case of *confidence*

the entire inference set is refined, there is no notion of confidence refinement from a particular state with GAM.

Both tightening and relaxation result in complete inference list traversals. Tightening results in a reduced inference presentation by iterating through the list and "turning off" invalid inferences. Hence all inferences remain within the listing, however only a subset are presented. Relaxation results in a similar process, but extending the presentation by "turning on" valid inferences. However if *confidence* is relaxed below the previously smallest value, naive inference derivation is undertaken by traversing the elementset model, appending to the list any new inferences. The effect of confidence refinement is shown in Figure 8.19.



(a) $\gamma(100)$ (b) $\gamma(80)$

Figure 8.19: GAM Confidence Adjustment highlighting *SodaWater* and *MineralWater, GingerBeer* for completed model based upon $\sigma(80)$

Confidence refinement received less attention, and was naively implemented, due to its relatively low significance, given the goals of this research, in which the focus was maintained upon exploration guidance, and hence the evolution of the elementset model. Inference filtering can be viewed as post-processing and the storage of inferences within GAM is currently within the presentation tool, in this case CARV, localising the presentation material.

Part V

Conclusion

Current state of the art data mining tools are automated, but the perfect data mining tool is interactive and highly participatory.

Georges Grinstein, 2002

This thesis extends the current body of knowledge in the field of guided knowledge discovery. This is achieved through the proposal of a generic guided analysis architecture for descriptive knowledge discovery tasks and the development of a guided association mining environment based upon this proposed architecture. The effective realisation of these artefacts supports recent statements by prominent researchers as to the the benefit of maintaining user-computer synergy during the analysis process, a concept, which although widely discussed, has seen little research activity.

Although the benefit of this synergy is not quantifiable, due to its subjective nature and the general case of an initial null hypothesis, the possible benefit that this synergy can provide is not in question. The question that this thesis addresses is how this synergy can be provided in an effective manner. Previous guided analysis research is minimal, with this thesis contributions providing a significant step in guided knowledge discovery, which will hopefully be used to stimulate further work in this important area of knowledge discovery research.

Furthermore this thesis also provides novel contributions to the foundation areas of analysis and presentation, through the proposal of MCL and CARV. The concept of MCL was elicited during a review into incremental association mining, in which it was thought that the maintenance of a condensed representation may be of interpretative benefit to the user. It has been shown to be effective in its maintenance of smaller structures. However further comparative work is required against other existing incremental association analysis algorithms. CARV (Ceglar et al. 2004) is more mature and has been used to support further research in which it has shown its usefulness, not only in hierarchical presentations but also in presentations where no hierarchy is evident.

The following sections complete this thesis. The first provides a discussion that compares the guidance contributions of this thesis against previous published work, illustrating the significant contribution of this thesis to guided knowledge discovery. The second section provides a discussion of avenues of further research based upon this thesis contributions.

V.1 Discussion

For a concept that theoretically shows promise in significantly improving inference quality, guided knowledge discovery is a field much in its infancy. Given the subjective nature of interestingness, guidance seems the next logical step in explorative knowledge discovery research, as user-computer synergy maintenance provides the best means to accurately incorporate interestingness within descriptive analysis. The previous research within this field is scarce, as presented in Chapter 5. This section discusses the contribution of this thesis in respect to this previous material.

Previous guided knowledge discovery research addresses guidance at two levels of granularity: dynamic tool selection and guided analysis. Dynamic tool selection (reviewed in Section 6.2) regards the dynamic selection of tools during a knowledge discovery session. This is not directly related to this thesis contributions as it does not enable analysis guidance but facilitates tool selection based upon the results of the previous processing stage.

Guided analysis research to date has been conducted in all three of the common descriptive knowledge discovery tasks: classification, clustering and association mining (reviewed in Sections 6.3 - 6.5). Current guided clustering techniques centered upon research by the MERL group (Anderson *et al.* 2000) in allowing the manual assignment, or constraint, of elements to clusters, while associated research by Aggarwal and Charu (2001) involved the user specification of interesting subspace clusters, from which higher dimensional clusters are automatically derived. The largest body of guided classification research has been undertaken by the Knowledge Discovery in Databases group at the University of Munich and focuses upon the interactive creation of the decision tree, which is subsequently used for predictive classification (Ankerst, Ester & Kriegel 2000).

Prior guided association mining research has three major contributors. Ng *et al.* (1998) propose an architectural extension enabling coarse-grained user involvement within the association mining process. This enables constraint refinement, within analysis, between elementset derivation and inference derivation through the inclusion of user feedback. If the user is satisfied with the quality of V they can instigate inference derivation, else they can refine constraints and

re-generate V. Brin and Page (1998) propose DDM an analysis process that produces a set of intermediate presentations through which the user can prioritise elements in subsequent analysis, through manipulation of a *weight* heuristic. Finally Hidber (1999) proposed CARMA that allows the dynamic adjustment of common functional constraints *support* and *confidence* during analysis.

This thesis significantly extends this research by the presentation of the first generic guided knowledge discovery architecture that, although presented in the context of association mining, can be used to facilitate guidance in all descriptive (or exploratory) tasks. This architecture is novel and generic, effectively merging the current knowledge discovery architecture with the HCI model-view-controller architecture, providing an effective framework for guided analysis. The architecture is non specific in regard to the analysis algorithm, presentation mediums and constraint types used, only requiring that both a process and model view be present to allow for effective analysis interaction.

As a proof-of-concept as to the viability of this guided architecture, GAM has been developed. GAM is superior to previous techniques in its provision of a complete guided association mining system that is flexible, generic and extensible. Based upon comprehensive research into association mining constraints and the additional constraints that guidance will allow, GAM illustrates the architecture's potential to incorporate the dynamic refinement of any constraint type, by including an instance for each constraint class. Furthermore the inclusion of process constraints, enables the dynamic adjustment of the level of interactivity incorporated within the analysis process. This ability to interrupt analysis at will, has not been provided before and addresses an interactive issue raised by Piatetsky-Shapiro, presented below.

> *The perfect data mining tool should have a more automated mode for beginners and a more interactive mode for experts.*
>
> Gregory Piatetysky Shapiro 2002

V.2 Future Direction

The objective of CARV was to develop a presentation technique that allowed the integration of discovered inferences within a hierarchical context to facilitate user interpretation. However for each set of element domain multiple hierarchies can be defined, it might therefore be interesting to develop a presentation technique that allows for the simultaneous presentation of multiple hierarchies and the inferences within them. This would require a supporting analysis algorithm that enabled the discovering of inference within multiple hierarchical contexts and also the use of presentation functionality such as *linking* to facilitate interpretation. This presentation could be realised as a set of simultaneous CARV structures.

Furthermore CARV has demonstrated the effectiveness by which inferences can be presented, using graph based techniques, within specific semantic domains. Further investigation is therefore required to find if the same principles can be applied to the inclusion of temporal and spatial semantics within association analysis. This would be realised as the development of presentation techniques in which the inferences are presented within a context that facilitates the understanding of their temporal or spatial nature.

The objective of MCL is to facilitate user interpretation of frequent elementsets maintained within a lattice by maintaining a smaller representative lattice, through the use of the closed-set concept. Although the objective of MCL was not process optimisation, but lattice reduction, further research regarding comparison against other current incremental association mining algorithms is required to quantify its overall efficiency. This is a priority for further work. Consideration is also being given to implementing a Maximal Frequent Set (MFS) version (see Section 1.3.1), which although resulting in an even more condensed representation will loose the ability for accurate derivation of inference confidence.

GAM at present provides a comprehensive proof-of-concept guided association mining system. Further work is required to make it robust enough for external use. In addition to general strengthening of the tool, three areas of extension present themselves. Firstly the extension of constraint manipulation, at present GAM allows the dynamic refinement of a typical constraint from each constraint class, the extension of its current capabilities to include more powerful constraint

refinement, especially complex filtering, may result in further significant quality improvement. Secondly, GAM currently uses an effective candidate generation algorithm and model views, based upon the prefix-tree structure built during valid elementset discovery. Furthermore thought should be given to the use of pattern growth based analysis techniques within a guided environment, although initial research has suggested their unsuitability (see Chapter 8).

The final extension to GAM is the refinement of heuristic constraint inclusion, allowing the specification of heuristic constraint refinement at specific elementsets, or view nodes. Within the current system constraint relaxation can occur from a particular prefix-tree level, however this can be extended to enable the specification of different thresholds at different elementsets. This new threshold will then be applied to the concept's sub-tree until it is replaced by another.

In regard to the proposed guided architecture three areas of further work present themselves. Firstly the validation of the framework's generic capabilities through its use in the development of both guided clustering and guided classification systems. Secondly, and perhaps more importantly, is the identification and development of a set of generic interaction mappings between the graphical interface and the underlying analysis process, to provide a form of standardisation. Thirdly further consideration needs to be given to the quantification of the benefits that guidance has upon the knowledge discovery process in regard to explorative tasks.

Part VI

Appendices and Bibliography

Appendix A

Miscellaneous Analysis Algorithms

This appendix presents brief descriptions of a couple of miscellaneous analysis algorithms that made a contribution to the research effort. HND was designed to support CARV, providing a dynamic hierarchical analysis environment and furthermore enabling hierarchical monotonic support to reduce high-level concept inferences, while HPtid was used to investigate preliminary ideas about prioritised mining.

Given the current state of research these algorithms now appear rather simple, however they were novel at the time and important to furthering our knowledge. They are included here to provide further indication as to the path travelled, during the evolution of this thesis and because of their mention in Chapter 1.

A.1 HND: Hierarchical Non-monotonic Dynamic Association Analysis

HND is a hierarchical Apriori extension that enables dynamic inference derivation as the elementset model is explored and the variable specification of *support* (interestingness measure) across hierarchical levels. A significant problem with current hierarchical algorithms is that given a support threshold specified in order

to discover concrete inferences, low-level inferences, results in the generation of a plethora of higher-level inferences, as abstract concepts are obviously more common within objects than elements and low-level concepts (Han & Kamber 2001). As a result of this, any presentations (in this case CARV) contain inference overload at the higher concept levels from which little if anything can be interpreted. The theory behind HND was to allow for the specification of a monotonic support, that increased as higher-level concepts were included within the elementset under consideration, thereby reducing the number of valid high-concept elementsets. Although enabling the required functionality, it also resulted in a less efficient algorithm as pruning optimisations, such as those outlined in Chapter 1, could not be incorporated. HND extended Apriori and was paired with CARV, to present our initial dynamic association mining environment in 2002.

As a first step HND incorporates the hierarchical information within the dataset, so that each element is replaced with its hierarchical position. For example, *Rye* is replaced, given an associated hierarchy with the concept *Bread-Loaf-Rye*, improving analysis by locating both data and hierarchical structure in a single source, reducing I/O and simplifying analysis. This process is similar to that presented by Han & Fu in the ML-T* family of hierarchical algorithms (Han & Fu 1995).

ML-T2 algorithm devised by Han and Fu undertakes inter-level analysis, by incorporating the hierarchical semantics during the generation of V_2. This results in set of length 2 valid elementsets, incorporating concepts from all abstraction levels. Subsequent analysis uses regular Apriori, which (being derived from V_2) generates valid elementsets from all levels of abstraction. The hierarchical inclusion during the generation of V_2 is achieved in a top-down fashion, only exploring lower-concepts given the validity of its generating higher-concept elementset. For example, given the hierarchy in Figure 1, the top-level V_2 elementsets, such as *Drink, PowderedProduct*, are first explored. Subsequent V_2 analysis is based upon the valid elementsets discovered within the previous hierarchical level. Given that *Drink, PowderedProduct* is valid, then elementsets such as *Drink,Cocoa*, *Drink,Coffee* and *PowderedProduct, Alcohol* are considered. If *Drink, Coffee* is not valid then V_2 elementsets generated from this are not considered.

This top-down method effectively prunes the exploration of the hierarchical V_2 space, given a static *support* threshold, as given an invalid elementset then lower-concept elementsets derived from this can not be valid either. For example, given that *Drink, Coffee* has $\sigma(12)$, then descendant elementsets such as *Alcohol, Coffee* can have a maximum support of 12. Therefore the hierarchical exploration space is bounded by invalid V_2 elementsets.

The requirement within HND of providing monotonic support cannot incorporate this bounded exploration as support is not static but is non-monotonic given a top-down traversal. Therefore V_2 generation considers all hierarchical permutations of all elements in the construction of V_2. From which point subsequent analysis uses Apriori, deriving and presenting inferences as their generating elementset is discovered. Figure A.1 presents the HND interface.

HND was effective in reducing top-level inference overload in CARV, reducing the number of higher level inferences produced. However this initial research was subsequently optimised in HPtid, through the use of vertical organisation and extended to allow a constrained form of prioritised analysis, which was later extended and implemented within GAM.

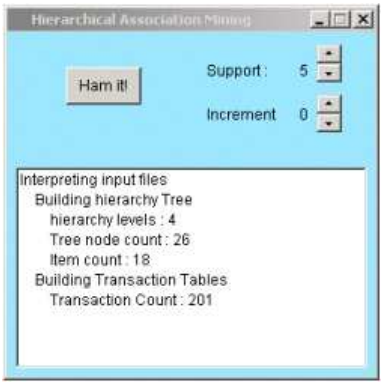


Figure A.1: Hierarchical Association Mining

A.2 HPtid: Hierarchical Prioritised TidList Association Analysis

HPtid optimises and extends HND through the use of vertical organisation and the enabling of prioritised analysis. The use of a vertical organisation, provides an element centric approach, whereby each higher-level concept could be considered as an element in its own right, with an associated tidList. This simplifies non-prioritised analysis to AprioriTid, see Chapter 1, after the vertical tidList structure has been constructed in memory, including hierarchical concepts. Further constraint can be included within the generation of V_2 to ensure that no elementsets are generated in which one element is the ancestor of the other.

The main goal of HPtid was to enable a preliminary investigation into concept focus, where focus or prioritisation (see Chapter 8) enables the user to focus exploration upon a specific area of interest during analysis. However unlike GAM, HPtid only enabled the static prioritisation of elements, whereby the focus concepts needed to be specified prior to analysis and could therefore only incorporate elements. As a HND extension, HPtid incorporates hierarchical semantics and dynamic presentation, enabling the prioritisation of abstract concepts and the subsequent presentation of focus derived inferences, before analysis of the remaining exploration space is undertaken. The HPtid interface is shown in Figure A.2, which was paired with CARV to provide a dynamic association mining system with limited concept prioritisation.

Vertical organisation facilitates prioritisation because of its element-centric layout through which element based analysis constraint can be easily incorporated. This is achieved by forcing a partial V_1 ordering, whereby the prioritised elements are placed at the head of the V_1 list. Prioritised analysis is subsequently undertaken by iterating over the subsequent V_1 members for each prioritised element, generating prioritised V_2, or P_2, where P_2 consists of a a list of length 2 elementsets, each containing at least one prioritised concept, and their relevant tidLists. Generation of subsequent P, is equivalent to that of AprioriTid (Agrawal & Srikant 1994) incorporating tidList projection.

The inclusion of dynamic presentation within HPtid, achieved through dynamic inference derivation, may however require knowledge of a non-prioritised elementset. For example, prioritisation of g and the subsequent discovery of the valid elementset bgt, will result in the derivation of the inference $b \Rightarrow gt$, however gt does not contain a prioritised concept and hence its presence has not been calculated. In these cases the required elementset's presence is calculated from V_1 through the intersection of tidLists, the result of which is used to calculate the confidence of the inference. Furthermore these valid non-prioritised elementsets can be stored within their appropriate V_κ listing for further reference.

Once prioritised analysis is complete HPtid pauses analysis to allow the user to explore the presentation. If specified the algorithm then resumes analysis from the first non-prioritised element within the V_1.

Figure A.2: Prioritisation of *large snapper* and *softdrink* using HPtid

Bibliography

Abraham, T. & Roddick, J. F. (1999), 'Survey of spatio-temporal databases', *Geoinformatica* **3**(1), 61–99.

Aggarwal, C. C. (2001), A human-computer cooperative system for effective high dimensional clustering, *in* 'Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'01)', San Francisco, CA,USA, pp. 221–226.

Aggarwal, C. C. (2002), 'Towards effective and interpretable data mining by visual interaction', *SIGKDD Explorations* **3**(2), 11–22.

Agrawal, R. C., Aggrawal, C. C. & Prasad, V. (2000), Depth first generation of long patterns, *in* 'Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', ACM Press, Boston,Massachusetts, pp. 108–118.

Agrawal, R. C., Aggrawal, C. C. & Prasad, V. V. V. (1999), A tree projection algorithm for generation of frequent itemsets, *in* 'Proceedings of High Performance Data Mining Workshop', ACM Press, Puerto Rico.

Agrawal, R., Imielinski, T. & Swami, A. (1993), Mining association rules between sets of items in large databases, *in* '1993 ACM SIGMOD International Conference Management of Data', ACM Press, Washington D.C. U.S.A, pp. 207–216.

Agrawal, R. & Srikant, R. (1994), Fast algorithms for mining association rules, *in* 'Proceedings of the 20th International Conference on Very Large Data Bases', Morgan Kaufmann Publishers Inc., Santiago, Chile, pp. 487–499.

Agrawal, R. & Srikant, R. (1995), Mining sequential patterns, *in* P. S. Yu & A. S. P. Chen, eds, 'Proceedings of the 11th International Conference on Data Engineering(ICDE'95)', IEEE Computer Society Press, Taipei,Taiwan, pp. 3–14.

Ale, J. M. & Rossi, G. H. (2000), An approach to discovering temporal association rules, *in* 'Proceedings of the 2000 ACM Symposium on Applied Computing (SAC)', Vol. 1, ACM Press, Villa Olmo, Como, Italy, pp. 294–300.

Allen, J. F. (1983), 'Maintaining knowledge about temporal intervals', *Communications of the ACM* **26**(11), 832–843.

Amant, R. S. & Cohen, P. R. (1997), Evaluation of a semi-autonomous assistant for exploratory data analysis, *in* 'Proceedings of the first international conference on Autonomous agents', ACM Press, Marina del Rey, California, United States, pp. 355–362.

Anderson, D., Anderson, E., Lesh, N., Marks, J., Perlin, K., Ratajczak, D. & Ryall, K. (2000), Human guided simple search: combining information visualization and heuristic search, *in* 'Proc. of the workshop on new paradigms in information visualization and manipulation. In conjunction with the eighth ACM international conference on Information and Knowledge Management', ACM Press, Kansas City, MO, pp. 21–25.

Anderson, P., Smith, R. & Zhang, Z. (1996), Frustrum: A novel distortion oriented display for demanding applications, *in* 'Proceedings of the 3rd SPIE Conference on Visual Data Exploration and Analysis', Vol. 2656, IEEE Press, San Jose, California, USA, pp. 150–156.

Ankerst, M. (2001), Human involvement and interactivity of the next generation's data mining tools, *in* 'ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery', Santa Barbara, CA.

Ankerst, M. (2002), The perfect data mining tool: Automated or interactive?, *in* 'Panel at ACM SIGKDD02', Edmonton, Canada.

Ankerst, M., Ester, M. & Kriegel, H.-P. (2000), Towards an effective cooperation of the user and the computer for classification, *in* 'Proc. 6th Int. Conf. on Knowledge Discovery and Data Mining (KDD'2000)', Boston, MA, pp. 179–188.

Ayan, N. F., Tansel, A. U. & Arkun, E. (1999), An efficient algorithm to update large itemsets with early pruning, *in* 'Proceedings of the fifth International Conference on Knowledge Discovery and Data Mining (SIGKDD'99).', ACM Press, San Diego, CA USA, pp. 287–291.

Baker, C. A. H., Carpendale, M. S. T., Prusinkiewicz, P. & Surette, M. G. (2002), Genevis: Visualization tools for genetic regulatory network dynamics, *in*

'Proceedings of the conference on Visualization'02', IEEE Press, Boston, Massachusetts, pp. 243–250.

Barass, S. (1995), Personify: a toolkit for perceptually meaningful sonification, *in* 'Australian Computer Music Conference ACMA'95'.

Bastide, Y., Taouil, R., Pasquier, N., Stumme, G. & Lakhal, L. (2000), 'Mining frequent patterns with counting inference', *SIGKDD Explorations* **2**(2), 66–75.

Battista, G. d., Eades, P., Tamassia, R. & Tollis, I. G. (1999), *Graph Drawing: Algorithms for the Visualization of Graphs*, Prentice Hall, Englewood, NJ.

Bayardo, R. & Agrawal, R. (1999), Mining the most interesting rules, *in* S.Chaudhuri & D.Madigan, eds, 'Fifth International Conference on Knowledge Discovery and Data Mining', ACM Press, San Diego, CA, USA, pp. 145–154.

Bayardo, R. J. (1998), Efficiently mining long patterns from databases, *in* 'Prococeedings of the International Conference on Management on Data (SIGMOD)', ACM Press, Seattle, Washington, United States, pp. 85–93.

Bettini, C., Wang, X. S., Jajodia, S. & Lin, J.-L. (1998), 'Discovering temporal relationships with multiple granularities in time sequences', *IEEE Transactions on Knowledge and Data Engineering* **10**(2), 222–237.

Boardman, R. (2000), Bubble trees: Visualization of hierarchical information trees, *in* 'Proceedings of the Conference on Human Factors in Computing Systems (CHI'00)', ACM Press, The Hague.

Bodon, F. (2003), A fast apriori implementation, *in* B. Goethals & M. J. Zaki, eds, 'Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03)', Vol. 90 of *CEUR Workshop Proceedings*, IEEE Press, Melbourne, Florida, USA.

Bolton, R. J. & Adams, N. M. (2003), An iterative hypothesis-testing strategy for pattern discovery, *in* 'Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD2003)', Washington D.C., USA, pp. 49–58.

Boulicant, J.-F., Bykowski, A. & Rigotti, C. (2001), 'Free-sets: A condensed representation of boolean data for the approximation of frequency queries', *Data Mining and Knowledge Discovery Journal* **7**(1), 5–22.

Boulicaut, J.-F. & Jeudy, B. (2001), Mining free itemsets under constraints, *in* 'Proceedings of the International Database Engineering and Applications Symposium', IEEE, Grenoble, France, pp. 322–329.

Brin, S., Motwani, R. & Silverstein, C. (1997), Beyond market basket: Generalizing association rules, *in* 'Proceedings of the ACM SIGMOD International Conference on Management of Data', ACM-Press, Tucson, Arizona,USA, pp. 265–276.

Brin, S., Motwani, R., Ullman, J. D. & Tsur, S. (1997), 'Dynamic itemset counting and implication rules for market basket data', *SIGMOD Record (ACM Special interest group on the Management of Data)* **26**(2), 255–276.

Brin, S. & Page, L. (1999), Dynamic data mining: Exploring large rule spaces by sampling, Technical Report SIDL-WP-1999-0122, Stanford University.

Bucila, C., Gehrke, J., Kifer, D. & White, W. (2002), Dualminer: A dual-pruning algorithm for itemsets with constraints, *in* 'The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD02)', Edmonton, Alberta, Canada.

Burdick, D., Calimlim, M. & Gehrke, J. (2001), Mafia: A maximal frequent itemset algorithm for transactional databases, *in* 'The 17th International Conference on Data Engineering', IEEE Press, Heidelburg, Germany, pp. 443–452.

Buxton, W. (1986), Chunking and phrasing and the design of human computer dialogues, *in* 'Proceedings of the IFIP 10th World Computer Congress', Morgan Kaufmann Publishers Inc., Dublin, Ireland, pp. 475–480.

Bykowski, A. & Rigotti, C. (2001), A condensed representation to find frequent patterns, *in* 'Proceedings of the 20th ACM SIGMOD Symposium on Principles of Database Systems', ACM Press, Santa Barbara, California, pp. 267–273.

Calders, T. & Goethals, B. (2002), Mining all non-derivable frequent itemsets, *in* T. Elomaa, H. Mannila & H. Toivonen, eds, 'Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery', Vol. 2431, Springer-Verlag, Helsinki, Finlanmd, pp. 74–85.

Carriere, J. & Kazman, R. (1995), Interacting with huge hierarchies: Beyond cone trees, *in* 'InfoViz'95, IEEE Symposium on Information Visualisation', IEEE Computer Society Press, Atlanta, Georgia, pp. 74–78.

Ceglar, A., Roddick, J. F., Calder, P. & Rainsford, C. P. (2004), 'Visualising hierarchical associations', *Knowledge and Information Systems (to appear).*

Chang, C.-H. & Yang, S.-H. (2003), Enhancing swf for incremental association mining by itemset maintenance, *in* 'Proceedings of the seventh Pacific Asia conference on Knowledge Discovery and Data Mining PAKDD'03', Seoul, Korea.

Chen, B., Haas, P. & Scheuermann, P. (2002), A new two-phase sampling based algorithm for discovering association rules, *in* 'Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.', ACM Press, Edmonton,Alberta,Canada, pp. 462–468.

Cheung, D. W., Han, J., Ng, V. T. & Wong, C. (1996), Maintenence of discovered association rules in large databases: An incremental updating technique, *in* 'Proceedings of the Twelfth International Conference on Data Engineering (ICDE'96)', IEEE Computer Society Press, New Orleans, Louisiana, pp. 106–114.

Cheung, D. W.-L., Lee, S. D. & Kao, B. (1997), A general increment technique for maintaining discovered association rules, *in* 'Proceedings of the fifth International Conference on Database Systems of Advanced Applications (DASFAA)', Melbourne, Australia, pp. 185–194.

Cheung, W. & Zaiane, O. R. (2003), Incremental mining of frequent patterns without candidate generation or support constraint, *in* 'The seventh International database Engineering and Applications Symposium (IDEAS'03)', Hong Kong, China.

Choudhuri, S., Datar, M., Motwani, R. & Narasayya, V. (2001), Overcoming limitations of sampling for aggregation queries, *in* 'Proceedings of the 17th International Conference on Data Engineering, ICDE', IEEE press, Heidelburg,Germany, pp. 534–542.

Chu, H. K. & Wong, M. H. (1998), Interactive data analysis on numeric-data, *in* 'Proceedings of the 1999 International Symposium on Database Engineering & Applications', IEEE Computer Society, Montreal, Canada, p. 226.

Cong, G. & Liu, B. (2002), Speed-up iterative frequent itemset mining with constraint changes, *in* 'Proceedings of the IEEE International Conference on Data Mining, (ICDM 2002).', Maebashi City, Japan, pp. 107–114.

Deshaspe, L. & Toivonen, H. (1998), Frequent query discovery: A unifying approach to association rule mining, Technical Report CW-258, Department of Computer Science, Katholieke Universiteit Leuven, Belgium.

Dunkel, B. & Soparkar, N. (1999), Data organization and access for efficient data mining, *in* 'Proceeding of the 15th International Conference on Data Engineering', IEEE, Sydney, Australia, pp. 522–532.

Eades, P. (1992), 'Drawing free trees', *Bulletin of the Institute of Combinatorics and its Applications* **5**, 10–36.

Engels, R., Lindner, G. & Studer, R. (1997), A guided tour through the data mining jungle, *in* D.Pregibon, D. Heckerman & H.Manilla, eds, 'Proceedings of the 3nd International Conference on Knowledge Discovery in Databases (KDD-97)', AAAI Press, Newport Beach, CA, pp. 163–166.

Evfimievski, A., Srikant, R., Agrawal, R. & Gehrke, J. (2002), Privacy preserving mining of association rules, *in* 'Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02)', ACM Press, Edmonton, Alberta, Canada, pp. 217–228.

Freitas, A. (1999), 'On rule interestingness measures', *Knowledge Based Systems* **12**(5-6), 309–315.

Fule, P. & Roddick, J. (2004), Detecting privacy and ethical sensitivity in data mining results., *in* V. Estivill-Castro, ed., 'Proceedings of the 27th Australasian Computer Science Conference (ACSC2004).', CRPIT, Dunedin, New Zealand, pp. 159–166.

Furnas, G. W. (1986), Generalized fisheye views, *in* 'Conference Proceedings of Human Factors in Computing Systems (CHI'86)', ACM Press, New York, USA, pp. 16–23.

Ganter, G. & Wille, R. (1999), *Formal Concept Analysis: Mathematical Foundations*, Springer Verlag.

Ganti, V., Gehrke, J. & Ramakrishnan, R. (2001), 'Demon: Mining and monitoring evolving data', *IEEE Transactions on Knowledge and Data Engineering (TKDE)* **13**(1), 50–63.

Gardarin, G., Pucheral, P. & Wu, F. (1998), Bitmap based algorithms for mining association rules, *in* 'Proceedings of the 14th Bases de Donnes Avances (BDA'98)', Springer Verlag, Hammamet, Tunisia, pp. 157–176.

Garner, W. R., Hake, H. & Erikson, C. W. (1956), 'Operationism and the concept of perception', *Psychological Review* **63**, 149–159.

Gibbons, P. (2001), Distinct sampling for highly-accurate answers to distinct values queries and event reports, *in* 'Proceedings of 27th International Conference on VLDB Conference', Morgan Kaufmann, Rome Italy, pp. 541–550.

Godin, R., Missaoui, R., Huchard, M. & Napoli, A. (2004), 'Galicia - galois lattice interactive constructor.'.

Goethals, B. & Van den Bussche, J. (2000), On supporting interactive association rule mining, *in* Y. Kambayashi, M. K. Mohania & A. M. Tjoa, eds, 'Data Warehousing and Knowledge Discovery, Second International Conference (DaWaK)', Springer, London, UK, pp. 307–316.

Goethals, B. & Zaki, M. J. (2003), Advances in frequent itemset mining implementations: Introduction to fimi03, *in* B. Goethals & M. J. Zaki, eds, 'Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03)', IEEE Press, Melbourne, Florida, USA.

Gopalan, R. P. & Sucahyo, Y. G. (2002), Itl-mine: Mining frequent itemsets more efficiently, *in* L. Wang, S. Halgamuge & X. Yao, eds, 'Proceedings of the 2002 Internatiional Conference on Fuzzy Systems and Knowledge Discovery', Vol. 1, Springer Verlag, Singapore, pp. 167–172.

Gouda, K. & Zaki, M. J. (2001), Efficiently mining maximal fequent itemsets, *in* 'IEEE International Conference on Data Mining', IEEE Press, San Jose.

Goulbourne, G., Coenen, F. & Leng, P. (2000), 'Algorithms for computing association rules using a partial support tree', *Knowledge Based Systems* **13**(2-3), 141–149.

Grahne, G. & Zhu, J. (2003*a*), Efficiently using prefix-trees in mining frequent itemsets, *in* B. Goethals & M. J. Zaki, eds, 'Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03)', Vol. 90, IEEE Press, Melbourne, Florida, USA.

Grahne, G. & Zhu, J. (2003*b*), High performance mining of maximal frequent itemsets, *in* 'Proceedings of the 6th SIAM International Workshop on High Performance Data Mining (HPDM '03)', San Francisco, CA,USA.

Gunopulos, D., Mannila, H. & Saluja, S. (1997), Discovering all most specific sentences by randomised algorithms extended abstract, *in* F.Afrati & P. Kolaitis, eds, 'Proceedings of the 6th International Conference on Database Theory', Springer Verlag, Delphi, Greece, pp. 251–229.

Han, J., Cai, Y. & Cerone, N. (1992), Knowledge discovery in databases: an attribute-oriented approach, *in* 'Proceedings of the 18th International Conference on Very Large Databases (VLDB)', pp. 547–559.

Han, J. & Fu, Y. (1994), Dynamic generation and refinement of concept hierarchies for knowledge discovery in databases, *in* 'Proceedings AAAI'94 Workshop on Knowledge Discovery in Databases (KDD94)', ACM Press, Seattle, WA, pp. 157–168.

Han, J. & Fu, Y. (1995), Discovery of multiple-level association rules from large databases, *in* '21st International Conference on Very Large Databases (VLDB'95)', Morgan Kaufmann Publishers Inc., Zurich, Switzerland.

Han, J., Fu, Y., Wang, W., Koperski, K. & Zaiane, O. (1996), DMQL: A data mining query language for relational databases, *in* 'Proceedings of the 1996 SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'96)', Montreal, Canada, pp. 27–33.

Han, J. & Kamber, M. (2001), *Data Mining: Concepts and Techniques*, Morgan Kaufmann, San Francisco California U.S.A.

Han, J., Lakshmanan, L. V. S. & Ng, R. T. (1999), 'Constraint-based, multidimensional data mining', *IEEE Computer* **32**(8), 46–50.

Han, J. & Pe, J. (2000), 'Mining frequent patterns by pattern growth: Methodology and implications', *ACM SIGKDD Explorations* **2**(2), 14 – 20.

Han, J., Pei, J. & Yin, Y. (2000), Mining frequent patterns without candidate generation, *in* W. Chen, J. Naughton & P. Bernstein, eds, '2000 ACM SIGMOD International Conference on Management of Data.l C', ACM Press, pp. 1–12.

Hao, M. C., Dayal, U., Hsu, M., Sprenger, T. & Gross, M. H. (2001), Visualization of directed associations in e-commerce transaction data, *in* 'Proceedings of VisSym'01, Joint Eurographics - IEEE TCVG Symposium on Visualization', IEEE Press, Ascona, Switzerland, pp. 185–192.

Herman, I., Melancon, G., de Ruiter, M. M. & Delest, M. (2000), 'Latour - a tree visualization system', *Lecture Notes in Computer Science* **1731**, 392–404.

Herman, I., Melancon, G. & Marshall, M. S. (2000), 'Graph visualisation and navigation in information visualization: a survey', *IEEE Transactions on Visualizations and Computer Graphics* **6**(1), 24–43.

Hetzler, B., Harris, W. M., Havre, S. & Whitney, P. (1998), Visualizing the full spectrum of document relationships, *in* 'Proceedings of the 5th International Society for Knowledge Organisation Conference (ISKO)', Springer Verlag, San Fransisco, California, pp. 168–175.

Hidber, C. (1999), Online association rule mining, *in* 'ACM SIGMOD Intl. Conf. on Management of Data', ACM Press, pp. 145–156.

Hilderman, R. J. & Hamilton, H. J. (1999), Knowledge discovery and interestingness measures: A survey, Technical Report CS 99-04, Department of Computer Science, University of Regina.

Hinneburg, A., Keim, D. A. & Wawryniuk, M. (1999), 'HD_Eye: Visual mining of high dimensional data', *IEEE Computer Graphics and Applications* **19**(5), 22–31.

Hipp, J., Guntzer, U. & Nakhaeizadeh, G. (2000), Mining association rules: Deriving a superior algorithm by analysing today's approaches, *in* 'Proceedings of the 4th European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'00)', Springer Verlag, Lyon, France, pp. 159–168.

Hipp, J., Myka, A., Wirth, R. & Guntzer, U. (1998), A new algorithm for faster mining of generalised association rules, *in* 'Proceedings of the 2nd Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'98)', Springer Verlag, Nantes, France, pp. 74–82.

Hofman, H., Siebes, A. & Wilhelm, A. (2000), Visualizing association rules with interactive mosaic plots, *in* 'Proceeding of KDD2000', ACM, Boston, MA USA, pp. 227–235.

Houtsma, M. & Swami, A. (1993), Set oriented mining of association rules, Technical Report RJ 9567, IBM Almaden Research Centre.

Imielinski, T. & Virmani, A. (1999), 'MSQL: A query language for database mining', *Journal of Data Mining and Knowledge Discovery* **3**(4), 373–408.

Jensen, D., Dong, Y., Staudt Legner, B., McCall, E. K., Osterweil, L. J., Sutton Jr., S. M. & Wise, A. (1999), Coordinating agent activities in knowledge discovery processes, *in* 'Proceedings of the international joint conference on

Work activities coordination and collaboration', ACM Press, San Francisco, California, United States, pp. 137–146.

Jeudy, B. & Boulicaut, J. F. (2002), Using condensed representations for interactive association rule mining, *in* T. Elomaa, H. Mannila & H. Toivonen, eds, 'In Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery', Vol. 2431, Springer, Helsinki, Finland, pp. 225–236.

John, B. E., Rosenbloom, P. S. & Newell, A. (1985), A theory of stimulus-response compatibility applied to human computer interaction, *in* 'Proceedings of the Conference on Human Factors in Computing Systems (CHI'85)', ACM Press, pp. 213–230.

Johnson, B. & Schneiderman, B. (1991), Tree-maps: a space-filling approach to the visualization of hierarchical information structures, *in* 'IEEE Visualization'91', IEEE Computer Society Press, pp. 275–282.

Kivinen, J. & Mannila, H. (1994), The power of sampling in knowledge discovery, *in* 'Proceedings of the 13th ACM AIGACT-SIGMOD-SIGART Symposium on the Principles of Database Systems (PODS'94)', ACM Press, Minneapolis,NM,USA, pp. 77–85.

Klemettinen, M., Mannila, H., Ronkainen, T. & Verkano, A. (1994), Finding interesting rules from large sets of discovered association rules, *in* N. R. Adam, B. K. Bhargava & Y. Yesha, eds, 'Third International Conference on Information and Knowledge Management (CIKM'94)', ACM Press, Gaitherburg Maryland USA, pp. 401–407.

Klemettinen, M., Mannila, H. & Toivonen, H. (1996), Interactive exploration of discovered knowledge: A methodology for interaction, and usability studies., Technical Report Report C-1996-3, Department of Computer Science, University of Helsinki,.

Klemettinen, M., Mannila, H. & Toivonen, H. (1997), A data mining methodology and its application to semi-automatic knowledge acquisition, *in* 'Proceedings of the 8th International Workshop on Database and Expert Systems Applications', IEEE Press, pp. 67–677.

Klemmer, E. T. & Frick, F. C. (1953), 'Assimilation of information from dot and matrix patterns', *Experimental Psychology* **45**, 15–19.

Koedinger, K. R. (1992), Emergent properties and structural constraints: Advantages of diagrammatic representations for reasoning and learning, *in* 'AAAI Spring Symposia on Reasoning with Diagrammatic Representations', AAAI Press, Stanford University.

Koike, H. & Yoshihara, H. (1993), Fractal approaches for visualizing huge hierarchies, *in* E. P. Gilbert & K. A. Olsen, eds, 'IEEE Symposium on Visual Languages VL'93', IEEE Computer Society, pp. 55–60.

Koperski, K. & Han, J. (1995), Discovery of spatial association rules in geographic information databases, *in* 'Proceedings of the 4th International Symposium on Large Spatial Databases (SSD'95)', Springer Verlag, Portland, Maine, pp. 47–66.

Kosters, W. A. & Pijls, W. (2003), Apriori, a depth first implementation, *in* B. Goethals & M. J. Zaki, eds, 'Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03)', IEEE Press, Melbourne, Florida, USA.

Krasner, G. & Pope, S. (1988), 'A cookbook for using the model-view-controller user interface paradigm in smalltalk', *Journal of Object Oriented Programming* **1**(3), 26–49.

Kreuseler, M. & Schuman, H. (1999), Information visualization using a new focus + context technique in combination with dynamic clustering of information space, *in* 'New Paradigms in Information Visualization and Manipulation', Springer Verlag, Kansas City, Missouri, pp. 1–5.

Lakshmanan, L. V. S., Ng, R., Han, J. & Pang, A. (1999), Optimization of constrained frequent set queries with 2-variable constraints, *in* 'Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'99)', ACM Press, Philadephia, Pennsylvania, USA, pp. pages 157–168.

Lee, C. H., Lin, C. R. & Chen, M. S. (2001), Sliding window filtering: An efficient algorithm for incremental mining, *in* 'Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM'01)', ACM, Atlanta, Georgia, USA.

Lesh, N., Marks, J. & Patrignani, M. (2000), Interactive partitioning, Technical report, Mitsubishi Electronic Research Laboratory.

Li, Y., Ning, P., Wang, X. & Jajodia, S. (2001), Discovering calendar-based temporal association rules, *in* 'Proceedings of the 8th Symposium on Temporal Representation and Reasoning (TIME'01)', IEEE Press, Cividale, Italy, pp. 111–118.

Lin, D. I. & Kedem, Z. M. (1998), Pincer search: A new algorithm for discovering the maximum frequent set, *in* 'Proceeding of the 6th International Conference on Extending Database Technology (EDBT'98)', Springer Verlag, Valencia, Spain.

Lin, J. L. & Dunham, M. H. (1998), Mining association rules: Anti skew algorithms, *in* 'Proceedings aof the 14th International Conference on Data Engineering', IEEE Computer Society Press, Orlando, Florida, USA, pp. 486–493.

Liu, B., Hsu, W., Chen, S. & Ma, Y. (2000), 'Analyzing the subjective interestingness of association rules', *IEEE Intelligent Systems* **15**(5), 47–55.

Liu, B., Hsu, W. & Ma, Y. (1999), Mining association rules with multiple minimum supports, *in* 'Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99)', ACM Press., San Diego, CA, USA, pp. 337–341.

Liu, B., Hsu, W., Mun, L. & Lee, H. (1999), 'Finding interesting patterns using user expectations', *IEEE Transactions on Knowledge and Data Engineering* **11**(6), 817–832.

Liu, J., Pan, Y., Wang, K. & Han, J. (2002), Mining frequent item sets by opportunistic projection, *in* 'Proceedings of Knowledge Discovery in Databases', Vol. 31, ACM Press, Edmonton, Canada, pp. 97–102.

Livingston, G. R., Rosenberg, J. M. & Buchanan, B. G. (2001*a*), Closing the loop: an agenda - and justification-based framework for selecting the next discovery task to perform., *in* 'Proceedings of the 2001 IEEE International. Conference on Data Mining (ICDM)', IEEE Computer Society Press., San Jose, CA., pp. 385–392.

Livingston, G. R., Rosenberg, J. M. & Buchanan, B. G. (2001*b*), Closing the loop: Heuristics for autonomous discovery., *in* 'Proceedings of the 2001 IEEE International. Conference on Data Mining (ICDM)', IEEE Computer Society Press., San Jose, CA., pp. 393–400.

Lucchese, C., Orlando, S., Palmerini, P., Perego, R. & Silvestri, F. (2003), kdci: a multi-strategy algorithm for mining frequent sets, *in* B. Goethals & M. J. Zaki, eds, 'Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03)', Vol. 90, IEEE Press, Melbourne, Florida, USA.

MacKinley, J., Robertson, G. & Card, S. (1991), The perspective wall: Detail and context smoothly integrated, *in* 'Proceedings of International Conference of Human Factors in Computing Systems (CHI'91)', Morgan Kaufmann Publishers Inc., New York, USA, pp. 173–179.

Mannila, H., Toivonen, H. & Verkamo, A. I. (1994), Efficient algorithms for discovering association rules, *in* U. M. Fayyad & R. Uthurusamy, eds, 'Proceedings of Knowledge Discovery in Databases', AAAI Press, Seattle, Washington, pp. 181–192.

Mannila, H., Toivonen, H. & Verkamo, A. I. (1997), 'Discovery of frequent episodes in event sequences', *Data Mining and Knowledge Discovery (DMKD)* **1**(3), 259–289.

Mao, R. (2001), Adaptive-FP: An Efficient and Effective Method for Multi-Level Multi-Dimensional Frequent Pattern Mining, PhD thesis, Simon Fraser University.

Meo, R., Psaila, G. & Ceri, S. (1996), A new sql-like operator for mining association rules, *in* 'Proceedings of the 22nd International Conference on Very Large Data Bases VLDB'96,', Morgan Kaufmann, Mumbay, India, pp. 122–133.

Miller, G. A. (1956), 'The magic number seven,plus or minus two: Some limits on our capacity for processing information', *Psychological Review* **63**, 81–97.

Miller, H. J. & Han, J. (2001), *Geographic Data Mining and Knowledge Discovery*, Taylor & Francis, Inc.

Mueller, A. (1995), Fast sequential and parallel algorithms for association rule mining: A comparison, Technical Report CS-TR-3515, Department of Computer Science, University of MaryLand -College Park, College Park, MD.

Nag, B., Deshpande, P. M. & DeWitt, D. J. (1999), Using a knowledge cache for interactive discovery of association rules, *in* 'Proceddings of KDD99', ACM Press, San Deigo Ca USA, pp. 244–253.

Nascimento, H. A. & Eades, P. (2001), Interactive graph clustering based upon user hints, *in* 'Proceedings of the Second International Workshop on Soft Computing Applied to Software Engineering', Springer Verlag, Enschede, The Netherlands.

Ng, R., Lakshmanan, L., Han, J. & Pang, A. (1998), Exploratory mining and pruning optimizations of constrained association rules, *in* 'Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems', ACM Press, Seattle, Washington, pp. 13–24.

Nielson, J. A. (1992), A layered interaction analysis of direct manipulation, Technical Report Tech. Rep JN-1990-7.2, Dept. of Computer Science, Tech. Univ. of Denmark,.

Omiecinski, E. & Savasere, A. (1998), Efficient mining of association rules in large dynamic databases, *in* 'Proceedings of 16th British National Conference on Databases (BNCOD'98)', Cardiff, Wales, UK, pp. 49–63.

Ong, K. H., Ong, K. L., Ng, W. K. & Lim, E. P. (2002), Crystalclear: Active visualization of association rules, *in* 'International Workshop on Active Mining (AM-2002) in Conjunction with the IEEE International Conference on Data Mining (ICDN'02)', IEEE Press, Maebashi City, Japan.

Ong, K. L., Ng, W. K. & Lim, E. P. (2001), Large mining multi-level rules with recurrent items using fp-tree, *in* 'Proceedings of the 3rd IEEE Conference on Information, Communications and Signal processing (ICICS'2001)', Springer Verlag, Singapore.

Orlando, S., Palmerini, P. & Perego, R. (2001*a*), DCI: a hybrid algorithm for frequent itemset counting, Technical Report CS-01-9-2001, Dip. di Informatica, Universita Ca Foscari.

Orlando, S., Palmerini, P. & Perego, R. (2001*b*), Enhancing the apriori algorithm for frequent set counting, *in* Y. Kambayashi, W. Winiwarter & M. Arikawa, eds, 'International Conference on Data Warehousing and Knowledge Discopvery', Springer-Verlag, Munich, Germany, pp. 71–82.

Ortega, M., Chakrabarti, K. & Mehrotra, S. (2003), Efficient evaluation of relevance feedback for multidimensional all-pairs retrieval, *in* 'Proceedings of the 2003 ACM symposium on Applied computing', ACM Press, Melbourne, Florida, pp. 847–852.

Ozel, S. A. & Guvenir, H. A. (2001), An algorithm for mining association rules using perfect hashing and database pruning, *in* A. Acan, I. Aybay & M. Salamah, eds, 'Proceedings of the Tenth Turkish Symposium on Artificial Intelligence and Neural Networks', Springer Verlag, Gazimagusa, T.R.N.C., pp. 257–264.

Padmanabhan, B. & Tuzhilin, A. (2000), Small is beautiful: Discovering the minimal set of unexpected patterns, *in* R. Ramakrishnan, S. Stolfo, R. Bayardo & I. Parsa, eds, 'Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD2000', ACM Press, New York, USA, pp. 54–63.

Padmanabhan, B. & Tuzhilin, A. (2002), 'Knowledge refinement based on the discovery of unexpected patterns in data mining', *Decision Support Systems* **33**(3), 309–321.

Park, J. S., Chen, M. S. & Yu, P. S. (1997), 'Using a hash-based method with transaction trimming and database scan reduction for mining association rules', *IEEE Transactions on Knowledge and Data Engineering* **9**(5), 813–825.

Pasquier, N., Bastide, Y., Taouil, R. & Lakhal, L. (1999*a*), Closed set based discovery of small covers for association rules, *in* 'Proceedings of the 15th Conference on Advanced Databases', Springer Verlag, Bordeaux, France, pp. 361–381.

Pasquier, N., Bastide, Y., Taouil, R. & Lakhal, L. (1999*b*), Discovering frequent closed itemsets for association rules, *in* 'Proceedings of the 7th International Conference on Database Theory (ICDT99)', Springer Verlag, Jerusalem, Israel, pp. 398–416.

Pasquier, N., Bastide, Y., Taouil, R. & Lakhal, L. (1999*c*), 'Efficient mining of association rules using closed itemset lattices', *Information Systems* **24**(1), 25–46.

Pei, J., Han, J. & Lakshmanan, L. V. (2001), Mining frequent itemsets with convertible constraints, *in* 'Proceedings of the 17th International Conference on Data Engineering ICDE'01', IEEE Computer Society Press., Heidelberg, Germany, pp. 433–442.

Pei, J., Han, J., Lu, H., Nishio, S., Tang, S. & Yang, D. (2001), H-mine: Hyper-structure mining of frequent patterns in large databases, *in* 'Proceedings

of the 2001 International Conference on Data Mining (ICDM)', IEEE, San Jose, California, pp. 31–39.

Pei, J., Han, J. & Mao, R. (2000), CLOSET: An efficient algorithm for mining frequent closed itemsets, *in* 'Proceedings of ACM SIGMOD International Workshop on Data Mining', ACM Press, Dallas, Texas, pp. 21–30.

Pietracaprina, A. & Zandolin, D. (2003), Mining frequent itemsets using patricia tries, *in* B. Goethals & M. J. Zaki, eds, 'Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03)', Melbourne, Florida, USA.

Pijls, W. & Bioch, J. C. (1999), Mining frequent itemsets in memory resident databases, *in* E. Postma & M. Gyssens, eds, 'Proceedings of the eleventh Belgium-Netherlands Conference on Artificial Intelligence (BNAIC'99)', Springer Verlag, Kasteel Vaeshartelt, Maastricht, The Netherlands, pp. 75–82.

Poulet, F. (2002), 'Full view: A visual data mining environment', *International Journal of Image and Graphics* **2**(1), 127–143.

Purchase, H. C. (1998), Which aesthetic has the greatest effect upon human understanding?, *in* 'Proceedings of the Symposium on Graph Drawing GD'97', Springer-Verlag, pp. 248–261.

Rabejij, D. R. (2001), Greedy random: A novel algorithm for vehicle routing optimisation, *in* '39th National Junior Science and Humanities Symposium: Powerpoint presentation', Orlando, Florida.

Raghavan, V. & Hafez, A. (2000), Dynamic data mining, *in* 'Industrial and Engineering Applications of Artificial Intelligence and Expert Systems', Lecture Notes in Computer Science, Springer-Verlag, pp. 220–229.

Rainsford, C. P. & Roddick, J. F. (1999), Adding temporal semantics to association rules, *in* J. Zytkow & J. Rauch, eds, 'Proceedings of the 3rd European Conference on Principles and Practice of Knowledge Discovery in Databases, PKDD'99', Springer Verlag, Prague, Czech Republic, pp. 504–509.

Rainsford, C. & Roddick, J. (2000), Visualisation of temporal interval association rules, *in* 'Proceedings of the 2nd International Conference on Intelligent Data Engineering and Automated Learning', Morgan Kaufmann Publishers Inc., Shatin, N.T. Hong Kong, pp. 91–96.

Ramaswamy, S., Mahajan, S. & Silberschatz, A. (1998), On the discovery of interesting patterns in association rules, *in* 'Proceedings of the 24th VLDB Conference', Morgan Kaufmann, New York, USA.

Rathjens, D. (1997), Mineset users guide, Technical report, Silicon Graphics, Inc.

Readt, L. & Kramer, S. (2001), The levelwise version space algorithm and its application to molecular fragment finding, *in* 'Proceedings of the seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001)', Acapulco, Mexico, pp. 853–862.

Relue, R., Wu, X. & Huang, H. (2001), Efficient runtime generation of association rules, *in* 'Proceedings of the 10th ACM International Conference on Information and Knowledge Management', Atlanta, Georgia,USA, pp. 466–473.

Ribarsky, W., Katz, J., Jiang, F. & Holland, A. (1999), 'Discovery visualization using fast clustering', *IEEE Computer Graphics and Applications* **19**(5), 32–39.

Robertson, G. G., Mackinley, J. D. & Card, S. S. (1991), Cone trees: Animated 3d visualizations of hierarchical information, *in* 'Proceeding of the International Conference on Human Factors in Computing Systems (CHI'91)', ACM Press, New Orleans, U.S.A., pp. 189–194.

Roddick, J. F. & Spiliopoulou, M. (2002), 'A survey of temporal knowledge discovery paradigms and methods', *IEEE Transactions on Knowledge and Data Engineering* **14**(4), 750–767.

Rymon, R. (1992), Search through systematic set enumeration, *in* 'Proceedings of the Third International Conference on the Principles of Knowledge Representation and Reasoning', Morgan Kaufmann, Cambridge MA, pp. 539–550.

Sahar, S. (1999), Interestingness via what is not interesting, *in* S. Chaudhuri & D. Madigan, eds, 'Fifth International Conference on Knowledge Discovery and Data Mining', ACM Press, San Diego, CA, USA, pp. 332–336.

Sarker, M. & Brown, M. (1994), 'Graphical fisheye views', *Communications of the ACM* **37**(12), 73–84.

Savasere, A., Omiecinski, E. & Navathe, S. (1995), An efficient algorithm for mining association rules in large databases, *in* U. Dayal, P. M. D. Gray & S. Nishio, eds, 'Proceedings of the 21st International Conference on Very Large Databases', ACM Press, Zurich, Switzerland, pp. 432–444.

Saygin, Y., Verykios, V. S. & Elmagarmid, A. K. (2002), Privacy preserving association rule mining, *in* 'Proceedings of the 12th International Workshop on Research Issues in Data Engineering: Engineering E-Commerce / E-Business Systems (RIDE'02)', Morgan Kaufmann Publishers Inc., San Jose,CA,USA.

Shah, D., Lakshmanan, L. V. S., Ramamritham, K. & Sudarshan, S. (1999), Interestingness and pruning of mined patterns, *in* K. Shim & R. Srikant, eds, 'Proceedings of the 1999 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD)', ACM Press, Philadelphia, USA.

Sheelagh, M., Carpendale, T., Cowperthwaite, D. J. & Francis, F. D. (1997), 'Extending distortion viewing from 2D to 3D', *Computer Graphics* **17**(4), 42–51.

Shekhar, S. & Huang, Y. (2001), Discovering spatial co-location patterns: A summary of results, *in* 'Proceedings of the 7th Internationl Symposium on Spatial and Temporal Data Databases (SSTD01)', Vol. 2121 of *Lecture Notes in Computer Science*, Springer Verlag, Redondo Beach,CA,USA, pp. 236–256.

Shenoy, P., Haritsa, J. R., Sudarshan, S., Bhalotia, G., Bawa, M. & Shah, D. (2000), Turbo-charging vertical mining of large databases, *in* W. Chen, J. F. Naughton & P. A. Bernstein, eds, 'Proceedings of the 2000 ACM SIGKDD International Conference on Management of Data', Vol. 29, ACM Press, Dallas Texas, pp. 22–33.

Shneiderman, B. (1996), The eyes have it: A task by data type taxonomy for information visualization, *in* '1996 IEEE International Symposium on Visual Languages', IEEE Press, Boulder, Colorado, pp. 336–343.

Silberschatz, A. & Tuzhilin, A. (1996), 'What makes patterns interesting in knowledge discovery systems?', *IEEE Transactions on Knowledge and Data Engineering* **8**(6), 970–974.

Sindre, G., Gulla, B. & Jokstad, G. (1993), Onion graphs: Aesthetics and layout, *in* 'IEEE/CS Symposium on Visual Languages', IEEE CS Press, pp. 287–291.

Spence, S. & Apperley, M. (1982), Database navigation: An office environment for the professional, *in* 'Behaviour and Information Technology', Morgan Kaufmann, pp. 43–54.

Srikant, R. & Agrawal, R. (1996), Mining sequential patterns: Generalizations and performance improvements, *in* P. M. G. Apers, M. Bouzeghoub & G. Gardarin, eds, 'Proceedings of the 5th International Conference on Extending Database Technology, EDBT', SpringerVerlag, Avignon, France, pp. 3–17.

Srikant, R. & Agrawal, R. (1997), 'Mining generalized association rules', *Future Generation Computer Systems* **13**(2–3), 161–180.

Srikant, R., Vu, Q. & Agrawal, R. (1997), Mining association rules with item constraints, *in* D. Eckerman, H. Mannila, D. Pregibon & R. Uthursamy, eds, '3rd Int. Conf. on Knowledge Discovery and Data Mining', AAAI Press, Newport Beach, C.A., U.S.A., pp. 67–73.

Sucahyo, Y. G. & Gopalan, R. P. (2003), CT-ITL: Efficient frequent item set mining using a compressed prefix tree with pattern growth, *in* K. Dieter-Schewe & X. Zhou, eds, 'Proceedings of the 14th Australiasian Database Conference', Vol. 25, Australian Computer Society Inc., Adelaide, Australia, pp. 95–105.

Tan, P., Kumar, V. & Srivastava, J. (2002), Selecting the right interestingness measure for association patterns, *in* 'Proceedings of the Eight ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD02)', Edmonton, Canada, pp. 32–41.

Thomas, S., Bodagala, S., Alsabti, K. & Ranka, S. (1997), An efficient algorithm for the incremental updation of association rules, *in* 'Proceedings of the 3rd International conference on Knowledge Discovery and Data Mining (KDD 97)', ACM Press, New Port Beach, California, pp. 263–266.

Thomas, S. & Sarawagi, S. (1998), Mining generalized association rules and sequential patterns using sql queries, *in* 'Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD'98)', ACM Press, New York City, New York, pp. 344–348.

Toivonen, H. (1996), Sampling large databases for association rules, *in* T. Vijayaraman, Alejandro, P.Buchmann, C. Mohan & N. Sarda, eds, 'Proceedings of the 22nd International Conference on Very Large Data Bases', Morgan Kaufman, Mumbia (Bombay), India, pp. 134–145.

Toivonen, H., Klemettinen, M., Ronkainen, P., Hatonen, K. & Mannila, H. (1995), Pruning and grouping discovered association rules, *in* 'MLnet: Familarisa-

tion Workshop on Statistics, Machine Learning and Knowledge Discovery in Databases', Springer Verlag, Heraklion, Crete, pp. 47–52.

Vaidya, J. & Clifton, C. (2002), Privacy preserving association rule mining in vertically partitioned data, *in* 'Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02)', ACM Press, Edmonton, Alberta, Canada, pp. 639–644.

van Wijk, J. J. & van de Wetering, H. (1999), Cushion treemaps: Visualization of hierarchical information, *in* 'IEEE Symposium on Information Visualization INFOVIS'99', IEEE Press, San Fransisco, California, pp. 73–78.

Veloso, A., Meira Jr, W., de Carvalho, M, B., Possas, B., Parthasarathy, S. & Zaki, M. (2002), Mining frequent itemsets in evolving databases, *in* 'Proceedings of the 2nd SIAM International Conference on Data Mining', IEEE Press, Arlington, Virginia, USA.

Veloso, A., Possas, B., Meira Jr, W. & de Carvalho, M, B. (2001), Knowledge management in association rule mining, *in* 'Integrating Data Mining and Knowledge Management, held in conjunction with the 2001 IEE International Conference on Data Mining (ICDM)', San Jose, California, USA.

Verykios, V. S., Bertino, E., Fovino, I. N., Provenza, L. P., Saygin, Y. & Theodoridis, Y. (2004), 'State-of-the-art in privacy preserving data mining', *In SIGMOD Record* **33**(1), 50–57.

Wang, J., Han, J. & Pei, J. (2003), Closet+: searching for the best strategies for mining frequent closed itemsets, *in* 'Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining', ACM Press, Washington, DC, USA, pp. 236–245.

Wang, K., He, Y. & Han., J. (2000), Pushing support constraints into frequent itemset mining, *in* 'Proceedings of the 26th International Conference on Very Large Databases (VLDB2000)', Cairo, Egypt, pp. 43–52.

Wang, K., Jiang, Y. & Lakshmanan, L. V. S. (2003), Mining unexpected rules by pushing user dynamics, *in* 'Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining', ACM Press, Washington, D.C., USA, pp. 246–255.

Wang, K., Tang, L., Han, J. & Liu, J. (2002), Top down fp-growth for association rule mining, *in* 'Proceedings of the 6th Pacific Asia Conference on Knowledge

Discovery and Data Mining (PAKDD'02)', Springer-Verlag, Taipei, Taiwan, pp. 334–340.

Warner, S. (1965), 'Randomised response: A survey technique for eliminating evasive answer bias', *Journal of the American Statistical Association* **60**(309), 63–69.

Weber, E. H. (1834), 'Weber's law britannica concise encyclopedia'.

Wetherell, C. & Shannon, A. (1979), 'Tidy drawing of trees', *IEEE Transactions on Software Engineering* **5**(5), 514–520.

Wills, G. (1998), An interactive view for hierarchical clustering, *in* 'IEEE Symposium on Information Visualization (InfoVis '98)', IEEE Press, Raleigh, North Carolina, USA.

Wong, P. C., Whitney, P. & Thomas, J. (1999), Visualizing association rules for text mining, *in* 'Proceedings of IEEE Symposium on Information Visualization'99', IEEE Computer Society Press, Los Alamitos, California,USA, pp. 120–124.

Wrobel, S., Wettschereck, D., Verkamo, I., Siebes, A., Mannila, H., Kwakkel, F. & Klosgen, W. (1996), User interactivity in very large scale data mining, *in* W. Dilger, M. Schlosser, J. Zeidler & A. Ittner, eds, 'FGML-96 Annual Workshop of the GI Special Interest Group Machine Learning', TU Chemnitz-Zwickau, pp. 125–130.

Xia, Y., Wang, W., Yang, J., Yu, P. & Muntz., R. (2002), Efficient filtering of large dataset – a user-centric paradigm, *in* R. L. Grossman, J. Han, V. Kumar, H. Mannila & R. Motwani, eds, 'Proceedings of the 2nd SIAM International Conference on Data Mining (SDM)', Arlington, VA, USA, pp. 112–127.

Xiao, Y. & Dunham, M. H. (2001), Interactive clustering for transaction data, *in* Y. Kambayashi, W. Winiwarter & M. Arikawa, eds, 'Third Int. Conf. Data Warehousing and Knowledge Discovery', Springer Verlag, Munich, Germany, pp. 121–130.

Yee, K. P., Fisher, D., Dhamija, R. & Hearst, M. (2001), Animated exploration of graphs with radial layout, *in* 'IEEE Symposium on Information Visualisation 2001, InfoVis'01', IEEE Press, pp. 43–50.

Yen, S. J. & Chen, A. L. P. (1996), An efficient approach to knowledge discovery from large databases, *in* 'Proceedings of the IEEE/ACM International

Conference on Parallel and Distributed Information Systems', ACM Press, pp. 8–18.

Yen, S. J. & Chen, A. L. P. (1997), An efficient data mining technique for discovering interesting association rules, *in* 'Proceedings of the 8th International Conference and Workshop on Database and Expert Systems Applications (DEXA97)', pp. 664–669.

Zaiane, O. R. & El-Hajj, M. (2003), Cofi-tree mining: A new approach to pattern growth with reduced candidacy generation, *in* B. Goethals & M. J. Zaki, eds, 'Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03)', Melbourne, Florida, USA.

Zaki, M. (2000*a*), Generating non-redundant association rules, *in* 'Proceedings of the 6th International Conference on Knowledge Discovery and Data Mining, (SIGKDD'00)', AAAI PressACM, Boston, MA, USA, pp. 34–43.

Zaki, M. J. (1998), Efficient enumeration of frequent sequences, *in* 'Proceedings of the 7th International Conference on Information and Knowledge Management, CIKM', ACM Press, Bethesda, pp. 68–75.

Zaki, M. J. (1999), 'Parallel and distributed association mining: A survey', *IEEE Concurrency, Special Issue on Parallel Mechanisms for Data Mining.* **7**(4), 14–25.

Zaki, M. J. (2000*b*), 'Scalable algorithms for association mining', *IEEE Trans. Knowledge and Data Engineering* **12**(3), 372–390.

Zaki, M. J. & Gouda, K. (2001), Fast vertical mining using diffsets, Technical Report 01-1, Computer Science Department, Renasslaer Polytechnic institute.

Zaki, M. J. & Hsiao, C.-J. (2002), Charm: An efficient algorithm for closed itemset mining, *in* 'Proceedings of the Second SIAM International Conference on Data Mining', ACM Press, Arlington, Vancouver, pp. 457–473.

Zaki, M. & Ogihara, M. (1998), Theoretical foundations of association rules, *in* 'In Proceedings of 3 rd SIGMOD'98 Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'98)', Seattle, Washington, USA, pp. 85–93.

Zaki, M., Parthasarathy, S., Ogihara, M. & Li, W. (1997), New algorithms for fast discovery of association rules, *in* 'Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining (KDD-97)', AAAI Press, Newport Beach, California, pp. 283–286.

Zhou, Z. & Ezeife, C. (2001), A low-scan incremental rule maintenence method, *in* E. Stroulia & S. Matwin, eds, 'Proceedings of the 14th Canadian Conference on Artificial Intelligence (AI'2001)', Springer, Ottawa, Canada.

Zimmermann, H. J. (1996), *Fuzzy set Theory and its Applications (3rd ed.)*, Kluwer Academic publishers, Norwell, MA, USA.