Learning predictive models from graph data using pattern mining

Thashmee M. Karunaratne

Learning predictive models from graph data using pattern mining

Thashmee M. Karunaratne

Doctoral Dissertation
Department of Computer and Systems Sciences
Stockholm University
March 2014

Stockholm University
ISBN 978-91-7447-837-2
ISSN 1101-8526
DSV Report series No. 14-003

Typeset by the author using LaTeX
Printed in Sweden by US-AB

This thesis is dedicated to my ever loving Dad...
Who was my all-time believer

ABSTRACT

Learning from graphs has become a popular research area due to the ubiquity of graph data representing web pages, molecules, social networks, protein interaction networks etc. However, standard graph learning approaches are often challenged by the computational cost involved in the learning process, due to the richness of the representation. Attempts made to improve their efficiency are often associated with the risk of degrading the performance of the predictive models, creating tradeoffs between the efficiency and effectiveness of the learning. Such a situation is analogous to an optimization problem with two objectives, efficiency and effectiveness, where improving one objective without the other objective being worse off is a better solution, called a Pareto improvement. In this thesis, it is investigated how to improve the efficiency and effectiveness of learning from graph data using pattern mining methods. Two objectives are set where one concerns how to improve the efficiency of pattern mining without reducing the predictive performance of the learning models, and the other objective concerns how to improve predictive performance without increasing the complexity of pattern mining. The employed research method mainly follows a design science approach, including the development and evaluation of artifacts. The contributions of this thesis include a data representation language that can be characterized as a form in between sequences and itemsets, where the graph information is embedded within items. Several studies, each of which look for Pareto improvements in efficiency and effectiveness are conducted using sets of small graphs. Summarizing the findings, some of the proposed methods, namely maximal frequent itemset mining and constraint based itemset mining, result in a dramatically increased efficiency of learning, without decreasing the predictive performance of the resulting models. It is also shown that additional background knowledge can be used to enhance the performance of the predictive models, without increasing the complexity of the graphs.

SAMMANFATTNING

Inlärning från grafer har blivit ett populärt forskningsområde då grafdatabaser har blivit allt mer förekommande för att representera webbsidor, molekyler, sociala nätverk, proteininteraktionsnätverk m.m. Standardmethoderna för inlärning från grafer begränsas ofta av beräkningskostnaden i inlärningsprocessen på grund av representationsspråkets komplexitet. Försök som har gjorts för att förbättra deras effektivitet är ofta förknippade med en risk för försämring av de prediktiva modellernas prestanda, vilket kräver en avvägning mellan effektivitet och verkningsgrad vid inlärningen. En sådan situation är jämförbar med ett optimeringsproblem med två mål, effektivitet och verkningsgrad, där förbättring av ett mål utan att det andra målet blir sämre betraktas som en bättre lösning, en s.k. kallad Paretoförbättring. I denna avhandling studeras hur man kan förbättra effektiviteten och verkningsgraden vid inlärning från grafer med hjälp av mönsterextraktionsmetoder. Två mål för studien är att undersöka hur man kan förbättra mönsterextraktionens effektivitet utan att minska de resulterande modellernas prediktionsförmåga, och hur den prediktiva förmågan kan förbättras utan att öka mönsterextraktionens komplexitet. Avhandlingens forskningsansats följer huvudsakligen den designvetenskapliga metodiken och inkluderar konstruktion och utvärdering av artefakter. Avhandlingens bidrag inkluderar ett datarepresentationsspråk, som kan beskrivas som en form mellan sekvenser (*sequences*) och oordnade mängder (*itemsets*) där information om grafens struktur kodas som element i mängderna. Resultat presenteras från ett antal studier som undersöker ett antal föreslagna metoders Paretoförbättringar med avseende på effektivitet och verkningsgrad vid analys av datamängder innehållande små grafer. En av de viktigaste slutsatserna i arbetet är att två av de föreslagna metoderna, nämligen maximal frequent itemset mining och constraint based itemset mining, dramatiskt ökar effektiviteten vid inlärning utan att minska modellernas prediktiva förmåga. Det visas också att ytterligare bakgrundskunskap kan användas för att förbättra de prediktiva modellernas prestanda utan att öka grafernas komplexitet.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor Professor Henrik Boström for his knowledge sharing, which made this work a possible task for me. He has been a tough teacher and a sincere friend, who extended his hand whenever I required support, guidance or encouragement. Associate Professor Lars Asker's comments as the co-supervisor for my PhD thesis and the opponent for my Licentiate thesis, were very helpful, and my heartfelt gratitude is extended to him for giving such an excellent review. I also would like to thank Professor Ulf Norinder from Astra Zeneca R&D for his help and comments. He was kind enough to share the datasets from Medicinal Chemistry, and the chemical descriptors, as well as some results from his programs. Being the domain expert, his contributions as a co-author for some of the papers, made me successful in ending up with good publications in chemoinformatics. Professor Asoka Karunananda supported me in several ways during the early days of my research. The comments of Professor Paul Johannesson and Dr. Sumithra Velupillai during the pre-doctoral seminar were invaluable, which contributed immensely to the thesis in its present form. It is an honour to have Professor Nada Lavrač, Head of Department of Knowledge Technologies, Jožef Stefan Institute, Slovenia, as the opponent of my thesis during the public defense. I must also thank the three examiners of my PhD thesis, Professor Paul Johannesson from the Department of Computer and Systems Sciences (DSV), Professor Andriy Andreev from the Department of statistics and Associate Professor Anne Håkansson from School of Information and Communication Technology, Royal Institute of Technology, for their valuable comments, which will definitely be useful in my future research life. I would also like to thank Anna Hansson for being so kind to proofread my thesis, in fact in a short time!

I am in debt to the Swedish International Development Cooperation Agency (SIDA) and the High-Performance Data Mining for Drug Effect Detection (DADEL) project from the Swedish Foundation for Strate-

gic Research for the financial support I received during my studies at Stockholm University. My heartfelt gratitude should also be extended to Associate Professor Henrik Hansson for his kind advices and support. My colleagues at DSV, especially David, Constantino, Rueben, Orlando, Thushani, Sampath, Geoffery, Rasika, Japhet (and many more of course!) deserve a note of thanks for their kind friendship. Nam and Maria for very nice PhDs outings etc. Sven, Christer, Nicolas and Ola from the DSV DMC helped me, mostly in crucial times, in solving the problems with my computers. Fatima, Birgitta, Rodolfo and Sören are also thanked for their kind support given to me from time to time in my life at Stockholm University. Without the encouragement, support and understanding of my husband Sena and daughters Dehani and Sahani, none of this would have been possible. Also, my parents and two sisters always encouraged me to uplift my enthusiasm. A big thanks should go to them for believing in me.

Thashmee Karunaratne
Stockholm, March 2014

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Most of us today are lucky enough to receive appropriate medical treatments for many illnesses considered serious in the past, thanks to new medicinal discoveries. However, the discovery and development of a medical drug is a long term process costing a huge amount of money and time, resulting in more failures than successes (Wale, 2011). But the time and cost incurred by drug discovery has dramatically reduced when computer programs are introduced into certain steps of the drug discovery process. Computational methods known as *in silico* utilize computer programs to design, understand and predict the chemical and biological reactions of the compounds that are considered for creating drugs (drug candidates) (Ekins et al., 2007). Such knowledge helps researchers in the pharmaceutical industry to decide which compounds should be synthesized and tested in laboratories (*in vitro*), something which is both costly and time-consuming. In silico methods explore large databases of chemical compounds and automatically learn relationships between the compounds and their chemical and biological reactions. Computer programs that can be used to automatically learn from the available information is the core idea of machine learning.

1.1 BACKGROUND - MACHINE LEARNING

As Mitchell (2006) states, the best approach for defining a scientific field is by the central question it studies. Machine learning is the field that studies the question of *'how to build computer systems that automatically improve with experience, and what are the fundamental laws that govern all learning processes'* (Mitchell, 2006). The following is a typical machine learning example. A set of emails, which are already known as spam or non-spam (*training set*) are available. It is required to build a spam filter using the given set of spam and non-spam emails. A single email in the training set is an *example*, *object* or an *instance* of data. The emails are *labeled* as spam or non-spam, separating the training set into two *classes*. Further, the emails may be represented in terms of a set of words, which are usually called *attributes*. These attributes may take the values 1 or 0, where the presence of a word in an example (email) is represented by 1, and 0 if the word is not included in the email (the number of times each word appears in the email is an alternative representation). Each email in the set of training examples can thereby be transformed into a vector consisting of 1's and 0's that represent the presence and absence of the words (attributes) in the email. The form of representing examples as a vector containing attributes and their values is referred to as propositional, attribute-value or feature vector representation (Han and Kamber, 2000). An algorithm that can *learn*, automatically, which words and/or combinations are more likely to determine the class label of the email may be used in building the spam filter. An algorithm that can carry out such learning is called a *machine learning algorithm* and the spam filter is a *predictive model*, which could be used to predict the label (spam or non-spam) of an unseen (new) example. The more accurate the model is, the higher the probability that the spam filter predicts the correct label of new examples.

Learning predictive models using a training dataset with labeled examples, e.g., the spam and non-spam emails, is also referred to as supervised learning. The task for the predictive model is *classification* when the

labels are categorized into several classes and *regression* when the label is a numerical value. In contrast, learning from unlabeled data is called unsupervised learning (Mitchell, 2006).

The performance of the predictive model that classifies emails into spam and non-spam (given a training set), is influenced by 1) the chosen set of words to represent the emails (the attribute set), and, 2) how and in which way the learning model is built. Different choices of the words used for representing emails in the propositional (attribute-value) form may result in models which perform differently, even if the same learning algorithm is used for model building. Also, different machine learning algorithms use data in different ways during model building, resulting in differences in the predictive performance (Alpaydin, 2009). The quality of a machine learning algorithm is determined mainly by two properties, *efficiency* and *effectiveness*. Efficiency is determined by the computational cost incurred by the algorithm, that is, in common practice, the memory consumed and the time taken to complete the algorithm. With respect to a prediction task, the *effectiveness* of a machine learning algorithm may be reflected by the quality of the model built by use of the said algorithm, i.e., whether the algorithm can build a model and produce accurate predictions (Landwehr, 2009).

1.2 DATA REPRESENTATION

Most of the standard machine learning algorithms deal with data in attribute-value encodings (Cook and Holder, 2006). In this encoding, data (examples) are represented using the same (fixed number of) attributes that can be included in a single table, where each row in the table represents an example. The attributes correspond to the columns, where one of those columns may be dedicated to the (class) label. However, real world datasets often contain plenty of data that has a richer representational form that may not directly fit into this format (Dietterich et al., 2008). A chemical

compound from the domain of chemoinformatics[1] in Figure 1.1(a) is one example of such data (Bringmann, 2009).

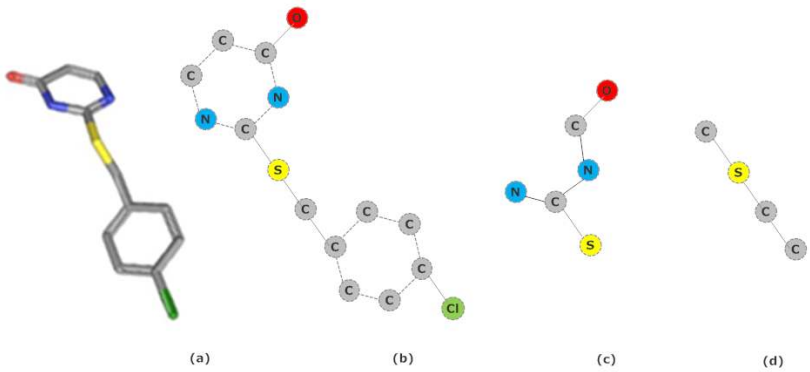

Figure 1.1: An example of a graph: (a) a molecule (b) the corresponding graph (c) a tree structure in (b) and (d) a sequence in (b).

If the information in this compound is encoded in the attribute-value (propositional) form, one may choose the atom labels as the attribute names (presented in any order), for example, (C, Cl, N, O, S), and the count of atoms with similar label as the value for each attribute. In Figure 1.1(a), there are 11 atoms with the label C, one with Cl, two with N, and one each with O and S, and therefore, the molecule could be represented by ($C = 11$, $Cl = 1$, $N = 2$, $O = 1$, $S = 1$) in attribute-value form. Such a representation ignores structural relationships, e.g. the bonds

[1] "Chemoinformatics is the mixing of those information resources to transform data into information and information into knowledge for the intended purpose of making better decisions faster in the area of drug lead identification and optimization" (Brown, 2005)

between the atoms. A richer representational form may be needed to encode these relations. Not only chemical data, but also data that could be found in repositories of social networks, computational biological data, web data and links, XML data, image data including handwritten documents, and many more, may require such a representational form.

Data containing structural relationships among the attributes are often referred to as *structured data* (Thomas, 2010). Further, along with this data there may be some additional information available, which could be useful for learning. This additional information is commonly referred to as *background knowledge* (Srinivasan et al., 1999). Usually this background knowledge comes from information that is specific to the given domain. For example, the chemoinformatics dataset called mutagenesis (Debnath et al., 1991) contains, in addition to molecular compounds, several other chemical properties of the molecules, such as *atomic weight* and *charge* and some structural information such as *carbon six ring* etc., which are specific to the domain.

Structured data can be represented in different forms, namely *logic programs*, *graphs*, *trees* and *sequences*.

1.2.1 LOGIC PROGRAMS

A logic program represents structured data in first order logic using so-called *predicates* (Srinivasan et al., 1999). For example, when representing the chemical compound in Figure 1.1(a) by a logic program, a predicate that represent the atoms, and a predicate to represent the relations between atoms (bonds) may be used. These two predicates can be defined in the form:

atm(molecule_id, atom_id, atom_name) and
bond(molecule_id, atom_id, atom_id, bond_type)

Here *molecule_id* and *atom_id* are the identifiers assigned to the molecule and the atoms within, respectively. The *atom_name* is the name of an atom (c for a carbon atom for example) and *bond_type*, which are single or double, are labeled 1 or 2. Accordingly, the compound in Figure 1.1(a) could be represented as:

atm(a,a_1,c)
atm(a,a_2,o)
atm(a,a_3,n)
atm(a,a_4,c)
atm(a,a_5,s)
...
bond(a,a_1,a_2,2)
bond(a,a_1,a_3,1)
bond(a,a_3,a_4,1)
...

Note that only a part of the encoding is presented here. Additional background information, if available, can also be encoded into logic programs. For example, the additional information that the molecule has 'a ring of six carbon atoms', which is a specific property in chemical terms, can be represented by a predicate *carbon_six_ring*. The compound in Figure 1.1(a), has a carbon six ring, which may be encoded as follows.

carbon_six_ring(a,a_11,a_12,a_13,a_14,a_15,a_16)

In this representation, the order of the presence of carbon atoms is arbitrary. Logic programs is recognized as a rich representational form since the structured data as well as the background knowledge can be easily encoded using first order logic (Srinivasan et al., 1999).

1.2.2 GRAPHS TREES AND SEQUENCES

The chemical compound in Figure 1.1(a) could be represented as a graph as shown in Figure 1.1(b) where the atoms are mapped to nodes and the bonds between atoms are mapped to edges in the graph. Here, nodes are labeled by the atom name. This representation of molecules are sometimes referred to as chemical graphs (Deshpande et al., 2003). In molecular graphs the edges have no directions such as 'directing from carbon atom to hydrogen atom'. Graphs containing such undirected edges are called undirected graphs. But, in general, an edge between two nodes may have a direction. For example, in a web link dataset, an edge would be a link from a particular web page to another web page.

Trees is a subclass of graphs. A graph that has its nodes arranged in a hierarchical form where no node can be revisited when traversing the structure (and thereby containing no cyclic paths in the structure) is referred to as a tree. A subgraph of Figure 1.1(b), presented in Figure 1.1(c) is an example for a tree structure. *Sequences* are graphs where the nodes are connected with each other in a sequence, i.e, no node has more than two edges and exactly two nodes have one edge each. Figure 1.1(d) is a sequence embedded within the graph in Figure 1.1(b). However, when general graphs are represented as trees, the cyclic paths of the graph is lost while all the structures other than the the nodes and edges in sequence are lost, when the representation is in a form of sequence. Therefore trees and sequences are less powerful and less complex representational forms than graphs.

Graph data broadly fall into two categories depending on their size and characteristics, namely set of (small) graphs and single (large) graph. Chemical graphs shown in Figure 1.1(b) for example, consist of graphs with small number of nodes (and edges) which do not usually exceed a couple of hundreds. There may be small to large numbers of such graphs in a dataset. Using these chemical graphs, one could model the relationship between the chemical structure of compounds and their chemical effects,

such as solubility, permeability, protein binding, mutagenicity, carcinogenicity, metabolic stability and so on (Srinivasan et al., 1999).

The graph of a social network given in Figure 1.2, is an example for the other category, single graph, which contain large numbers of nodes (and edges). In such a graph, nodes represent the individuals attached to the network (actors) and edges represent the relationship of a particular individual with the other individuals in the network. Typically, a social network is a single graph, possibly having hundreds of thousands of nodes. By analyzing a social network one could identify the relations between individuals, such as friends, neighbours, etc. (Aggarwal and Wang, 2010). A fragment that has a large number of connections may be representing an actor who is popular within the network.

In learning from graphs in these two categories, methods may need to address different challenges. In this thesis, the main focus is on learning predictive models when the dataset is a set of (small) graphs.

1.3 LEARNING FROM GRAPH DATA

Consider the chemoinformatics dataset in Figure 1.3, which contains chemical graphs. The learning task is prediction, where a predictive model is built using compounds (training data) that are labeled as mutagenetic or not, e.g. the compounds *a–e* in Figure 1.3, so that the model can be used to label previously unseen (non-labeled) molecules, e.g., the compound f in Figure 1.3.

When the data are represented as logic programs, inductive logic programming algorithms could be used to learn predictive models in the form of so called relational descriptions (Muggleton, 1991). In the ILP approach, for a given set of examples (E) and background knowledge (B), a hypothesis (H) is produced, where B and H together classify E. Here B, H and E are logic

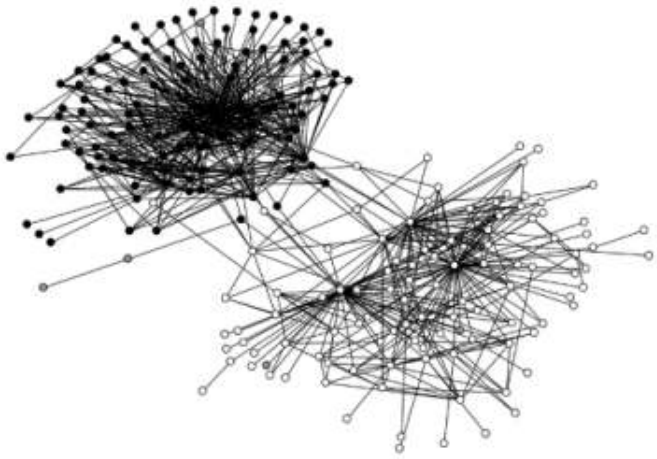

Figure 1.2: A single large graph – citation patterns in the small world literature (Freeman, 2004), where the black, white and gray nodes refer to physicists, people engaged in social networks, and others respectively, and the edges represent the citations.

programs. E can be separated into positive examples (E+) and negative examples (E-) (Muggleton and Raedt, 1994). Logic is used for defining hypotheses. A hypothesis is a set of rules that cover as many examples as possible from the positive class, and exclude the examples belonging to the negative class. There are several techniques within ILP that can be used for learning classifiers, such as Inverse Resolution, (Muggleton and Buntine, 1988), Relative Least General Generalization (Muggleton and Feng, 1992), Inverse Implication, Inverse Entailment (Muggleton, 1995) etc. There exist several studies that show effective applications of ILP methods, e.g., (Nédellec et al., 1996), (Lavrač et al., 2002), (Muggleton and Raedt, 1994) and (Srinivasan, 2004). However, various constraints used by the ILP based methods during the search and construction of hypotheses may limit

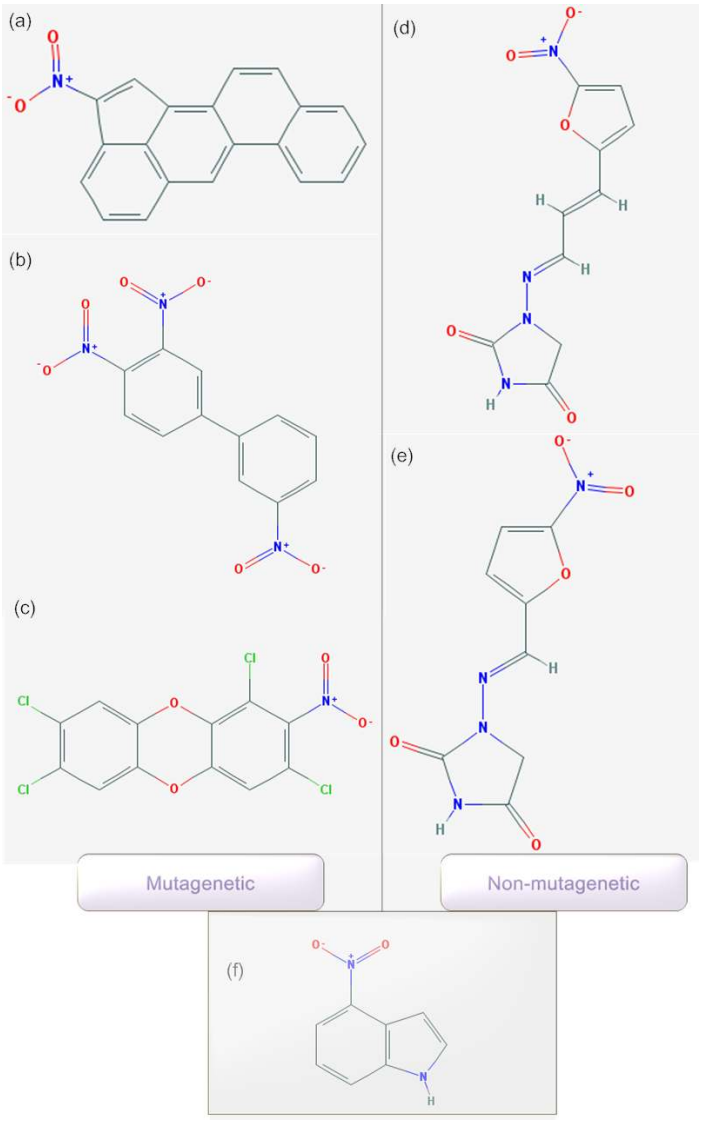

Figure 1.3: Chemical graph dataset (a) 2-Nitrobenz(j)aceanthrylene (b) 3,4,3'-Trinitrobiphenyl (c) 2-nitro-1,3,7,8-tetrachlorodibenzo-1,4-dioxin (d) 1-((3-(5-Nitro-2-furyl)allylidene)amino)hydantoin (e) nitrofurantoin (f) 4-nitroindole (Nicklaus, 1996).

the applicability of these methods (Landwehr, 2009).

When the data are represented as graphs, predictive models can be built by either using graph learning algorithms that accept input data in the form of graphs (Tsuda and Saigo, 2010), or, by transforming the graphs into a form that can be used in standard machine learning algorithms (i.e., attribute-value representation) (Deshpande et al., 2005; Krogel et al., 2003) as illustrated in Figure 1.4.

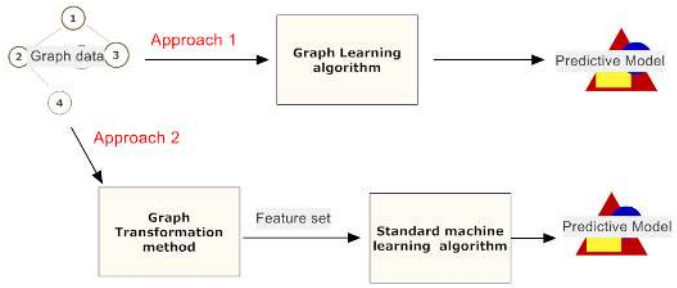

Figure 1.4: Two approaches to learning from graphs.

A majority of the machine learning algorithms that deal with graphs as their input use the similarity between the graphs in the dataset as a measure for determining which of the graphs belong to the same class. In these methods, a function called the *kernel* is used to compute the similarity. A *similarity matrix* (a square matrix of the size of the dataset) stores the values, where each entry in the matrix represents a value that indicates the similarity between the two graphs in the respective row and column. In these methods, both kernel computation and model building are integrated in the graph learning algorithm (Borgwardt and Kriegel, 2005; Gärtner, 2003; Horváth et al., 2004; Vishwanathan et al., 2010). The most challenging aspect of using graph kernel based methods is the selection of a suitable kernel, as the predictive performance of the model is highly

dependent on the kernel function (Gärtner, 2003). Further, since the kernel computation is embedded within the learning algorithm, standard machine learning algorithms may not be reused in this approach (Platt, 1999).

The other approach to building predictive models is by transforming the graphs into a form that a standard machine learning algorithm accepts as the input, which is a two step approach, i.e., 1) selection of an attribute set to represent graphs, and 2) building predictive models using any standard machine learning algorithm as illustrated in Figure 1.4. This approach is also called as *Propositionalization* (Helma et al., 2003; Krogel et al., 2003; Lavrač et al., 2002). In general, the attribute set consists of patterns within the graph data, which are correlated with the class label. Pattern mining methods can be used to discover these patterns (Han et al., 2007). Further descriptions of the pattern mining methods are presented in the next section.

The two approaches to graph learning concern two different problems in learning. When building global models, the main concern is how to learn efficiently and effectively from graphs. The principal concern of the methods of learning from attribute-value representations of graphs is the discovery of a pattern set efficiently that maximizes the predictive performance (Saitta and Sebag, 2010). In this thesis we focus on learning from attribute-value encodings of graphs, motivated by the possibility of re-using standard machine learning algorithms.

1.4 PATTERN MINING METHODS

Pattern mining methods concern several types of patterns, namely, *graphs*, *trees*, *sequences* and *itemsets*. Among other patterns graphs serve as the richest representation language (Aggarwal and Wang, 2010; Bringmann, 2009; Deshpande et al., 2003; Gärtner, 2003; Gonzalez et al., 2003; Han and Kamber, 2000; Landwehr, 2009). These methods employ *fast counting techniques* on data to discover *interesting* features (Borgelt, 2002; Bringmann and Zimmermann, 2005; Cook and Holder, 1994; De Raedt

and Kramer, 2001; Hasan and Zaki, 2009; Helma et al., 2003). Subsequent sections include discussions of these methods.

1.4.1 FREQUENT SUBGRAPH MINING

Frequent subgraph mining is the name given to the methods that discover frequently appearing subgraphs within a graph dataset. For example, the three graphs in Figure 1.5 below are some subgraphs present in the graphs *a*-*d* in Figure 1.3.

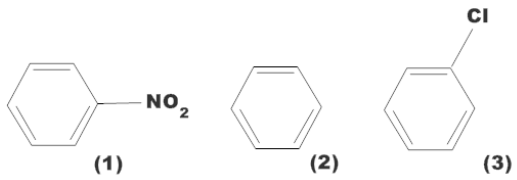

Figure 1.5: Three subgraphs present in the chemoinformatics dataset.

Suppose we are interested in discovering subgraphs with frequency (support) ≥ 2, i.e., those subgraphs present in two or more graphs in the graph database. In this particular dataset, subgraphs (1) and (2) are frequent. If the presence of a subgraph is denoted by 1, and, 0 if not, the graphs *a*, *b*, *c* and *d* in Figure 1.3 could be encoded in terms of these two frequent subgraphs as in Table 1.1[1]. In this table, the presence of subgraphs (1) and (2) in graph *b*, for example, is represented by 1 and 1 in the two columns

[1]The two frequent graphs are present in the mutagenetic examples, thereby we can say that if any of these subgraphs are presented in an unlabeled example, e.g., the Figure 1.3(f), that example can be labeled as mutagenetic.

of the second row [1].

Table 1.1: The propositional representation of the chemoinformatics dataset using frequent subgraphs.

compound/ subgraph	**1**	**2**
a	0	1
b	1	1
c	1	1
d	0	0
e	0	0

The typical approach to discovering frequent subgraphs is an iterative procedure where in each iteration a set of possible frequent subgraphs (candidates) is selected, followed by counting the frequency of each candidate subgraph and extracting the subgraphs that exceed a predefined level of support (Bringmann, 2009). Although the concept of frequent subgraphs is simple, discovering the candidate set and computing the support are not straightforward. The combinatorial explosion of the pattern search space when the size of the pattern set increases results in 1) a longer processing time (reduced *efficiency*) of the subgraph discovery algorithm, and 2) generating a massive set of frequent subgraphs, cf. the 'curse of dimensionality' (Wang and Yang, 2005), limiting the applicability of the discovered patterns to tasks such as classification and regression (Aggarwal and Wang, 2010). Solving the second problem is challenging within the approach to frequent graph mining, yet there is an abundance of methods for solving the first problem, i.e., speeding up the processing time of subgraph discovery. Algorithms that are found in the literature for frequent subgraph mining broadly fall into two major categories: apriori based and

[1] The subgraph (3) is not included in the table since it is not frequent.

pattern-tree based (Han et al., 2007).

Apriori based approaches employ the *apriori* property that any subgraph of a frequent graph is frequent (Inokuchi et al., 2000). By applying the apriori condition, a large number of subgraphs could possibly be removed from the candidate set. The mining algorithm is iterative. In each iteration, two *small sized* graphs are joined to incrementally generate larger (candidate) graphs, followed by the frequency computation of each new candidate. The apriori-based graph miner AGM (Inokuchi et al., 2000), and the frequent subgraph mining algorithm FSG (Kuramochi and Karypis, 2001) were among the first to use apriori based breadth-first search for discovering candidate subgraphs. AGM's candidate generation is vertex (node) based (the size metric is the number of nodes), which means that during each iteration, the size of the subgraph is increased by one node. In contrast to the vertex (node) extension of the AGM algorithm, FSG's candidate graph discovery uses edge extension on graphs. The size of the subgraphs is increased by merging two $k+1$ sized graphs that contain k identical edges. However, the complexity of the candidate generation step of apriori based methods increases with increasing graph size (number of nodes/edges in the graph) (Hasan and Zaki, 2009). Furthermore, all possible k sized graphs are checked when determining whether a $k+1$ sized graph is frequent, which requires a substantial amount of memory(Han and Kamber, 2000).

In the pattern growth approach, subgraphs are extended by iteratively adding new edges to the existing subgraphs, allowing the extension of patterns directly from a single graph. The advantage of this approach over the apriori is that there is no need to complete the computation of all the k-sized graphs prior to computing $k+1$-sized graphs. However an additional cost might be involved when graphs could be extended in many ways resulting in the same graph being discovered many times (Han and Kamber, 2000). For example, an n-edge graph could be discovered from n different $n-1$ edge graphs. To avoid generating these *duplicate* graphs, one could extend the graphs in a conservative manner (Cook and Holder,

2006). The pattern growth based algorithm gSpan (Yan and Han, 2002) and several other methods presented in (Borgelt, 2002; Huan et al., 2003) and (Nijssen and Kok, 2004) are inspired by this approach. gSpan, the most cited method in this category, improves the efficiency of the mining by constructing depth first search (DFS) trees from the graphs, which are then converted into a lexicographically ordered sequence, called DFS code. Using mathematical (logical) operations, one sequence referred to as the *minimum DFS code* is selected. Thereby the complete frequent pattern set, i.e., all possible frequent patterns, is discovered (Yan and Han, 2002). The method GASTON (Nijssen and Kok, 2004) extends this approach by separating sequences, trees and cyclic graphs during the discovery process. Since matching subgraphs generally require the computationally costly operation of subgraph isomorphism test, (Cook and Holder, 2006), a gain in the efficiency could be obtained by excluding sequences and trees when testing subgraph isomorphism.

As shown in many publications such as (Han et al., 2007; Kuramochi and Karypis, 2001; Nijssen and Kok, 2004; Yan and Han, 2002), the size of the frequent set grows exponentially when reducing the minimum support. This is a limitation of frequent patterns when the task is classification or clustering.

1.4.2 SUBSETS OF FREQUENT SUBGRAPHS AND OTHER ALTERNATIVE METHODS

As a way of obtaining a pattern set that could be used for representing graphs for prediction tasks, *closed* (Yan and Han, 2003) and *maximal* (Burdick et al., 2001) frequent patterns have been introduced (Han et al., 2007). Several studies including (Yan and Han, 2003), (Huan et al., 2003) and (Cook and Holder, 2006) have shown that by the use of closed and maximal sets the size of the pattern set may be reduced by about 90%. CloseGraph (Yan and Han, 2003) introduces an extension to gSpan (Yan and Han, 2002), which has the same efficiency as gSpan, but determines the closed set from the gSpan frequent subgraphs. In (Huan et al., 2003) it is shown

that the maximal frequent subgraphs can be efficiently discovered using the frequent set. Top k frequent patterns (Ke et al., 2009) is another subset of frequent patterns. As an alternative to prioritizing the frequency for choosing the subset of frequent patterns, a weighting scheme is assigned to reduce the number of subgraphs discovered by gSpan in (Jiang et al., 2010).

Alternative measures to frequency in finding the pattern set, such as, pattern compression (Cook and Holder, 2006; Ketkar et al., 2005) and summarization (Hasan and Zaki, 2009; Yan et al., 2005), using *interestingness* measures (Hintsanen and Toivonen, 2008) and correlations (Ke et al., 2009), methods that discover significant patterns (not necessarily frequent but important in terms of correlation with the class variable) (Ranu and Singh, 2009), and methods that filter subsets from the frequent patterns (Thoma et al., 2009) are also among the related work. A slightly different approach to the discovery of interesting (approximate) subgraphs using the minimum description length, called SUBDUE, is presented in (Cook and Holder, 2006). Measures such as *contrast*, which maximizes the frequency of one class of graphs against the other classes (Borgelt, 2002), and evolutionary computing methods that can discover interesting patterns (Jin et al., 2010) are also being used in pattern mining.

1.4.3 THE EFFICIENCY VS. EFFECTIVENESS TRADEOFF IN GRAPH MINING METHODS

As stated above, limiting the search space and the candidate set improves the efficiency of the mining. Pruning the search tree at a certain level, may result in a substantial reduction of the number of candidates to be evaluated, and hence a gain in efficiency by reducing the number of subgraph matches, but large and possibly potential subgraphs may then be left out (Washio and Motoda, 2003). Controlling the *beam size* of the methods that use the minimum description length (MDL) principle also results in a potential loss by missing large subgraphs (Hasan and Zaki, 2009). In SUBDUE (Cook and Holder, 1994), significant patterns may

be missed due to limiting the candidate subgraphs by the MDL principle (Inokuchi et al., 2000). Therefore pruning the search space may result in an incomplete search, and, consequently missing large subgraphs; limiting the candidate set may result in failing to discover important subgraphs. Missing potentially useful subgraphs, which could have been captured if a complete enumeration had been carried out, may lead to a reduced effectiveness of the learning models (Cook and Holder, 2006).

Representing graphs in other less complex pattern languages before matching the subgraphs improves the efficiency of the mining (Bringmann, 2009). For example, gSpan's (Yan and Han, 2002) transformation of graphs into minimum DFS code, transforms graph matching into a matching of two sequences. Several methods transform graphs into trees to avoid the expensive steps of graph mining (Bringmann and Zimmermann, 2005). Some pattern mining algorithms look for certain forms of patterns such as trees and sequences. MolFea (De Raedt and Kramer, 2001), for example, introduces a complete search over some sequences, by a non-greedy search called the level-vise version-space algorithm, and discovers only the linear molecular fragments. However, since the purpose of using graphs as the representational form is to include the complex internal structure of the data that cannot be captured by other forms of representation, there could be an information loss due to this approach. Restricting the pattern set to a specific form may also result in losing important patterns that are not in the considered form. Both transforming graphs and restricting the search to less complex representational form may therefore affect the effectiveness of the predictive models.

As illustrated in Section 1.2, there may be background knowledge available along with the graph data. Typical graph based learners use only the topological structures (atom-bond relations) of the molecules in the representation language, ignoring other available background information. In Gonzalez et al. (2003), a method to include background knowledge is presented as follows. A molecular fragment with two carbon atoms

sharing an aromatic bond, for example, which may be represented as a chemical graph as in Figure 1.6, where *c* is the label of the two nodes and the aromatic bond between the two atoms is denoted by the edge label, 7, is expanded as illustrated in Figure 1.7. In this representation, background information, such as, the charge (-13), the atomic value of the carbon atom (22), and, two sub groups, Halide and Six-ring, each of which contains the atoms as a part, are presented as additional nodes.

Figure 1.6: General representation of a molecular fragment.

Using additional nodes and edges to represent background knowledge in the above fashion results in large graphs. The Figure 1.7 corresponds to two nodes (atoms) and additional background knowledge of three properties only, but a typical small molecule contains at least 5–6 atoms and several structural relations. When graphs become large, the number of candidate graphs increases, requiring additional memory to accommodate the search space and/or the candidate graphs (Hasan and Zaki, 2009). Graph mining methods may be subject to the risk of exhausting the memory at such a point. Large candidate sets may also require longer enumeration times. The runtime of a typical frequent graph mining algorithm increases exponentially with decreasing minimum support (Ranu and Singh, 2009). Therefore, when the graph size is increased the efficiency of the mining method is substantially reduced. This may be a reason for why there has been no systematic investigation of efficiently and effectively using

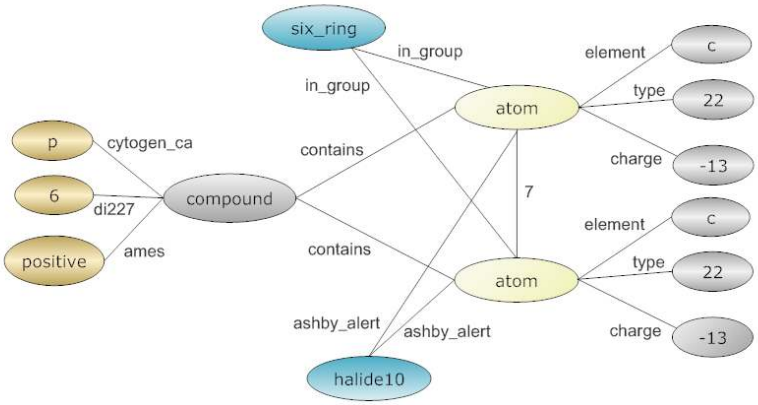

Figure 1.7: Including available background information.

available background knowledge to enhance the classification accuracy of existing graph based learning methods. Even in (Gonzalez et al., 2003), there was no examination of an enhancement in the model accuracy by representing a graph of Figure 1.6 in the form of Figure 1.7. However, methods such as MoFa (Borgelt, 2002), for example, use some of this background knowledge (such as six-ring) to compress the graphs and evaluate the discovered patterns.

1.5 PROBLEM

Graphs are expressively rich compared to other patterns such as trees, sequences, itemsets etc. More accurate predictive models could be built using graph patterns (Bringmann, 2009). Nevertheless, graph pattern mining methods often encounter complexity related issues during the mining process due to the richness in the representation (Thomas, 2010). Reducing the complexity of the graph mining method by using constraints

at various levels, and restricting the search to specific patterns and/ or transforming graphs into structures that are less complex to handle, may however result in a spectrum of efficiency vs. effectiveness tradeoffs as described in the previous section. The propositional setting illustrated in Section 1.2.2 lies at one extreme of this spectrum, while the methods that fully utilize the expressiveness of graphs lie at the other end. In between are the trade-offs of different pattern mining methods found in the literature.

As stated in the introductory section, the objective of machine learning is to learn from experience efficiently and effectively (Mitchell, 2006). In learning from graph data using pattern mining methods, a tradeoff exists between efficiency and effectiveness, where attempts taken to maximize one of them may lead to a reduction in the other, as discussed above. This can be viewed as an optimization problem with two objectives, i.e., minimize computational cost and maximize predictive performance. Therefore, any method that is capable of making one of the two objectives better off without making the other worse off compared to any other method could be ranked better (Zitzler et al., 2003). Such a state in an optimization problem is a *Pareto improvement*. Any such improvement, which cannot further get better off without making the other objective worse off, is called *Pareto efficient* or *Pareto optimal*. There might exist several methods that produce such improvements (*Pareto front*, i.e., the set of choices that are Pareto efficient). Konak et al. (2006) contains a detailed discussion of Pareto optimality.

When learning efficient and effective predictive models in the presence of a tradeoff between efficiency and effectiveness, a method that is Pareto efficient should be chosen. A comparatively efficient pattern mining method that results in competitive predictive models, and/ or a method that perform comparatively better without making efficiency worse off, should be on the Pareto frontier. However, which of the pattern mining methods results in Pareto improvements in efficiency vs. effectiveness is an open question. If the graphs can be represented in an alternative form which is computation-

ally less complex to handle than graphs, a gain in the efficiency of mining could thereby be expected. If such an approach results in models competitive in predictive performance compared to the methods using graphs, a Pareto improvement can be achieved. A graph mining method that explores a part of the graph search space (due to the restrictions imposed for improving the efficiency) might not perform any better than a method that uses any other less complex representational form (hence computationally efficient) than graphs (Thomas, 2010). Further, the effectiveness of predictive models may be improved by the use of background knowledge, but, the efficiency of the pattern mining methods decreases when the background knowledge is included in the graphs, due to the expansion of the size of the graphs, as pointed out in the previous section. However, it may be possible to incorporate background knowledge into learning from graphs, without increasing the complexity of the pattern mining process, i.e., without expanding the size of the graphs. Some light has to shed along these lines of research.

1.5.1 RESEARCH QUESTION AND THESIS OBJECTIVES

The research question of this thesis concerns *how to improve efficiency and effectiveness of methods for learning predictive models from graph data using pattern mining*.

In addressing this research question, we set the following objectives:

1. Investigate whether different pattern language settings could be used for improving the efficiency without affecting the performance of the predictive models.

2. Investigate whether background knowledge can be incorporated into learning from graphs without increasing the complexity of pattern mining.

1.5.2 THESIS CONTRIBUTIONS

The main contributions of this thesis have been published in nine included papers. Investigations of the predictive performance when efficiency is improved by using a less complex pattern language, as well as improving the predictive performance of those models by the use of background knowledge are included in these papers as summarized below.

Paper I

Thashmee Karunaratne and Henrik Boström. DIFFER: A Propositional Approach for Learning from Structured Data. In A. Pashayev, ed. *Transactions on Science Engineering and Technology*, 15: 49 – 51, Barcelona, Spain, 2006. World Enformatika Society.

In this paper, we introduce a less complex representational form than graphs, and study the performance of the predictive models built using data in this representation. A graph representation language, which is called *fingerprint* in this paper, is introduced. A method that discovers a feature set using the similarity between the fingerprint transformations of graphs, referred to as the maximal common substructure discovery method (*mcs*) is also presented. The predictive performance of the models using the feature sets from this method is compared with an inductive logic programming based method. Results show that the models built using the feature sets from maximal common substructures perform equally well as the comparison method. Further in this experiment, the feature set of one method is used as the background knowledge for the other method, since background knowledge is the additional knowledge available that is not encoded in the input and the feature sets of the two methods are different from each other. More accurate predictive models are obtained by combining features derived by both methods. Therefore, the conclusions were drawn that transforming graphs into a form that is less complex to handle, performs equally well as the standard method, and thereby makes a Pareto improvement in efficiency and effectiveness. Also, by the use of additional background

knowledge without increasing the complexity of the pattern mining, an enhancement of the model accuracy is achieved. Comparing the proposed method with only one standard method is a limitation of this experiment.

Paper II

Thashmee Karunaratne and Henrik Boström. Learning from structured data by finger printing. In *Proceedings of the 9th Scandinavian Conference on Artificial Intelligence*, pages 120 –126, Helsinki, Finland, 2006. Finnish Artificial Intelligence Society.

This paper extends Paper I by larger experimental comparison. The methods mcs, SUBDUE-CL (Gonzalez et al., 2003), $Tree^2\chi^2$(Bringmann and Zimmermann, 2005) and RSD (Lavrač et al., 2002), were compared. The results showed that there is no difference between the predictive performance of the methods, concluding that predictive models that are competitive with graph mining methods and logic based methods can be built using a representational form that is less complex than graphs (hence more efficient than graph mining), i.e., the proposed methods show Pareto improvement in efficiency and effectiveness compared to some standard methods.

Paper III

Thashmee Karunaratne and Henrik Boström. Learning to classify structured data by graph propositionalization. In B. Kovalerchuk, ed. *Proceedings of the Second IASTED International Conference on Computational Intelligence*, pages 393–398, San Francisco,USA, 2006. ACTA press.

In this paper, a method of adding background knowledge into the graphs is presented. Instead of adding background knowledge as new nodes and edges, which enlarges the graph (hence increasing the complexity of the mining), the representation language (fingerprint) is extended in order to include some background knowledge in the node/edge labels of the graphs.

By such addition the number of nodes in the graph does not increase. The proposed method is compared with the two methods, RSD (Lavrač et al., 2002) and MolFea (Helma et al., 2003). Adding background knowledge into the extended node/edge labels resulted in a significant increase in classification accuracy of the models built using the proposed pattern mining method *mcs*. In addition to concluding that combining feature sets from the other methods improve the predictive performance in Papers I and II, further enhancement of the predictive performance is shown when the feature sets with additional background knowledge are combined with the feature sets discovered by the other comparison methods, showing that background knowledge can be included in graphs without increasing the complexity of the pattern mining.

Paper IV

Thashmee Karunaratne and Henrik Boström. The effect of background knowledge in graph-based learning in the chemoinformatics domain. In Oscar Castillo, Li Xu, and Sio-Iong Ao, editors, *Trends in Intelligent Systems and Computer Engineering*, volume 6, pages 141–153. Springer US, 2008.

Use of different levels of background knowledge without increasing the cost of pattern mining is presented in this paper. The relation between the amount of the background knowledge and the predictive performance is studied. The method of representing background knowledge introduced in Paper III is extended to five different levels, where in each level the amount of the background knowledge included is increased. This paper concludes that the amount of the relevant background knowledge encoded into graphs is directly related to the performance of the predictive models, and the more the additional background knowledge included, the better the predictive performance of the models.

Paper V

Thashmee Karunaratne and Henrik Boström. Graph propositionalization for random forests. In *8th International Conference on Machine Learning and Applications*. pages 196 – 201, Miami, Florida, 2009. IEEE Computer Society.

This paper presents a performance analysis of different methods of encoding graphs as itemsets. Use of frequent itemset mining methods for graph data is also presented. Both graph structured data and data in attribute-value form (unstructured data) are used in the experiments. The results showed that there is a significant difference between the predictive performances of the models built using different graph encoding methods, indicating that it is not the expressiveness of the representational form of the graphs, but the way the graphs are encoded is what is responsible for the differences in the predictive performance. For unstructured data there was no significant difference in the predictive performance of these different encoding methods. This justifies that, for graph data, failing to take into account the edges in the representation affects the performance of the models built. This experiment also showed that graphs encoded as conventional itemsets, i.e., attribute-value encodings where the attribute names are the node labels, results in a significantly low predictive performance of the learning models built using the Random forest algorithm. Note that this is one of the extremes of the spectrum of efficiency and effectiveness caused by conventional propositional setting discussed in the Section 1.5. The conclusions drawn in this experiment include that the use of itemset mining algorithms on the fingerprint representation of graphs is Pareto efficient compared to the other methods presented in this study.

Paper VI

Thashmee Karunaratne, Henrik Boström, and Ulf Norinder. Pre-processing structured data for standard machine learning algorithms by supervised graph propositionalization – A case study with medicinal chemistry

datasets. In *9th International Conference on Machine Learning and Applications*, pages 828 – 833, Washington DC, USA. 2010. IEEE Computer Society.

Inspired by the conclusions of Paper V, the frequent itemset mining approach is applied to chemoinformatics datasets. This method is extended to a supervised form of discovering features (called supervised maximal frequent itemset mining *Smfi*), which produces feature sets that are frequent for a given class in the dataset. Any reduction in the predictive performance of *Smfi* is investigated by comparison with the graph mining methods SUBDUE (Cook and Holder, 1994) and MoFa (Borgelt, 2002), using three learning algorithms. Further, any improvement of predictive performance by using background knowledge, not included in the node definitions but as additional features to the learning models, is also investigated. The results showed that predictive models more powerful than from the pattern sets alone can be obtained by this method of adding background knowledge as additional features. The conclusion drawn in this paper is that supervised maximal frequent itemsets, which is less complex than graph mining, lead to producing models that are competitive in predictive performance, compared to graph mining methods. Investigating other itemset mining methods to see which of them produce Pareto improvements in efficiency-effectiveness are further works of this study.

Paper VII

Thashmee Karunaratne. Is frequent pattern mining useful in building predictive models? In *ECML/PKDD Workshop of Collective Learning and Inference on Structured Data*. pages 61–72, Athens, Greece, 2011.

In this paper, the usefulness of frequent pattern mining in building predictive models is further studied by investigating the applicability of frequency based methods for discovering patterns for classification (regression) tasks. Eight different methods, namely, SUBDUE (Cook and Holder, 1994), MoFa (Borgelt, 2002), graphSig (Ranu and Singh, 2009), Super-

vised Maximal frequent itemset mining (which is a contribution of the previous paper), Maximal frequent itemset mining (mfi), constraint programming based itemset mining for graphs (CP), and two domain specific feature sets SELMA (Olsson and Sherbukhin, 1999) and ECFI (Rogers and Hahn, 2010), are compared. The method, itemset mining using constraint programming (CP), is applied to graph data in this study, as a new contribution. The results showed that one of the methods that included the domain specific features alone resulted in models with significantly better performance than the others. Using additional methods to the previous study, namely mfi, CP and graphSig, the same conclusion as in Paper VI, i.e. domain specific background knowledge used as separate features in conjunction with patter sets result in Pareto improvements, is drawn. Bu the lack of a statistical comparison of the efficiency of the pattern mining methods is a limitation of this study.

Paper VIII

Thashmee Karunaratne, Henrik Boström, and Ulf Norinder. Comparative analysis of the use of chemoinformatics-based and substructure-based descriptors for quantitative structure–activity relationship (QSAR) modeling. In *Intelligent Data Analysis*, 17(2): 327 – 341, 2013. IOS press.

Investigation of the efficiency-effectiveness tradeoff spectrum resulting from different methods related to a domain specific problem in chemoinformatics, namely, Quantitative structure–activity relationship (QSAR) analysis is the main focus of this study. Two domain specific approaches, ECFI and SELMA, and five approaches for graph mining, CP, graphSig, MFI, MoFa, and SUBDUE are compared. The contribution of this paper includes modifying the graph representation language, to accompany the nodes with identical labels. The empirical investigation concluded that one of the chemoinformatics-based approaches, ECFI, builds significantly more accurate models than all other methods. Also, in this paper, the use of feature sets from another method to enhance the predictive performance of the models is investigated. In doing so, all possible combinations of the feature

sets from the methods were combined two at a time. The results showed a significant improvement of the predictive performance when the features from the method ECFI are used with any other feature set, while the ECFI also led to improved performance in many cases when the features generated by the other methods were added.

Paper IX

Thashmee Karunaratne and Henrik Boström. Can frequent itemset mining be efficiently and effectively used for learning from graph data? In *11th International Conference on Machine Learning and Applications*, pages 409 – 414. 2012. IEEE Computer Society.

This paper presents a study of different representational forms of graphs and their efficiency and effectiveness in building predictive models. Based on the results of the previous studies, two of the proposed methods, *mfi* (maximal frequent itemset mining) and CP (constraint programming approach based itemset mining) are selected to compare with graph mining methods. A comprehensive empirical evaluation is carried out for classification as well as regression tasks using 18 medicinal chemistry datasets, with a randomly chosen learning algorithm for each dataset (for drawing conclusions independent of the learning algorithm). The efficiency of the pattern mining methods had not been systematically investigated in the previous papers. In this paper the efficiency of graph and itemset mining methods is statistically evaluated. Comparisons of the predictive performance of the classification and regression models showed that employing frequent itemset mining results in significant speedups than the graph mining methods, without sacrificing predictive performance, leading to the conclusion that less complex representational forms of graphs indeed save significant computational costs.

1.6 ORGANIZATION OF THE THESIS

The rest of this thesis is organized as follows. The methodology adopted in order to address the research question is discussed in Chapter 2. Chapter 3 gives a description of the proposed methods for graph representation and pattern mining, which are the theoretical contributions of the thesis. Chapter 4 is dedicated to a discussion of the results reported in the nine papers, and finally, the concluding remarks of the thesis along with some possible directions to extend this work are presented in Chapter 5. The publications that include the studies discussed in this thesis are attached in the Appendices A – I.

CHAPTER 2

RESEARCH METHODS

The choice of methods for addressing the research question are discussed in this chapter. It starts with a discussion of the research methodology adopted and the learning framework proposed. The design of the framework, including the choice of datasets as well as learning, evaluation, comparison, and statistical testing methods, are also discussed.

2.1 RESEARCH METHODOLOGY

Research is a rigorous and systematic approach to investigation and study of materials and sources in order to establish relations between natural phenomena or technical problems and reach new conclusions (Oxford, 2010). Methodology refers to the theoretical rationale or the principles that justify the research methods (Carr, 2006). Methodologies for scientific research may be guided by philosophical assumptions under several paradigms, such as, interpretive vs. positivist, empiricist vs. rationalist, qualitative vs. quantitative, etc. (Morgan, 2007). Hevner et al. (2004) discusses another paradigm, namely, behavioural vs. design science, specifically for information systems research. As elaborated in Chapter 1, machine learning concerns the question of how to build a computer system that automatically learns through experience. In other words, it

has the goal of modeling the experience in terms of relations between observations (McCarthy, 2007). In this research, we argue that there is a gap in the knowledge for finding which methods have those efficiency and effectiveness that lie in the Pareto front, and the ways of improving both these properties, knowing that there exist a tradeoff between the two, are sought as means of bridging that gap.

2.1.1 RESEARCH PHILOSOPHY

From the philosophical point of view, there are several aspects that give shape and definition to research. The philosophical stances of claims and assumptions of what exists, or reality (*ontology*), and the ways of acquiring knowledge about reality or how can the assumed knowledge be known (*epistemology*) allow understanding the interrelation between the important strategies of research, namely the strategic approach taken in finding out knowledge and carrying out the research (*methodology*), the techniques or the procedures taken to solve the research question(s) (*methods*), and how and in which ways the methods are used to address the research question (*design*) (Guba, 1990).

In this research we are interested in approaches to pattern mining that result in Pareto improvements in the efficiency vs. effectiveness of the predictive models constructed from data represented in the form of graphs. In doing so, different pattern mining methods are compared. Comparison of different methods involves establishing relationships between the concepts by means of formulating and refuting one or more hypotheses. Such research is often guided by the *positivist* paradigm since solving this research problem concerns establishing relations between theory and evidence (Danks). Positivist research involves falsifiable theories, assuming that there exist real world objects (objective reality) that can be discovered, described or explained using symbols (Cohen and Crabtree, 2006). This approach relies on experimental methods, in contrast to the interpretive approach, which concerns describing the research in a descrip-

tive manner, that does not involve quantifiable statements (Kuhn, 1970). The positivist stance is governed by the epistemological assumptions of *objectivism* (Guba, 1990). Objectivism assumes the existence of both covered and uncovered (absolute) knowledge. Objectivists attempt to find causes/effects/explanations, predicting events and testing theories and hypotheses (Crotty, 1998). This stands in opposition to the subjectivist approach within interpretive research, which seeks understanding and describing events rather than explaining (Blaikie, 2000).

Under the positivist paradigm, *how can the research question be answered*, or the methodology, would be an approach '*that controls the possibility of inquirer bias on the one hand and nature's propensity on the other hand, and, empirical methods that place the point of decision with the nature rather than with the inquirer*' (Guba, 1990). The *experimental approach* allows questions and/or hypotheses to be formulated and falsified by empirical tests. We require a comparison of several pattern mining methods in the context of graph mining, in order to establish relations between efficiency and effectiveness, and therefore we choose experimental research, which is a collection of research designs that includes the manipulation of causes under controlled testing to understand any relations between them (Hinkelmann and Kempthorne, 2007). Empirical studies permit the researcher to study, intensively, the relations between a few variables through designed experiments, by the use of quantifiable statements that could possibly be generalized for real time situations (Galliers, 1991). As stated above, in this study, we assume that the graph data exists, and we try to establish relations between the complexity of the pattern languages, and the efficiency and predictive performance of machine learning methods (by means of testing hypotheses). The methods applied in such research could mainly be quantitative. The research objectives are accomplished by building models that could be used to measure the intended quantities and statistically falsify *hypotheses* using *controls* (Hevner and Chatterjee, 2010).

2.1.2 THE DESIGN SCIENCE PARADIGM

Among the two paradigms classified by Hevner and Chatterjee (2010), as behavioral and design science, machine learning research mostly fall under the design science paradigm as it mainly focuses on design and development of artifacts (Pelillo et al., 2011). Artifacts are theoretical approaches, practical applications or combinations of both. Theories can be based on mathematical foundations or algorithms or concepts, which can be mathematically proven, while practical applications include developing software, tools or programs that could be empirically evaluated, comparing them with the performance of other methods used in the area of application. In addressing the research question in this thesis, different representational forms of graphs are examined and several methods are thereby proposed. This may be viewed as developing algorithms. The approach followed is, thus, building artifacts that could be empirically evaluated in compliance with the seven guidelines illustrated in Hevner and Chatterjee (2010), as described below.

1. Design as an artifact. A viable artifact must be produced in the form of a model, method or an instantiation. We develop artifacts, which are included in a framework consisting of several methods for data representation and pattern mining. The framework is presented in the next section and the methods proposed are given in the next chapter.

2. Problem relevance. In Chapter 1, we have discussed why it is important to address this research question, and how answering the research question will fill a certain knowledge gap in the field. This discussion explains the relevance of the problem.

3. Evaluation of the design. A rigorous assessment of the designed artifact in terms of *utility*, *quality*, and *efficacy* using well-executed evaluation methods is carried out as illustrated in the succeeding sections. The artifacts are compared with each other as well as with the standard methods in terms of the efficiency and performance of the methods. As

described in the preceding section, hypotheses are formulated to study their relations. Statistical tests are used to try to refute the hypotheses. Descriptions of the methods used for comparison are presented in Section 2.4 and the choices of the statistical tests are discussed in Section 2.5.

4. Contributions of the research. Effective research in design science must produce clear and verifiable contributions to the field. A set of artifacts are developed in order to answer the research question, which are the contributions of the research. The outcome of the studies are summarized and how the research findings contribute to fill a knowledge gap is discussed in detail in the concluding chapter.

5. Research rigor. Design science research requires rigorous methods for both the design and the evaluation of the artifact. Several methods are proposed to fulfill the objectives set in this thesis, as discussed in the next chapter, and these methods are evaluated rigorously, using several datasets. The evaluation is carried out as stated under guideline 3, and the descriptions of the datasets used for evaluation are presented in Section 2.3.

6. Design as a search process. During this research, we have tested the efficiency and effectiveness of several methods. Some methods performed better than the other methods. We have selected the methods that performed well for further experiments. Descriptions of the proposed methods are given in the next chapter. The analysis of the performances of the methods is presented in Chapter 4.

7. Research communication. The discoveries have been disseminated into the research community using nine peer reviewed publications. These publications are attached in the appendices of this thesis.

2.2 LEARNING FRAMEWORK

The focal point of study in this thesis is how to improve the efficiency and effectiveness of methods for learning predictive models from graphs. Two objectives were set to answer the question: improving efficiency, and improving predictive performance. In this thesis we propose several artifacts. With respect to the first objective, different representational forms of graphs that are computationally less complex to handle than graphs and pattern mining methods, which are described in the next chapter are proposed. Further, with respect to the second objective, methods for representing background knowledge to enhance performance without increasing the complexity of the pattern mining algorithms (again, described in Chapter 3) are also proposed. The evaluation of the performance of these artifacts is based on comparisons of predictive performances of some standard methods. Several experiments were designed for evaluating these proposed artifacts, each of which involves testing a hypothesis that *there is no difference between the efficiency/predictive performance of the proposed methods and the standard methods used for comparison*.

Predictive models are built from the proposed artifacts as well as standard methods presented in Section 2.4 using a learning framework presented in Section 2.2.1. The performance of these models for several datasets described in Section 2.3 are statistically evaluated using the tests described in 2.5.

2.2.1 THE FRAMEWORK

The process of learning from graphs using pattern mining methods is illustrated in Figure 2.1. We refer to this setup as a *framework*, where different combinations of the methods for graph representing, pattern mining and learning would instantiate the framework.

In Figure 2.1, the transformation of graphs into one of the proposed representational forms followed by discovering patterns is highlighted

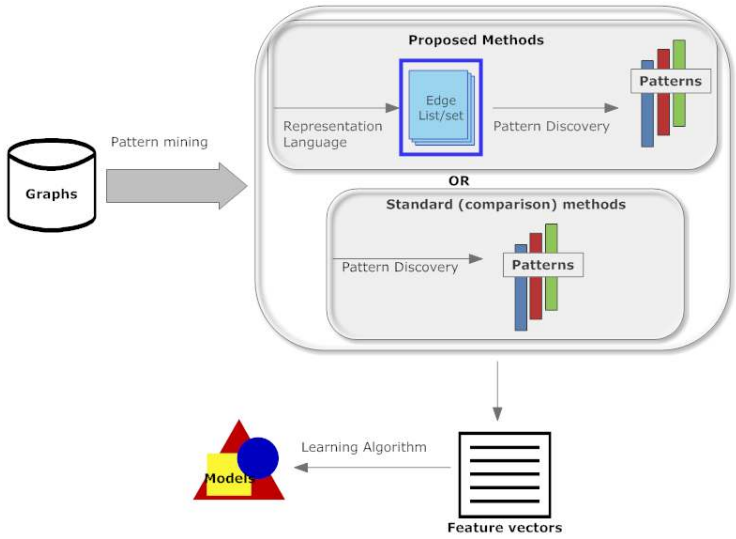

Figure 2.1: The learning framework.

under proposed methods. For standard pattern mining methods, graphs are the input. The pattern set[1] discovered by the pattern mining method is used as the set of attributes to represent the graphs in a propositional form. Any standard machine learning algorithm can thereafter be used for building predictive models, as described below.

2.2.2 LEARNING ALGORITHMS

One of the main advantages of learning from feature sets (resulting from pattern mining methods) is the flexibility of being able to choose any machine learning algorithm for model building. There exist several

[1] In this thesis, the terms 'pattern set' and 'feature set' are used interchangeably.

machine learning algorithms that can be used for prediction tasks. In our experiments, the learning algorithms were chosen among random forests (Breiman, 2001), support vector machines (Platt, 1999), and the k-nearest neighbour algorithm (Aha et al., 1991) to build the classification models. For building regression models, support vector machine for regression (Vapnik, 1999) is used. Although any machine learning algorithm can be used for building the predictive model when learning from graphs using pattern mining methods, our choice of these learning algorithms was based on the acceptance of them for building predictive models with higher accuracy compared to the other methods (Saitta and Sebag, 2010).

For validating the predictive models, a k-fold cross validation or, splitting the dataset into training and testing sets, which uses the complete dataset for model building, and evaluates the model using a validation set, is commonly used (Luger, 2002). Splitting the datasets into training and test sets, or using a separate set for validation may be a choice when the datasets contain a large number of instances. The k-fold cross validation splits the dataset into k folds (subsets), where, for each fold, the remaining k-1 folds are used for building the model that is validated using the fold left out. When k= the size of the dataset, the cross validation is called leave-one-out, where one example is used for validation, and repeated until all the examples are considered in validation. Leave-one-out may however be computationally complex when the dataset is large, due to increased repetitions of training. Instead, k=10 is considered reasonable (Kohavi, 1995). The data set is separated into ten folds and the machine learning algorithm is iterated ten times with nine of those ten folds taken as training data and the remaining set (test set) to validate the model.

There are several ways of analyzing the performance of a machine learning algorithm. Model accuracy, which is the percentage of correctly classified instances (examples), precision, recall, Receiver Operating Characteristics curve (ROC), area under ROC (AUC), and the F-score are some of the commonly used metrics (Flach, 2003). Among them, precision, recall and the

F-score are widely used when the datasets are large and imbalanced (i.e., the number of instances in one class can be comparatively larger than that of the other class), or the prediction accuracy of one class is more important than that of the other class (Omary and Mtenzi, 2009). Predictive accuracy and the area under ROC are popular for measuring the performance of classification tasks (Flach, 2003). We have selected predictive accuracy as the performance measure since it is a simple measure that considers each and every data instance as equally important and shows the percentage of accurate predictions from all the classes. Further, accuracy is a meaningful measure when the datasets are approximately balanced and none of the classes are prioritized. The percentage of correctly classified test instances of each of the ten iterations in a ten-fold cross-validation are averaged for obtaining the model performance. When the task is related to regression, mean squared error (MSE), root mean squared error (RMSE), relative absolute error etc., are used as measures of the error of regression models. Among them, we choose RMSE, which gives the mean squared difference of the estimated and observed values for each of the instances, since it is a widely used measure. According to (Hyndman and Koehler, 2006), '*often the RMSE is preferred to the MSE as it is on the same scale as the data*'.

2.3 DATASETS

As pointed out in Chapter 1, we focus on learning from a set of (small) graphs. Most of the datasets of this type can be found in the domain of medicinal chemistry (chemoinformatics), where the number of nodes in each graph rarely exceeds a few hundred. However, these graphs, i.e., graphs corresponding to chemical compounds, contain plenty of nodes with the same label. For example 3,4,4'-trinitrobiphenyl given in Figure 1.3(b), contains 28 atoms (nodes) of which 12 contain label C. Pattern mining methods are often challenged by the large search space and/or by the need for a subgraph isomorphism test in handling this type of data, as

discussed in Chapter 1.

Datasets in the UCI repository (Bache and Lichman, 2013), which are widely used in machine learning research barely contain any datasets of small graphs, and therefore we have picked appropriate datasets available in other databases. Descriptions of these datasets are given below.

2.3.1 DATASETS OF SMALL GRAPHS

The datasets given in Table 2.1 are bench-marked for pattern mining from small graphs. The majority of them are from medicinal chemistry, and are used to examine the relation between the chemical structure of a compound and its effects, such as solubility, permeability, protein binding, mutagenicity, carcinogenicity, metabolic stability and so on. These chemical effects are presented as real values as well as categorical values in most of the datasets.

2.3.2 THE NCI REPOSITORY

The NCI repository (ChemDB, 2008), contains about 70,000 compounds, categorized into 72 (overlapping) datasets, which inhibit the growth of different human tumor cells. Sixty datasets out of this 72 are used in several studies including (Ralaivola et al., 2005). Table 2.2 gives the names and numbers of positive and negative examples in these datasets.

2.3.3 SYNTHETIC DATASETS

As pointed out in the previous chapter, frequency based pattern mining algorithms usually encounter issues of completing the mining process when the size (number of nodes and edges), the number of node and edge labels, and the number of repeated node labels increases, due to the construction of a huge search tree and thereby a large pattern set (Hasan and Zaki, 2009). The main idea behind using synthetic datasets that contain large graphs of large size is to study how large the discovered

Table 2.1: Summary of the datasets used in the papers included in the thesis.

Dataset Name	Data set size	#posi -tive	#nodes in the larg -est graph	#nodes in the small -est graph	Ave. graph size	#vert -ex la -bels
ace*	114	57	79	18	42	7
ache*	111	55	77	41	56	7
bzr*	163	82	52	23	36	8
caco*	100	50	196	6	45	8
cox2*	322	161	48	32	41	8
dhfr*	397	199	60	20	41	8
gpb*	66	33	45	22	32	8
nct*	131	72	44	8	20	8
therm*	76	38	94	13	52	6
thr*	88	44	101	47	68	5
AI**	69	35	32	20	24	8
AMPH1 **	130	65	104	73	87	5
ATA **	94	47	34	13	22	5
COMT **	92	46	32	11	20	7
EDC **	119	60	43	8	19	7
HIVPR **	113	57	64	26	45	9
HIVRT**	101	51	35	19	25	9
HPTP **	132	66	50	24	38	9
Mutagen -ecis[x]	188	125	40	12	31	8
carcinog -enecis[y]	298	162	80	22	52	18
trains[z]	20	10	19	12	16	16
Satellite faults[p]	600	300	300	40	40	40

*(ChemDB, 2008), **(Mittal et al., 2009), [x](Debnath et al., 1991), [y](Srinivasan et al., 1997), [z](Michie et al., 1994), [p](Muggleton and Feng, 1992)

Table 2.2: Sixty datasets from the NCI repository (ChemDB, 2008).

Dataset*	+ve	− ve	Dataset*	+ve	−ve
786-0	1,832	1,674	NCI-H226	1,781	1,683
A498	1,782	1,698	NCI-H23	1,968	1,751
A549	1,901	1,833	NCI-H322M	1,765	1,925
ACHN	1,795	1,736	NCI-H460	2,049	1,550
BT-549	1,399	1,379	NCI-H522	2,138	1,435
CAKI-1	1,865	1,715	OVCAR-3	2,001	1,690
CCRF-CEM	2,217	1,263	OVCAR-4	1,840	1,742
COLO-205	1,943	1,702	OVCAR-5	1,651	2,019
DU-145	1,416	1,529	OVCAR-8	1,979	1,735
EKVX	1,968	1,713	PC-3	1,522	1,460
HCC-2998	1,804	1,373	RPMI-8226	2,116	1,448
HCT-116	2,049	1,674	RXF-393	1,850	1,551
HCT-15	1,993	1,738	SF-268	2,020	1,701
HL-60-TB	2,188	1,198	SF-295	2,027	1,718
HOP-62	1,888	1,740	SF-539	1,920	1,464
HOP-92	1,982	1,521	SK-MEL-28	1,774	1,950
HS-578T	1,550	1,320	SK-MEL-2	1,783	1,817
HT29	2,004	1,708	SK-MEL-5	2,034	1,651
IGROV1	1,956	1,734	SK-OV-3	1,711	1,792
K-562	2,139	1,479	SN12C	1,918	1,764
KM12	1,941	1,764	SNB-19	1,840	1,885
LOX-IMVI	2,053	1,550	SNB-75	2,131	1,359
M14	1,815	1,736	SR	1,869	1,137
MALME-3M	1,886	1,621	SW-620	1,940	1,813
MCF7	1,733	1,306	T-47D	1,550	1,359
MDA-MB-231	1,475	1,473	TK-10	1,650	1,840
MDA-MB-435	1,519	1,462	U251	2,044	1,711
MDA-N	1,503	1,459	UACC-257	1,873	1,808
MOLT-4	2,175	1,359	UACC-62	2,046	1,638
NCI-ADR-RES	1,586	1,525	UO-31	1,994	1,621

*Columns with labels +ve and −ve represent the numbers of instances in positive (cancer) and negative (non-cancer) classes respectively.

pattern set would be in frequency based pattern mining methods used in the experiments. Synthetic graphs are generated using the graph generation software (Cheng et al., 2006). We have used three different graph set-ups referred to as *g*, *h* and *i*, as described below:

g: graph size 20, 20 node labels, 20 edge labels, edge-density 0.3, number of edges 100

h: graph size 200, 200 node labels, 200 edge labels, edge-density 0.3, number of edges 1,000

i: combination of graphs of size up to 4,000 (2,000 node labels, 100 edge labels, edge-density 0.2, number of edges 2,000) and graphs of size up to 3,000 (1,000 node labels, 200 edge labels, edge-density 0.2, number of edges 2,000)

For each category *g*, *h* and *i*, 10, 100, 1,000, 5,000 and 10,000 graphs are included (e.g., g10, g100, g1000, g5000 and g10000 and so on).

2.4 COMPARISON METHODS

2.4.1 STANDARD GRAPH MINING METHODS

According to the guidelines of Hevner and Chatterjee (2010), the proposed artifacts have to be evaluated rigorously for their utility, quality and efficacy. In doing so, several graph pattern mining methods are compared. The choice of these comparison methods are based on their approaches to improving the efficiency of the pattern mining (the main point of discussion in learning from graphs is the computational complexity, which affects the efficiency), the availability of the method as a software tool, and the use of the method in pattern mining (a state-of-the-art method). Accordingly, the following methods were selected.

GraphSig (Ranu and Singh, 2009): This method discovers significant patterns from graph databases. GraphSig was selected as a comparison method since it introduces a method of efficient mining by avoiding the subgraph isomorphism test. Significant patterns are selected using a probability measure (the $p-$value) related to the support of each graph in the graph database. If this $p-$value lies below a user defined threshold, the graph is considered significant. The mining process includes clustering the dataset using domain knowledge (class labels) and finding frequent subgraphs in those clusters. The graphs are converted into a feature space and the significant graphs are discovered from the feature space. The conversion of the graphs into the feature space results in a loss of structural information, but avoids the generation of large sets of random graphs and the calculation of the frequency of the query graph. The clustering approach allows GraphSig to overcome the scalability issue in discovering subgraphs with low frequencies.

Molecular Fragment miner (MoFa) (Borgelt, 2002): This method is used for discovering frequent molecular fragments in chemoinformatics databases. MoFa claims to be efficient since it considers ring structures in the molecules as single units and uses *wildcard atoms* (atom types that are chemically equivalent), during the search for significant fragments. The algorithm searches for arbitrarily connected subgraphs, avoiding frequent embeddings of previously discovered subgraphs by using a specific search strategy. The algorithm maintains parallel embeddings of a fragment in all molecules throughout the growth process and exploits a local order of the atoms and bonds of a fragment to effectively prune the search. MoFa selects subgraphs that have a certain minimum support in a given set of molecules, i.e., they are part of at least a certain percentage of the molecules. However, in order to restrict the search space, the algorithm considers only connected subgraphs, i.e., subgraphs for which all vertices are (directly or indirectly) connected by edges.

SUBDUE (Cook and Holder, 1994): This method was selected as a comparison method since it avoids subgraph isomorphism by finding interesting subgraphs using the so-called minimum description length, as stated in the first chapter. This is a graph-based knowledge discovery system that finds structural and relational patterns in data represented as entities and relations. It uses the minimum description length (MDL) principle to measure the interestingness of the subgraphs discovered. SUBDUE employs a step-by-step procedure, which starts from a single vertex and performs a computationally constrained beam search in order to expand the considered subgraphs by other vertices or edges. It aims at generating small sets of subgraphs that optimally compress the dataset.

RSD (Lavrač et al., 2002): Relational Subgroup Discovery (RSD) is an ILP based method. We have selected this method since it has an option to construct a set of features from graph data represented as logic programs, which are used as attributes in the transformed propositional representation. Similar to the methods used in this thesis, any propositional learner can then be used for learning the predictive model.

MolFea (Helma et al., 2003): Molecular feature miner (MolFea) is a method specific to the chemoinformatics domain for mining molecular fragments, which, again, could be used as features for any machine learning algorithm.

gSpan (Yan and Han, 2002): A widely used method in frequent subgraph mining (approximately 7000 hits in Google Scholar). Also, gSpan transform graphs into sequences as described in Chapter 1.

GASTON (Nijssen and Kok, 2004): This method can discover subgraphs in terms of paths, trees and cycles, thereby improves the efficiency compared to the other approaches of frequent pattern mining, as stated in the first chapter.

2.4.2 DOMAIN SPECIFIC METHODS

As stated in Section 2.3, the majority of the datasets used in our studies are from the domain of medicinal chemistry. In this domain there exist methods to represent compounds as a set of features. We call these methods as domain specific methods. The reasons for why we consider these methods for comparing with pattern mining methods used in the context of graph learning are that they also represent chemical datasets as attribute-value encodings, followed by building predictive models. Further, these are the state-of-the-art methods used in medicinal chemistry for prediction tasks. We have chosen two domain specific methods that produce such feature sets (called descriptor sets), namely, SELMA (Olsson and Sherbukhin, 1999), and ECFI (Rogers and Hahn, 2010), which are described below. The choice of these descriptor sets are based on the availability and high performance of the predictive models shown in the literature (Karunaratne et al., 2013).

Extended Connectivity Fingerprints (ECFI) (Rogers and Hahn, 2010): ECFI represents structural fragments of various sizes. Molecules are encoded as an array of binary values or counts. These structural fragments are computed using an iterative updating procedure on the non-hydrogen atoms of the structure, which is based on the number of atoms in the neighbourhood, the atomic number, the attached hydrogen count, etc. In the initial assignment stage, each non-hydrogen atom of the structure is assigned an integer identifier. During each iteration, larger and larger circular neighbourhoods (of atoms) around each atom are covered and the respective identifiers are updated. After removing the duplicate identifiers, the remaining set of identifiers are collected into a list. This integer list could be used directly for model building or converted into a fixed length vector containing binary values. A commonly used ECFI descriptor set contains 1024 binary attributes.

SELMA (Olsson and Sherbukhin, 1999): The SELMA descriptor set is a collection of commonly used 2D molecular descriptors related to molecular

size, flexibility, ring structure, connectivity, polarity, charge, lipophilicity, hydrogen bonding etc. This collection includes 94 such descriptors.

2.5 STATISTICAL TESTS

As described in the methodology section, our research is based on the assumptions of the positivest stance, quantitative methods, thereby falsifying hypotheses in order to answer the research question. Hypotheses are set to compare predictive performances (or efficiency) of the pattern mining methods, and statistical tests 1) to refute the null hypothesis set in the experiment, and 2) to identify which methods cause the rejection of the null hypothesis (in the case of the null hypothesis being rejected), are used in the respective studies.

Prior to statistical testing, the methods are ranked (in terms of each dataset) so that a method that achieves the best performance gets the lowest rank and the worst performance gets the highest rank. The average rank for each method is obtained by averaging over the individual ranks obtained from each dataset. For the hypothesis tests, the $F-$ statistic is used on the average of the ranks of the performance measures of each of the methods compared. The $F-$distribution can be used for testing the variances of two (or more) distributions (Demsar, 2006). The following two equations from (Demsar, 2006), referred to in (Garcia and Herrera, 2008),

$$\chi^2{}_F = \frac{12N}{k(k+1)}\Big[\sum_{j=1}^{k} R_j^2 - \frac{k(k+1)^2}{4}\Big]$$

and

$$F_F = \frac{(N-1)\chi^2{}_F}{N(k-1)-\chi^2{}_F}$$

are used for testing all hypotheses. Here $\chi^2{}_F$ is the Friedman statistic, N is the number of datasets we used in the experiment, R_j^2 is the average rank

(average of the individual ranks from all the datasets) of the j^{th} pattern mining method, and k is the number of methods we compared in the experiment. F_F is the refined $F-$statistic defined in (Demsar, 2006). If the computed statistic $F_F > F_{((k-1),(k-1)(N-1),\alpha)}$, the tabulated F value with respect to $(k-1)$ and $(k-1)(N-1)$ degrees of freedom in the standard table of F distribution at significance level α, the null hypothesis is rejected.

In the second part of the statistical testing, i.e., in the case of finding the cause of rejection of the null hypothesis, a test is used to compare, pairwise, which methods perform differently. The two-tailed Nemenyi test (Demsar, 2006) is used for this test, since it allows multiple comparisons, i.e., produces a set of statistical inferences simultaneously. Accordingly, if the statistic z between two methods i and j with average ranks R_i and R_j, i.e.,

$$z = \frac{(R_i - R_j)}{\sqrt{\frac{k(k-1)}{6N}}}$$

exceeds the value called the critical difference, i.e.,

$$CD = q_\alpha \sqrt{\frac{k(k-1)}{6N}},$$

where q is the tabulated value corresponding to α in the table of critical values of Nemenyi test (Demsar, 2006), then the performances of the two methods are significantly different. The method with the lowest average rank in such a case is considered to outperform the other method.

2.6 EXPERIMENTAL DESIGN

As stated in the previous sections, the methods proposed for efficient and effective learning from graphs are evaluated empirically. The design science approach is followed and the methods are evaluated quantitatively by formulating and testing hypotheses. The general setup of the experiments

is presented in Figure 2.2.

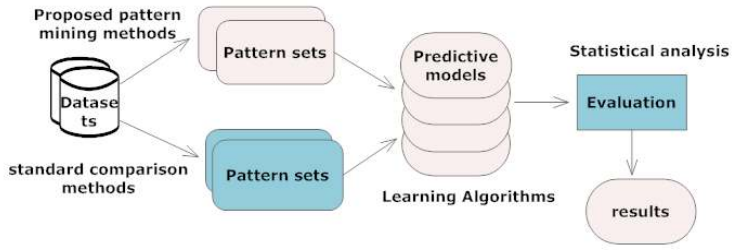

Figure 2.2: Experimental design.

The general process of the experiments includes formulating a hypothesis, selecting one or more datasets, selecting one or more pattern mining methods from the proposed methods and comparison methods, discovering pattern sets, selecting one or more learning methods, and building prediction models followed by evaluating the predictive performances statistically. The results would be whether the null hypothesis is refuted or not, and in the case of the null hypothesis being rejected, the methods that perform differently are identified, which leads to a conclusion as to which methods outperform the others. The specific descriptions of how each and every experiment was carried out are further given under the respective empirical evaluations presented in Chapter 4.

2.7 SUMMARY

This research was designed under the philosophical assumptions of the positivist stance. A design science approach was followed. This chapter described the basis for the selection of the research methodology and the choices of the methods for learning, evaluating and comparing. It also included a brief description of the datasets used in the experiments.

CHAPTER 3

PROPOSED METHODS

The proposed methods for representing graphs, discovering patterns and incorporating domain-specific background knowledge into the graphs are presented in this chapter. Experiments designed to evaluate the proposed methods in terms of efficiency and effectiveness are also included.

3.1 GRAPHS AS A REPRESENTATION LANGUAGE

The mathematical foundations of graphs are rooted in the classical graph theory. Attributed graphs with an unrestricted label alphabet are the most general way to define graphs (Cook and Holder, 2006).

Definition 3.1: Graph

A labeled graph G is defined as a quadruple (V, E, λ, μ), where V is the set of nodes, E is the set of edges and $\lambda : V \rightarrow L_v$ and $\mu : E \rightarrow L_e$ are the labeling functions for nodes and edges respectively, given the alphabets of nodes and edges, L_v and L_e. Further, an edge e from $u \in V$ to $v \in V$ is denoted by $e = (u, v)$, if $(u, v) \in E$. For undirected graphs, $(u, v) = (v, u)$.

Definition 3.2: Subgraph

A subgraph G_s of $G = (V,E,\lambda,\mu)$, denoted by $G_s \subseteq G$, is a graph $G_s = (V_s,E_s,\lambda_s,\mu_s)$, such that $V_s \subseteq V, E_s \subseteq E$, $\lambda_s(v) = \lambda(v) \forall v \in V$ and $\mu_s((u,v)) = \mu((u,v)) \forall (u,v) \in E$.

As stated in Chapter 1, although graphs is a rich and informative representation language for data with complex structures, pattern mining from graph data is a computationally costly process. Transforming graphs into trees and sequences has been shown to be beneficial in reducing this cost in the literature, as discussed in Chapter 1. Less expressive representations for graphs than sequences, with attempts to minimize the loss of information during the transformation, are proposed in this thesis.

3.2 ITEMSETS AND ITEMSET MINING

As a representational form, *Itemset* is a set of literals (features), such as a set of names of the commodities in a grocery shop for example. A transaction, i.e, the commodities purchased by a person, is a subset of the itemset. A dataset that includes several such transactions is a set of subsets of the itemset. The term itemset first appeared in conjunction with the problem of market basket analysis, which studies interesting patterns from transaction databases. Itemset mining has been identified as an important pattern mining problem since the 1980s (Han and Kamber, 2000). It includes counting different items in purchase lists of a store, and searching for associations among two or more items (Agrawal and Srikant, 1994). However, itemset representation is limited to the form of a set of subsets, and thereby, in general, may not include any relation between the features. Therefore, itemset representation is less expressive than sequences, trees and graphs, as stated in Chapter 1.

Mining itemset patterns from a dataset of itemsets, on the other hand, is computationally less expensive than mining graphs (Thomas, 2010) from

a graph dataset, although graphs and itemsets mining basically follow the same procedure, i.e., iterative process of expanding the search space, candidate generation, support calculation and evaluation. As discussed in Chapter 1, graph pattern mining is costly, mainly due to subgraph matching. Matching two itemsets, in contrast, is less expensive than matching subgraphs, since a subset of an itemset is again a set, and hence, does not involve any subgraph isomorphism test.

3.3 THE ITEMSET APPROACH TO THE REPRESENTATION OF GRAPHS

Typical itemsets are not expressive enough to encode the relations (edges) in graphs as stated above. Therefore, a graph in an itemset representation may simply be the node set of the graph. Recall the graph in Figure 1.1. This graph could be represented as $\{c,cl,n,o,s\}$. This representation leads to a loss of a considerable amount of information of the graph structure, namely, 1) the relations between the nodes (edges), 2) the information about repeated items (nodes with identical labels) and 3) the topology of the graph, i.e., the arrangement of the nodes and edges within the graph structure. The more of this information is included in the itemset representation, the less the loss of information due to the transformation of graphs into itemset will be.

3.3.1 EDGE LIST

The term *edge list* is given to the representation language that allows including the relation between the nodes (edges) in the itemset (node set). The approach to representing a graphs as an edge list was introduced in Paper I. Prior to defining the edge list formally, let us consider the examples below.

Example 3.1:

Consider the chemical graph of a benzene ring in Figure 3.1(left) below.

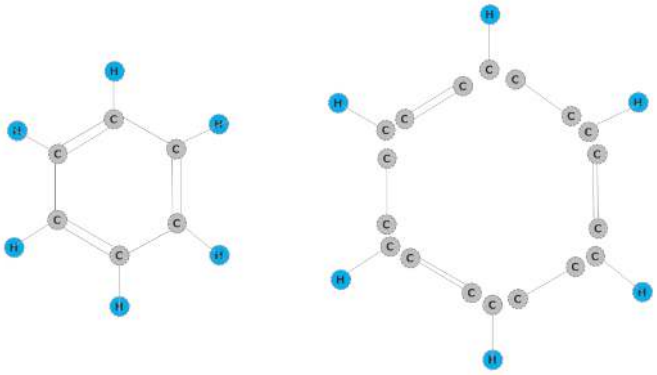

Figure 3.1: Graph structure of the benzene ring (left) and how the structure is split by transforming it into the edge list (right).

The graph in Figure 3.1 (left) is split into fragments where one fragment represents an edge of the graph and two nodes connected to the edge. All such fragments created are included in a list, in the form of $(node\_label, node\_label, edge\_label)$. For example, two carbon atoms connected by a double bond is a fragment $C = C$, and is represented by $(C, C, 2)$. Each element of the list is therefore a triplet that contains the (lexicographically or numerically) ordered labels of two nodes, followed by the edge label, and thereby the graph in Figure 3.1 (left) has the edge list *L= ((C,C,1),(C,C,1), (C,C,1), (C,C,2), (C,C,2), (C,C,2), (C,H,1), (C,H,1), (C,H,1), (C,H,1), (C,H,1), (C,H,1))*. Here 1 is the label assigned for a single bond in the benzene ring and 2 is for the double bond.

Example 3.2:

The graph in Figure 3.2(left) has a unique feature in that the node labels are not repeated. The edges in this graph are labeled by 1. The transformation of the fragments in Figure 3.2(right) into a list would yield *L= ((A,B,1), (A,F,1), (B,C,1), (C,D,1), (D,E,1), (E,F,1))*. Note that since this graph has unique node labels, the original graph can be reconstructed from the edge list, unlike the graph in Example 3.1.

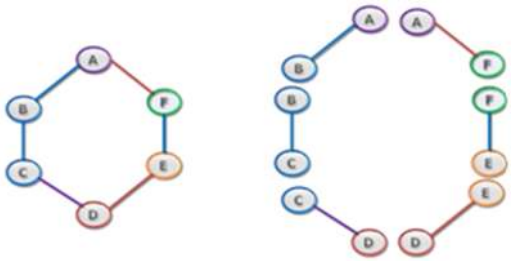

Figure 3.2: Arbitrary graph containing unique node labels (left) and its edge fragments (right).

Example 3.3:

Figure 3.3 contains some graphs from a chemical graph database. Each graph in this graph database can be transformed into an edge list as given below. The same procedure described in the previous two examples is followed, i.e., splitting the graph into node-edge-node fragments, and aggregating all such fragments in to a list.

$$
\begin{aligned}
L(G_1) &= ((C,N,1),(C,N,2),(C,S,1),(N,O,1)) \\
L(G_2) &= ((C,C,1),(C,N,1),(C,N,2),(C,S,1),(C,S,1),(C,S,1)) \\
L(G_3) &= ((C,C,1),(C,O,1),(C,O,1),(C,O,2),(C,S,1))
\end{aligned}
$$

Here, $L(G_1), L(G_2), L(G_3)$ are the edge lists of the graphs G_1, G_2 and G_3 in Figure 3.3. Let us formally define the method demonstrated in the examples.

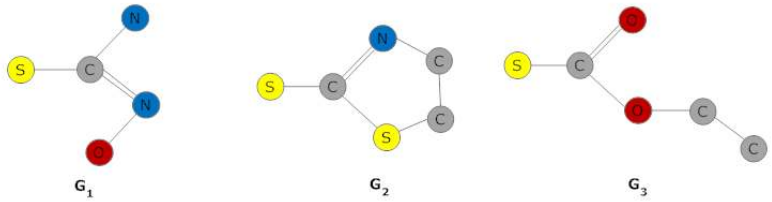

Figure 3.3: Chemical graphs.

3.3.2 CONSTRUCTION OF THE EDGE LIST

The list of edges, referred to as the *edge list*, of a graph is constructed according to Definitions 3.3 or 3.4 below.

Definition 3.3: The edge list of a directed graph

The edge list L of a directed graph $G = (V, E, \lambda, \mu)$ is:

$$L = \begin{cases} (\lambda(v_i), \lambda(v_j), \mu(e_k)) & \text{for all } v_i, v_j \in V \text{ such that } e_k = (v_i, v_j) \in E \\ \lambda(v_i) & v_i \text{is not connected to any other node in the graph (no edges)} \end{cases}$$

Definition 3.4: Edge list of an undirected graph

The edge list L of an undirected graph $G = (V, E, \lambda, \mu)$, where for any $e_k = (v_i, v_j) \in E$, there exists $e_k = (v_j, v_i)$, is defined as

$$L = \begin{cases} (\lambda(v_i), \lambda(v_j), \mu(e_k)) & \text{for all } v_i, v_j \in V \text{ such that } e_k = (v_i, v_j) \in E \\ & \text{and } \lambda(v_i) \prec \lambda(v_j) \\ \lambda(v_i) & v_i \text{ is not connected to any other node in the} \\ & \text{graph (no edges)} \end{cases}$$

We define each unit $(\lambda(v_i), \lambda(v_j), \mu(e_k))$, or, $(\lambda(v_i)$ (if v_i has no edges), to be an *element* of the edge list. In this thesis, a graph is an undirected graph, unless stated otherwise.

Definition 3.5: Vertex Label

A vertex label is a sequence of literals, i.e, a word,

$$X = x_1, x_2, \ldots, x_k$$

where k is a positive integer and each x_i is an element from the set $a, b, c, \ldots, x, y, z$ or a language where an ordering on the characters is defined. The integer k is called the length of the word X, i.e., *length (X)*. Hereafter, we use 'word' and 'label' interchangeably.

Definition 3.6: Lexicographical order of words

The lexicographic order of two words X and Y is defined by the relation

$$X \prec Y$$

If $length(X) = 1$,
let $X = x$ and $Y = \alpha Z$, where Z is any word and $\alpha \in \{a, b, \ldots, x, y, z\}$.

First we order the alphabet:

$$\{a \prec b \prec c \prec \cdots \prec x \prec y \prec z\}$$

Then $X \prec Y$ if and only if :$\alpha \succeq y$

If $length(X) \neq 1$: Suppose that $X = x_1, x_2, \ldots, x_k$ and $Y = y_1, y_2, \ldots, y_l$ are two words. Then

$$X \prec Y$$

if and only if there is a non-negative integer t such that:

$$\begin{gathered} \forall i = 1, \ldots, t-1 : x_i = y_i \text{ and} \\ x_t \prec y_t \text{ or,} \\ \forall i = 1, \ldots, k : x_i = y_i \text{ and } l > k \end{gathered}$$

For example, considering the usual order on the English alphabet, the following words are ordered as

$$a \prec aa \prec aaa \prec ab \prec aba \prec abb \prec abba$$

3.3.3 THE EDGE SET

When the graphs are represented as edge lists, some of the structural information of the graphs may be lost. If a graph contains nodes with identical node labels, the corresponding elements in the edge list may be identical. These identical elements will vanish when the edge list is considered as an itemset (e.g., in any subsequent use with itemset mining algorithms for discovering patterns). For example, the edge list given in Example 3.1, i.e., $((C,C,1),(C,C,1),\ (C,C,1),\ (C,C,2),\ (C,C,2),\ (C,C,2),\ (C,H,1),\ (C,H,1),\ (C,H,1),\ (C,H,1),(C,H,1))$ is transformed into a set with elements $\{(C,C,1),\ (C,C,2),\ (C,H,1)\}$. The *edge set* is what we define to accommodate these identical elements of the edge list. This representation method is used in the experiments reported in *Papers VIII* and *IX*.

Lexicographically ordered edge list

Suppose $W^1 = (X^1, Y^1, Z^1)$ and $W^2 = (X^2, Y^2, Z^2)$, are two elements from the edge list, where X^1, Y^1, X^2 and Y^2 are node labels and Z^1 and Z^2 are edge labels. Then

$$W^1 \prec W^2$$

if and only if,

$$\begin{array}{lll} i). & X^1 \prec X^2 & \text{or,} \\ ii). & X^1 = X^2 \text{ and } Y^1 \prec Y^2 & \text{or,} \\ iii). & X^1 = X^2 \text{ and } Y^1 = Y^2 \text{ and } Z^1 \prec Z^2 & \end{array}$$

The edge set

Consider the lexicographically ordered edge list

$$L = (W^1, W^2, \ldots, W^n)$$

The identical elements in this edge list L are indexed as follows.

Each element in the edge list is a tuple $\langle W, i \rangle$, where W is the element and $1 \leq i \leq n$, where n is the length of L.

Therefore, the indexed edge list, when, say, a is the number of repetitions of W^1 in L, and so on, is

$$L_{index} = \left(\sum_{i=1}^{a} \langle W^1, i \rangle, \sum_{i=1}^{b} \langle W^2, i \rangle, \ldots, \sum_{i=1}^{m} \langle W^r, i \rangle \right)$$

where $a + b + \cdots + m = n$, and r is the number of different elements in L. L_{index} is a set, and is represented by the *edge set* S as follows.

$$S = \left\{ \sum_{i=1}^{a} \langle W^1, i \rangle, \sum_{i=1}^{b} \langle W^2, i \rangle, \ldots, \sum_{i=1}^{m} \langle W^r, i \rangle \right\}$$

Example 3.4:

Consider the edge list *L = ((C,C,1),(C,C,1), (C,C,1), (C,C,2), (C,C,2), (C,C,2), (C,H,1), (C,H,1), (C,H,1), (C,H,1), (C,H,1), (C,H,1))* given in Example 3.1. This has several identical elements out of the three elements *(C,C,1), (C,C,2)* and *(C,H,1)*. According to the edge set representation *S*, with *n=12, r=3*, *a=3, b=3* and *m=6*, the resulting edge set is,

$\{(C,C,1,1),(C,C,1,2),(C,C,1,3),(C,C,2,1),(C,C,2,2),(C,C,2,3),$
$(C,H,1,1),(C,H,1,2),(C,H,1,3),(C,H,1,4),(C,H,1,5),(C,H,1,6)\}$

Therefore, the edge set retains the repeated node labels (thereby repeating elements of the edge list). However the edge set may not be capable of holding the topological structure of the graphs when there are identical node labels. For example, the graph in Figure 3.1(left) cannot be reconstructed using the edge set while Figure 3.2(left) may possibly be reconstructed.

3.4 PATTERN MINING METHODS

An *element* of the edge set (list) is analogous to an *item* in the itemset representation. Therefore, an edge list is analogous to a transaction in itemset mining. Item, itemset and the itemset mining problem (Agrawal and Srikant, 1994) can therefore be represented in terms of elements and edge sets (lists) as follows.

Definition 3.9:

Let an *element* of the edge set *S* of a graph *G* be an *item*. Let the database *D* consist of edge sets of the graphs in the graph database and *I* be the set of items from *D* where $X \subseteq I$ is an itemset. Then, *D* is a multiset of subsets of *I*. For itemset *X*, an edge set including *X* is an occurrence of *X* and the $support(X)$ is the number of edgesets in which *X* is present.

This definition replaces an item in the conventional itemset by a fragment of a graph, i.e, a triplet that consists of a node–edge–node relation (an *element*), and an itemset by a set of such fragments.

3.4.1 MAXIMAL FREQUENT ITEMSET (MFI)

Let D, and X be as in Definition 3.9. The problem of frequent item set mining thereby becomes finding the frequent set, $F(D,\sigma)$, given D, I and σ (the *minimum support*) where

$$F(D,\sigma) := \{X \subseteq I | \text{support}(X,D) \geq \sigma\}$$

Further, a frequent itemset X which is included in no other frequent itemset is called a *maximal frequent itemset (mfi)*.

In this thesis, hereafter, we use the term 'itemset' presumably in accordance with Definition 3.9, i.e., an item in an itemset is an element of the edge set (list), unless stated otherwise.

Example 3.5

Consider the chemical dataset in Example 3.3. The edge sets of the three graphs are

$$\begin{aligned} S(G1) = &\ \{(C,N,1,1),(C,N,2,1),(C,S,1,1),(N,O,1,1)\} \\ S(G2) = &\ \{(C,C,1,1),\ (C,N,1,1),\ (C,N,2,1),(C,S,1,1),\\ &\ (C,S,1,2),\ (C,S,1,3)\} \\ S(G3) = &\ \{(C,C,1,1),(C,O,1,1),(C,O,1,2),(C,O,2,1),(C,S,1,1)\} \end{aligned}$$

and the itemset,

$$\begin{aligned} I = &\ \{(C,N,1,1),(C,N,2,1),(C,S,1,1),(N,O,1,1),(C,C,1,1),\\ &\ (C,S,1,2),(C,S,1,3),(C,O,1,1),(C,O,1,2),(C,O,2,1)\}. \end{aligned}$$

Given $\sigma = 2$, the frequent itemset and the maximal frequent itemset of chemical dataset are given below.

Frequent itemsets	Maximal frequent itemsets
{(C,N,1,1), (C,N,2,1), (C,S,1,1)}	{(C,N,1,1), (C,N,2,1), (C,S,1,1)}
{ (C,N,1,1), (C,N,2,1)}	{(C,C,1,1), (C,S,1,1)}
{(C,N,2,1), (C,S,1,1)}	
{(C,N,2,1)}	
{(C,N,1,1), (C,S,1,1)}	
{(C,N,1,1)}	
{(C,C,1,1), (C,S,1,1)}	
{(C,C,1,1)}	
{(C,S,1,1)}	

3.4.2 SUPERVISED MAXIMAL FREQUENT ITEMSETS

When the data mining problem is a classification task, each element in the graph database is assumed to be associated with a value for the target variable (class). Maximizing the correlation between the target variable and the discovered pattern is a commonly used pattern selection/evaluation approach. Whenever the class attribute is present, *mfi* could be refined in the following manner.

Supervised Maximal Frequent Itemset (*Smfi*)

Suppose each example in the graph database D is associated with a target variable (class) c_j, where $1 \leq j \leq m$, and m is the number of different classes. The graph database D can then be presented as,

$$D = \sum_{1}^{m} (D_j, c) | c \in \{c_1, \ldots, c_m\}$$

where $D_j \subset D$ is the cluster of the graphs that are associated with the class c_j. Let $I_j \subseteq I$ be the itemset corresponding to the j^{th} class. Then, the respective frequent itemset would be,

$$F\left(\sum(D_j, c = c_j), \sigma\right) := \{X \subseteq I_j | \text{support}(X, (D_j, c = c_j)) \geq \sigma\}$$

Then,

$$mfi_{(c=c_j)} = \textit{maximal set of} \{X \subseteq I_j | \text{support}(X, (D_j, c = c_j)) \geq \sigma\}$$

where the *maximal set* consists of the X that are included in no other frequent itemset, and

$$Smfi = \sum_{1}^{m}(mfi_{(c=c_j)})$$

3.4.3 CONSTRAINT PROGRAMMING ON EDGE SETS

Use of various monotonic and anti-monotonic constraints is proposed in Nijssen and Kok (2004), as an alternative measure to frequency. This approach has the flexibility of allowing the user to define which constraint, among several, is used in pattern discovery (Nijssen et al., 2009). For example, maximal and closed frequent itemsets can also be transformed into constraint based problems (Guns et al., 2011). In the framework of constraint programming, itemsets are represented as a set of binary vectors, and, supposing for example that the itemset is the edge set as in Definition 3.9, then $I = \{1, \ldots, m\}$ and $n = |D|$ is the size of the database D of edge sets, then D can be represented as a binary matrix of size $n \times m$.

The discriminative pattern mining problem is therefore defined as:

Definition 3.10:

Given a database D, a measure of discrimination f and a threshold parameter σ, the discriminative itemset mining problem is to discover all itemsets in

$$\{I | f(I) \geq \sigma\}$$

Definition 3.11:

Suppose that database D contains two target variables positive and negative, where the subset of D such that the target variable is positive (from the binary class) is D^+, and D^- for the negative class. Then an itemset I is discriminative if and only if the discriminative measure f with threshold σ for the given variables,

$$p = \sum_{t \in |D^+|} T_t$$

and

$$n = \sum_{t \in |D^-|} T_t$$

where T_t is the transaction of the t^{th} entry of the database, satisfies

$$f(p,n) \geq \sigma$$

In this method $f(p,n)$ is called the bounded PN space. The PN space is used together with the constraints, coverage and support as defined in (Guns et al., 2011) to find correlated patterns.

3.4.4 MAXIMAL COMMON SUBSTRUCTURES

The method called 'maximal common substructures' measures the similarity between two graphs (edge sets) in terms of the number of common elements in the edge sets of the two graphs.

Maximal common substructures (*mcs*)

Let S and S' be the edge sets of the graphs G_1 and G_2 respectively. The maximal common substructure between G_1 and G_2 is

$$S_{mcs_{(G_1,G_2)}} = \{S \cap S'\}$$

Let D be the graph database and $|D|$ be the size of D. Then the set of maximal common substructures is

$$S_{mcs} = \left\{ \sum_{i=1}^{|D-1|} \sum_{j=i}^{|D|} S_{mcs_{(G_i,G_j)}} \right\}$$

For a given maximal and minimal support σ_{max} and σ_{min}, *mcs* represent the set of substructures that satisfy

$$\forall i, j, \sigma_{max} \leq F(S_{mcs_{(G_i,G_j)}}) \leq \sigma_{min}$$

Here, $F(S_{mcs_{(G_i,G_j)}})$ is the number of edge sets in D such that $S_{mcs_{(G_i,G_j)}}$ is present.

Example 3.6

Recall the chemical dataset in Example 3.3. The edge lists of the three graphs are,

$$\begin{array}{lcl} L(G1) & = & ((C,N,1),(C,N,2),(C,S,1),(N,O,1)) \\ L(G2) & = & ((C,C,1),(C,N,1),(C,N,2),(C,S,1),(C,S,1),(C,S,1)) \\ L(G3) & = & ((C,C,1),(C,O,1),(C,O,1),(C,O,2),(C,S,1)) \end{array}$$

The corresponding edge sets are

$$\begin{array}{lcl} S(G1) & = & \{(C,N,1,1),(C,N,2,1),(C,S,1,1),(N,O,1,1)\} \\ S(G2) & = & \{(C,C,1,1),(C,N,1,1),(C,N,2,1),(C,S,1,1),(C,S,1,2),(C,S,1,3)\} \\ S(G3) & = & \{(C,C,1,1),(C,O,1,1),(C,O,1,2),(C,O,2,1),(C,S,1,1)\} \end{array}$$

The set of S_{mcs} is

$$\begin{array}{lcll} S_{mcs}^{(G_1+G_2)} & = & \{S(G_1) \cap S(G_2)\} & = & \{(C,N,1,1),(C,N,2,1),(C,S,1,1)\} \\ S_{mcs}^{(G_1+G_3)} & = & \{S(G_1) \cap S(G_3)\} & = & \{(C,S,1,1)\} \\ S_{mcs}^{(G_2+G_3)} & = & \{S(G_2) \cap S(G_3)\} & = & \{(C,C,1,1),(C,S,1,1)\} \end{array}$$

The set of *mcs* when $\sigma_{max} = 2$ and $\sigma_{min} = 1$ is

$$\{(C,N,1,1),(C,N,2,1),(C,S,1,1)\}$$
$$\{(C,C,1,1),(C,S,1,1)\}$$

3.5 PROPOSED METHODS

In this section, we present several methods proposed for the framework in Figure 2.1, using the itemset representation methods for graphs, *edge list* and *edge set*, and, pattern mining with *mfi*, *Smfi*, *CP*, and *mcs*, as discussed in the preceding sections.

3.5.1 MAXIMAL COMMON SUBSTRUCTURES (*mcs*)

The method of maximal common sub structures, *mcs*, defined in Section 3.4.4 is used in papers *I–V*. In these papers, the edge list is used for discovering *mcs*. The maximal common substructure discovery algorithm (*Paper I*) is shown in the algorithm below.

Data: edge lists of graphs
Result: MaximalCommonSubstructures
$j = 1$;
while $j \leq n-1$ **do**
  $k = j+1$;
  while $k \leq n$ **do**
    $s[j,k] = s[k,j] = f_j \cap f_k$;
    add $s[j,k]$ and $s[k,j]$ to *MaximalCommonSubstructures*;
    $k++$;
  end
  $j++$;
end

Algorithm 1: Maximal common substructure search algorithm.

A framework called **DI**scovery of **F**eatures using **F**ing**ER**prints (DIFFER) is constructed in *Paper I*, which includes transforming graphs into edge list (fingerprint), discovery of maximal common substructures using mcs algorithm, and build predictive models using the feature vectors constructed from the discovered *mcs* set. Here, minimal support σ_{min} is set to 5% (a maximal support threshold is not applied).

3.5.2 INFORMATION GAIN (IG) FOR *mcs*

The maximal common substructure search algorithm does not make use of class labels when selecting the features, but only ensures that the selected features fulfill the user-defined upper and lower bounds (maximal and minimal support σ_{max} and σ_{min}, as defined in Section 3.4.4). Instead of relying on these bounds, another method is proposed that calculates the information gain of each of the features generated by the maximal common substructure search algorithm and chooses the n most informative features, where n is a parameter of the method. The best 10, 20, 50 and 100 substructures are used in model building. 10-fold cross-validation on the training set is used to determine what number of substructures to use when generating the model that is evaluated on the test set. This method is used in the study presented in *Paper V*.

3.5.3 MAXIMAL FREQUENT ITEMSETS (*mfi*)

The maximal frequent itemset mining method defined in Section 3.4.1 is implemented using the maximal frequent itemset mining algorithm (Burdick et al., 2001), which is an algorithm that discovers maximal frequent itemsets from transaction databases. The edge set representation is used to transform the graphs into itemsets. A minimum support value σ is required in computing the maximal frequent patterns, as described in Section 3.4.1. Seven values namely, 0.5, 0.4, 0.2, 0.1, 0.025 and 0.01 are used for σ, since a formal method for choosing such a value does not exist. The relatively best support value that provides a pattern set resulting in the best classification model with the training set is selected for building the model that is evaluated on the test set. This method is used in various experiments presented in *Paper V–IX*.

3.5.4 HYBRID MODEL WITH *mfi* AND *mcs*

We have used a hybrid method comprising the two methods *mfi* and *mcs*. Using edge lists, the maximal common substructures are discovered. These mcs's, which are subsets of the edge lists, are used for mining maximal

frequent itemsets. The minimum support $\sigma_{min} = 5\%$ is applied in *mcs*. Support values 0.5, 0.4, 0.2, 0.1, 0.025 and 0.01 are applied for *mfi*. This method is introduced in *Paper V*.

3.5.5 THE VECTOR SPACE MODEL

The vector space model is quite popular within the text classification community for its simplicity. This model, in general, is used in information retrieval from a large set of documents (called document collection), where each document in this set is represented by a vector of terms (one or more words). Therefore, if a document collection consists of the set of terms $T = \{t_1, \ldots, t_n\}$, an arbitrary vector for a particular document could be represented as $d_i = \{w_{(i,1)}, \ldots, w_{(i,n)}\}$ where each $w_{(i,j)}$ corresponds to a *weight* assigned for the respective term t_j within the d_i^{th} document. There are several approaches to obtain the weights, where term frequency is one of them.

The edge list of a graph could be interpreted as a document that consists of a set of terms. Therefore considering each element (X^i, Y^i, Z^i) as a term, we could represent the edge list by a term frequency vector. Here the weight of each element of the feature vector is the frequency of (X^i, Y^i, Z^i) in each edge list. Predictive models are built using these term frequency vectors. This method is introduced in *Paper V*.

3.5.6 SUPERVISED MAXIMAL FREQUENT ITEMSETS (*Smfi*)

The supervised maximal frequent itemset mining method is applied to the datasets containing class labels as described in Section 3.4.2. In this method, the *mfi* algorithm is applied separately on the edge lists from the graphs of different classes. The maximal frequent patterns discovered for each class is aggregated. Similarly to *mfi*, several values, namely, 0.5, 0.4, 0.2, 0.1, 0.025 and 0.01 are used for the minimum support σ, and, the level of support resulting in the highest accuracy, as estimated by 10-fold

cross-validation on the training set, is used for generating the model that is evaluated on the test set. This method was introduced in *Paper VI*.

3.5.7 CONSTRAINT PROGRAMMING BASED ITEMSET MINING (CP)

When the target variable (class label) is available, a constraint on the correlation between the target variable and the set of items could be used to discover discriminative itemsets (from the edge sets) as described in Section 3.4.3. This is an extension of the constraint programming on transaction databases described in (Nijssen et al., 2009). When the graphs are represented as edge sets, the existing algorithm for discriminative itemset mining by Nijssen et al. (2009) could be used on these edge sets. In all the experiments based on the constraint programming approach we have used the complete set of discovered patterns and allowed the learning algorithm to select the most important patterns for model learning. Experiments with constraint programming method CP are included in the studies of *Papers VII, VIII* and *IX*.

3.6 BACKGROUND KNOWLEDGE

As pointed out in Chapter 1, background knowledge can be encoded into graphs as new edge labels for the additional relations and new nodes for additional entities/attributes. SUBDUE (Gonzalez et al., 2003) has used this method as illustrated in Figure 1.7 of Chapter 1. Incorporating new knowledge as new nodes and edges is straightforward, yet this representation may produce very large graphs. Recall that the graph in Figure 1.6, which contains only two atoms, may be expanded to a graph such as Figure 1.7 and a typical molecule may contain about 20 atoms on average. When handling large graphs, the pattern mining algorithm requires several constraints due to the computational demands, resulting in incomplete search, or in missing important relations, as pointed out in Section 1.4.3. Instead of expanding the graphs by new relations of existing elements with new entities or attributes for background knowledge, we propose two different

strategies to incorporate such additional information: embedding the background knowledge into the node label of the graph, or using the additional information as additional features in the feature vector, as discussed below.

3.6.1 ENCODING BACKGROUND KNOWLEDGE INTO GRAPHS

In this approach we propose to incorporate the background knowledge as a part of the existing node label. As illustrated in Figure 3.4 below, the node labels are renamed in terms of the background knowledge used. In this representation, the node labels contain information related to the entities (and their attributes) and edge labels correspond to the relations between entities (and attributes). For example, Figure 1.6 could be renamed as Figure 3.4 (a). The additional information of the atomic value 22 for the carbon atoms may thereby be included as in Figure 3.4 (b). Further, *helide10* and *six_group* may be included in the graph by renaming the nodes (and edges) as in Figure 3.4 (c).

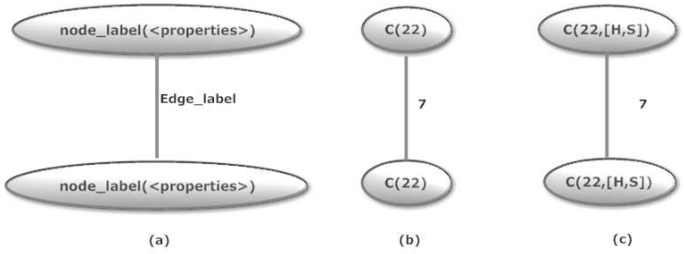

Figure 3.4: (a) General description of a graph, (b) Molecular fragment with atom name and type, (c) Graph including 2-dimensional substructures of the molecular fragment.

3.6.2 USING BACKGROUND KNOWLEDGE AS ADDITIONAL FEATURES TO THE LEARNING ALGORITHM

Adding background knowledge as additional features to the learning algorithm may be quite straightforward since the same method of encoding graphs for pattern mining need not necessarily be applied to encode background knowledge. In this approach the feature sets discovered by a pattern mining method can simply be concatenated with the feature set that encodes the background knowledge. We have used this method in almost all the papers, with different interpretations. In *Papers I–IV*, feature sets from different pattern mining methods are aggregated, arguing that patterns discovered by different pattern mining methods may not be the same, and thereby the feature set from one method could be considered as additional knowledge for any other method. In *Papers VI–IX*, domain specific background knowledge of medicinal chemistry datasets, encoded as feature vectors, are used in conjunction with the pattern sets discovered by pattern mining methods. The encodings of background knowledge into feature sets (chemical descriptors), SELMA (Olsson and Sherbukhin, 1999) and ECFI (Rogers and Hahn, 2010), which are described in Section 2.4.2, are combined with the feature sets obtained by pattern mining methods.

3.7 SUMMARY

Our approaches for efficient and effective learning from graphs are based on methods that represent graphs in forms similar to itemsets, which can be used with itemset mining algorithms for pattern discovery. Pareto improvements to the efficiency are sought by this representation, which is computationally less complex to handle than graphs. To improve the predictive performance, background knowledge is utilized in the pattern sets. The empirical evaluations of these methods are discussed in the next chapter.

CHAPTER 4

EMPIRICAL EVALUATION

This chapter includes an empirical evaluation of the methods we proposed for investigating how to increase the efficiency of pattern mining methods without affecting their predictive performance, and how to enhance the predictive performance without sacrificing their efficiency.

4.1 PREDICTIVE PERFORMANCE

In this research, methods that transform graph data into pattern sets that are computationally less complex to handle are proposed. Predictive performance of the classification and/or regression models built from representations derived from the proposed pattern mining methods are evaluated by comparing them with standard methods, as described in the following sections.

4.1.1 MAXIMAL COMMON SUBSTRUCTURES (*mcs*)

Two studies were conducted to evaluate the method, maximal common substructures (*mcs*) on edge lists. The *mcs* and the framework, which is called DIFFER in *Paper I*, is compared with an Inductive logic programming method, RSD (Lavrač et al., 2002) for four datasets from Table 2.1. Classification models are generated using the feature sets from DIFFER

and RSD in the Random Forest algorithm with 50 trees. In *Paper II*, in addition to RSD, DIFFER is compared with two other graph mining methods, SUBDUE (Gonzalez et al., 2003) and $Tree^2\chi^2$ (Bringmann, 2009) for three datasets from Table 2.1. The results are shown in Table 4.1.

Table 4.1: Comparison of performance of DIFFER with some state-of-the-art methods (*Paper I & II*).

Dataset	Accuracy				
	DIFFER	RSD	SUBDUE -CL	$Tree^2\chi^2$	DIFFER + RSD
Trains	80.00%	75.00%	-	-	85.00%
Mutagenesis	80.61%	88.86%	-	80.26%	92.76%
Carcinogenesis	65.25%	54.37%	61.54%	-	65.33%
Satellite faults	71.43%	71.43%	-	-	80.95%

The column DIFFER + RSD in Table 4.1 shows the prediction accuracy of the models built using the combined feature set of DIFFER and RSD. Results show that DIFFER, which contains *mcs* for pattern mining, is competitive in predictive performance with the standard methods considered in this experiment. Since the maximal common substructures are computationally less complex to handle than graphs (hence efficient), showing competitive predictive performance is a Pareto improvement on efficiency-effectiveness.

4.1.2 SUPERVISED MAXIMAL FREQUENT ITEMSET MINING METHOD (*Smfi*)

Supervised maximal frequent itemsets (*Smfi*) on the edge lists is evaluated using 21 datasets from Table 2.1, in *Paper VI*. The predictive performance of *Smfi* with three classifier algorithms, Random Forest (RF), Support vector machine (SVMP), and the k-nearest neighbour algorithm (KNN), is compared with SUBDUE (Cook and Holder, 1994) and MoFa (Borgelt, 2002). The number of trees generated by the random forest algorithm was

set to 50. Two kernels for the SVM algorithm were investigated; the RBF kernel with complexity 2, and the polynomial kernel with complexity 2. The IBk algorithm with the number of nearest neighbours k = 3 was used as the nearest neighbour classifier. As described in Section 2.2.2, classification accuracy was chosen as the performance criterion, which was estimated using 10-fold cross-validation. The class labels were used for feature construction in all the methods. For *Smfi*, the minimum support was optimized by cross-validation on the training sets, and the optimized parameter was used for the test set. The same training and test folds were used for all methods. Predictive performances of the three methods are statistically compared. The null hypothesis in this experiment is that there is no difference between the predictive performances of the models built using the three mining methods. The null hypothesis is rejected by the Friedman test. Table 4.2 presents the average ranks, each method obtained with respect to each machine learning algorithm. In calculating the average rank, a method that obtained the highest accuracy, with respect to a dataset was given the lowest rank. The average rank for a method is obtained by averaging the ranks of that method over all the datasets.

Table 4.2: The average ranks of the performance of classifier models using 21 datasets (*Paper VI*).

	RF	SVMP	KNN
Smfi	1.24	1.10	1.30
SUBDUE	2.14	2.24	2.10
MoFa	2.62	2.67	2.62

The results of this experiment lead to the conclusion that the proposed method *Smfi* is significantly better than both comparison methods, SUBDUE and MoFa with respect to all the learning models while SUBDUE outperforms MoFa when the models are built using KNN.

4.1.3 MAXIMUM FREQUENT ITEMSET MINING (*mfi*) AND CONSTRAINT PROGRAMMING (CP) BASED METHODS

Classification and regression tasks were designed to compare the model performances of the methods *mfi*, CP, SUBDUE (Cook and Holder, 1994), MoFa (Borgelt, 2002) and graphSig (Ranu and Singh, 2009) in *Paper VII*, using 18 datasets from Table 2.1. For classification, one of the learning algorithms from random forests (RF), support vector machine with non-polynomial kernel (SVMP) and radial bias kernel (SVMR), and the k-nearest neighbour classifier (KNN) along with the same parameter settings as the previous experiment, was randomly chosen for each of the datasets, i.e., for each dataset, all the standard and proposed methods build predictive models using the same (randomly chosen) learning algorithm. For regression, the SVM algorithm for regression problems (SMOReg) with two different parameter settings were chosen arbitrarily, i.e., the nonlinear polynomial kernel with complexity 2 and the RBF kernel with complexity 2. The root mean squared error (RMSE) using 10-fold cross validation is ranked for statistically evaluating the methods. Figure 4.1 presents the classifier accuracies of the classification models for 18 datasets. The learning algorithm used for model building is given in the parentheses next to the name of the dataset.

In comparing the performances of the methods the null hypothesis was that there is no difference between performances of the methods. The significance of the differences of the classification accuracies was tested by comparing the average ranks using the Friedman test (cf. Section 2.5). The Friedman test did not reject the null hypothesis for this experiment, showing that there is no significant difference between the predictive performances of the methods compared. We also have summarized, in Table 4.3, the performances of the five methods with respect to pairwise wins of each method. In Table 4.3, the first value of each cell gives the wins of the method in the column label over the method in the row label, and the second value is the losses of the same. For example, the numbers 7 and 9 in the cell in row 2 and column 1 correspond to wins of *mfi* over

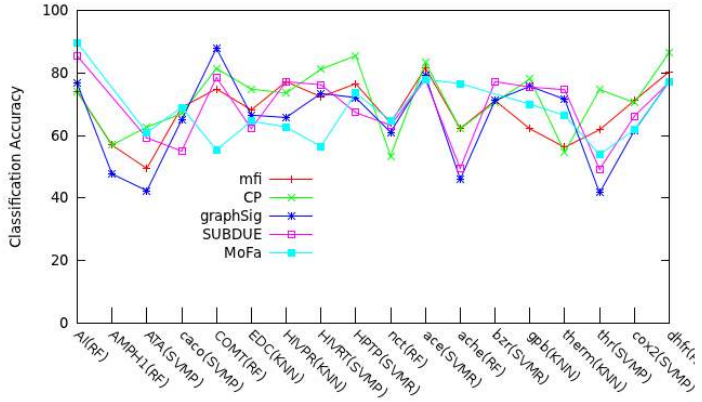

Figure 4.1: Classification accuracies of the classifier models.

CP in 7 instances out of 18, and 9 losses for *mfi* over CP. The cell in row 1, column 2 gives the opposite, i.e., the win of CP over *mfi* followed by the losses of CP over *mfi*. Also, there are two ties since CP wins 7 out of 18 and *mfi* wins 9 out of 18 (18- (7+9) = 2).

Table 4.3 shows that the methods *mfi* and CP achieved more wins than those from the graph mining methods. Further, the results lead to the conclusion that the proposed methods *mfi* and CP have not shown any statistically significant reduction in predictive performance of classification models, due to the use of less complex form than graphs to improve their efficiency.

Regression models

Figure 4.2 shows the results for the predictive performance of regression models for the same experiment as above. The null hypothesis is again not rejected by the Friedman test, thereby showing that there are no significant differences among the performance levels of the methods. The wins/losses

Table 4.3: Pairwise comparison of performance of methods (*Paper IX*).

	mfi	CP	graph -Sig	SUB -DUE	MoFa
mfi	-	7/9	5/13	7/10	6/11
CP	7/9	-	4/14	5/13	5/13
graphSig	13/5	14/4	-	11/7	7/10
SUBDUE	10/7	13/5	7/11	-	9/9
MoFa	11/6	13/5	10/7	9/9	-

for pairwise comparisons of the methods are presented in Table 4.4.

Table 4.4: Pair-wise comparison of performance of methods (*Paper IX*).

	mfi	CP	graph -Sig	SUB -DUE	MoFa
mfi	-	9/8	6/11	8/9	4/13
CP	8/9	-	6/11	9/8	4/14
graphSig	11/6	12/6	-	10/8	6/10
SUBDUE	9/8	8/9	8/10	-	5/13
MoFa	13/4	14/4	10/6	13/5	-

Again, *mfi* and CP have more wins than the other methods. The relatively high computational cost of the graph mining methods compared to the itemset mining approaches can again not be motivated by a corresponding gain in predictive performance. On the contrary, the itemset mining approaches appear to have Pareto improvements to efficiency–effectiveness compared to graph mining approaches.

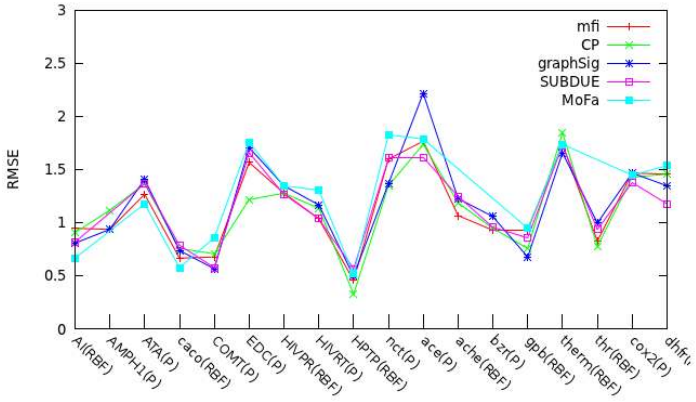

Figure 4.2: Root Mean Squared errors (RMSE) of regression models.

4.1.4 PATTERN SETS FROM DIFFERENT METHODS

Which of the performances of those itemset mining and graph mining methods are closer to the Pareto frontier with respect to a specific problem in the medicinal chemistry domain, i.e., the quantitative structure–activity relationship modeling is investigated in *Paper VIII*. In this experiment, specific methods for the particular domain in handling such problems were also considered. Therefore, the domain specific feature sets SELMA (Olsson and Sherbukhin, 1999) and ECFI (Rogers and Hahn, 2010), itemset mining methods *mfi* and CP and graph mining methods SUBDUE, MoFa and graphSig were compared. Again, two separate experiments were designed for classification and regression. Learning algorithms were randomly picked as described in a previous experiment. The results were statistically compared (cf. Section 2.5) using the null hypothesis that there is no difference between the performances of the methods. The results of the predictive performances lead to the rejection of the null hypothesis set for this experiment, showing that there exists differences between the performances of the methods. Further tests using the differences of

ranks show that the pairs presented in the gray cells in Table 4.5, have significantly different performances. Here, a positive value indicates that the method in the column label outperforms the method in the row label, and vice versa for negative values. The classification models showed significant differences against graphSig, as seen from Table 4.6. All the other methods turned out to perform similarly.

Table 4.5: Differences of average ranks of performance of regression models (*Paper VIII*).

	ECFI	SELMA	*mfi*	CP	graph -Sig	SUB -DUE	MoFa
ECFI	-						
SELMA	2.17	-					
mfi	2.17	0.00	-				
CP	2.33	0.17	0.17	-			
graphSig	3.17	1.00	1.00	0.83	-		
SUBDUE	2.39	0.22	0.22	0.06	-0.78	-	
MoFa	3.28	1.11	1.11	0.94	0.11	0.89	-

4.1.5 EFFICIENCY OF PATTERN MINING METHODS

In *Paper IX* we investigate to what degree the efficiency of the pattern mining methods vary, using 18 datasets from Table 2.1. Two proposed methods, *mfi* and CP and three standard methods, SUBDUE, MoFa and graphSig were compared in this experiment. The CPU time consumed for the completion of the mining algorithm of each method (in seconds) was recorded. A null hypothesis was set that there is no significant difference between the execution times of the methods. The Friedman test (cf. Section 2.5) was used to test the hypothesis on the ranks of the execution times. The method with the lowest CPU time was assigned rank 1 while the method with the maximum CPU time ranked the highest. A method that

Table 4.6: Differences of average ranks of performance of classification models (*Paper VIII*).

	ECFI	SELMA	*mfi*	CP	graph -Sig	SUB -DUE	MoFa
ECFI	-						
SELMA	0.5	-					
mfi	1.44	0.94	-				
CP	0.78	0.28	-0.67	-			
graphSig	3.17	2.67	1.72	2.39	-		
SUBDUE	2.06	1.56	0.61	1.28	-1.11	-	
MoFa	1.67	1.17	0.22	0.89	-1.5	-0.39	-

failed to complete was also assigned the highest rank. The null hypothesis is rejected for this experiment. Table 4.7 gives the differences of the ranks for all the pairwise tests. Further tests to investigate which pairs of methods perform significantly differently compared to each other using the Nemenyi test (Section 2.5, resulted in the pairs corresponding to the gray-colored cells in Table 4.7 performed significantly different, where a positive value indicates that the method in the row label outperformed the method in the column label, and vice versa for negative values. Both the itemset mining methods, *mfi* and CP have significantly higher efficiency than the graph mining methods.

Summarizing the results, it can be concluded that the proposed pattern mining methods, which are computationally less complex to handle than graphs are efficient in mining patterns. Also, any reduction of the predictive performance is not showed up as a result. Therefore, the proposed methods are closer to the Pareto front than the standard methods compared in the experiments.

Table 4.7: Differences of average ranks of efficiency of the graph and itemset mining (*Paper IX*).

	Itemset mining methods		Graph mining methods		
	mfi	CP	SUBDUE	MoFa	graphSig
mfi	-	0.67	2.11	2.89	3.22
CP	-0.67	-	1.44	2.22	2.56
SUBDUE	-2.11	-1.44	-	0.78	1.11
MoFa	-2.89	-2.22	-0.78	-	0.33
graphSig	-3.22	-2.56	-1.11	-0.33	-

4.2 ENHANCING THE PREDICTIVE PERFORMANCE OF PATTERN MINING METHODS

Enhancing the predictive performance of the pattern mining methods is sought by 1) different pattern language settings and 2) background knowledge. We propose using the background knowledge 1) as a part of the node and edge definitions or, 2) as additional features to the learning algorithm. Empirical evaluation of these methods are presented below.

4.2.1 PATTERN LANGUAGE SETTINGS VS. PREDICTIVE PERFORMANCE

The aim of this study is to identify which of the different methods we used for encoding graphs lead to predictive models that are comparatively better/worse. The experiment presented in *Paper V* compares the predictive performance of the methods that use different forms of the edge list, namely, *mfi*, *mfi* + *mcs* (combined set of *mfi* and *mcs* features), *mcs*, subset of *mcs* derived using information gain measure (ig) and the edge list in attribute value format (*vs*) as described in Section 3.5. In addition, a baseline representation of the inputs (*av*), which transforms graphs into typical

itemsets by ignoring edges, is also used for comparison. Fifteen structured datasets from Table 2.1 are used for building classification models using the Random forest algorithm (Breiman, 2001). Their classification accuracies are compared statistically with the null hypothesis that there is no difference among the predictive performances of the methods compared. The average ranks of the six methods are presented in Table 4.8. The null hypothesis is rejected in this experiment, due to the differences in the performance of the pairs with respect to the gray cells in Table 4.9. This experiment lead to the conclusion that the way of representing a graph as an itemsets results in significant differences in predictive performance.

Table 4.8: Average ranks for the structured datasets (*Paper V*).

Method	av	*mfi*	*mfi+mcs*	ig	vs	*mcs*
Average rank	5.00	2.03	2.17	3.50	4.13	3.87

Table 4.9: Differences of average ranks (*Paper V*).

Method	av	*mfi*	mfi+mcs	ig	vs	*mcs*
av	-					
mfi	2.97	-				
mfi+mcs	2.83	0.14	-			
ig	1.5	1.47	1.33	-		
vs	0.87	2.1	1.96	0.63	-	
mcs	1.13	1.84	1.7	0.37	-0.26	-

4.2.2 INCORPORATING BACKGROUND KNOWLEDGE INTO NODE AND EDGE LABELS

In this approach the node definitions are expanded to accommodate the additional background information. Five different levels of background

knowledge have been used in the experiments as given below (*Paper IV*).

D1: Each node is labeled with an atom's name, its type, and a set of bonds by which the atom is connected to other atoms. The node definition for D1 is represented by (atom name, atom type, [bond type/s]). For example, a node in Figure 1.6 representing a carbon atom of type 22, which is connected to other atoms by an aromatic bond is labeled with $(c, 22, [7])$, where 7 denotes the aromatic bond. No edge label is associated with this representation (or all edges can be considered to have the same label *connected*).

D2: The amount of information used for encoding is similar to D1, but the node and edge labels are different. Each node label in D2 is of the bonds by which the atom is connected to other atoms and it can be represented by (*atom name, atom type*). The edges are labeled with the bond type by which two atoms are connected. For example, a node representing a carbon atom of type 22, which is connected to two other atoms by one single and one double bond is labeled with $(c, 22)$, and the edges to the nodes corresponding to the other atoms are labeled with single and double, respectively.

D3: Edge set of the graphs with node and edge labels similar to D2 is used.

D4: In addition to node–edge labels, the atom's presence in complex structures such as benzene rings or nitro groups is also included in the node labels. Accordingly, the node label for D4 would be (*atom name, atom type, [list of complex structures]*). For example, a carbon atom of type 22 that is part of a *nitro group*, a *benzene group*, and a *5-aromatic ring* is labeled with $(c, 22, [nitro, benzene, 5-aromaticring])$. The edge labels are the same as for D2.

D5: Edge set of the graphs with node and edge labels similar to D4 is used.

Predictive models are built using the random forest and support vector machine algorithms for mutagenesis and carcinogenesis datasets in Table 2.1. The results presented in Table 4.10 show that the predictive performances of the models increase with the addition of more background knowledge. The learning algorithm used for model building is given in the parentheses next to the model accuracy. The difference in accuracy between the lowest and highest levels of background knowledge is statistically significant. There is almost no difference in the accuracies of the graph encodings D1 and D2, reflecting the fact that these encodings do not differ in information content, but only in their formulation of the node labels.

Table 4.10: Accuracy of the models using different levels of background knowledge (*Paper IV*).

Data Set	Accuracy %				
	D1	D2	D3	D4	D5
Mutagenesis	80.61 (RF)	80.61 (RF)	84.04 (SVM)	87.77 (SVM)	88.3 (SVM)
Carcinogenesis	61.25 (RF)	61.1 (RF)	68.73 (SVM)	71.03 (SVM)	75.0 (SVM)

4.2.3 BACKGROUND KNOWLEDGE AS ADDITIONAL FEATURES

Background knowledge encoded in the form of so-called chemical descriptors are used in this experiment. In *Paper VI*, descriptor sets of the methods ECFI (Rogers and Hahn, 2010) and SELMA (Olsson and Sherbukhin, 1999) are combined with the feature sets of *Smfi*, SUBDUE and MoFa. The predictive performances of the combined models are statistically tested (separately for three classifiers, Random forest, Support vector machine and k- nearest neighbour, with the same parameter settings used in the other experiments) with the null hypothesis that there is no difference between the predictive performances of the combined models

with constituent models, using Friedman test (Section 2.5). In *Paper VIII* a similar experiment is designed with methods *mfi*, CP, SUBDUE, MoFa and graphSig. The graphs in Figure 4.3 show the results when background knowledge in the form of ECFI and SELMA feature sets, are incorporated as additional features into the feature vector from pattern mining methods. The same experiment was repeated for regression models and the results are presented in Figure 4.4.

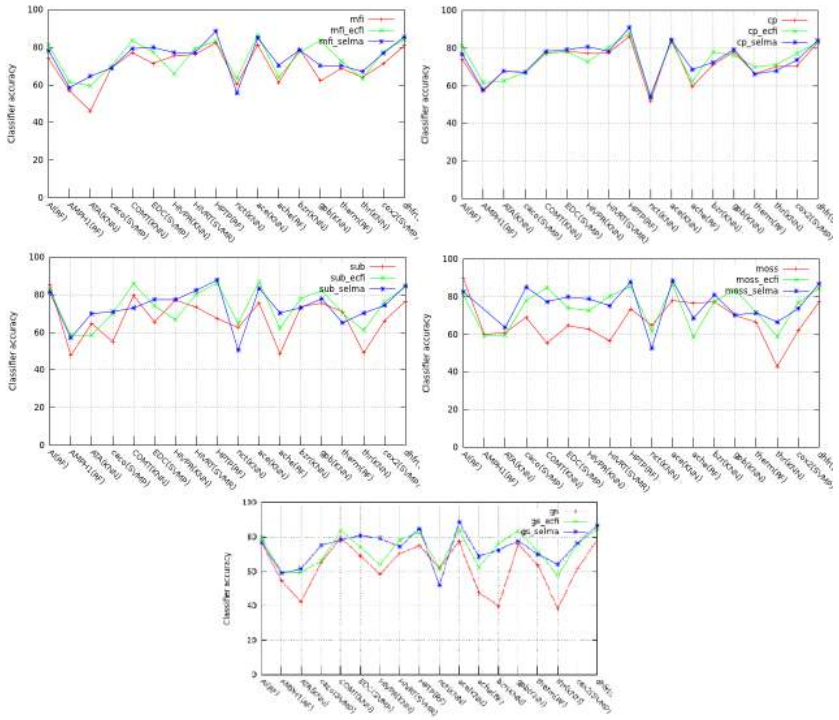

Figure 4.3: Classification accuracies of combined feature set of background knowledge and pattern mining methods for methods *mfi*, CP, SUBDUE, MoFa and graphSig.

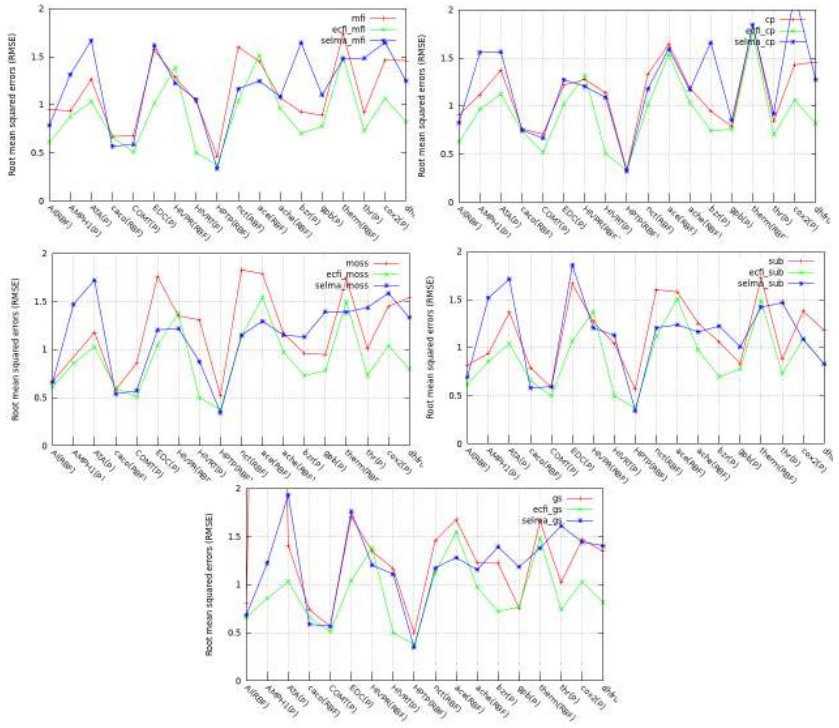

Figure 4.4: For regression models.

According to Figure 4.3 and Figure 4.4, the majority of the pattern mining methods achieve better predictive performance by either increased accuracy or reduced error when the domain specific background knowledge is used in conjunction with the features produced by their own method.

4.2.4 FEATURE SETS FROM DIFFERENT PATTERN MINING METHODS AS BACKGROUND KNOWLEDGE

Due to the differences in the pattern sets discovered by different pattern mining methods, any pattern set could contain additional information

with respect to another pattern set, which, in principle, could be used as background knowledge. In almost all the papers, investigations of the predictive performance of models by adding different pattern sets are included. Domain specific features, features from graph mining methods and features from itemset mining methods are compared in *Paper VIII*. The combined sets were used as features for the classification and regression models. The learning algorithms were selected randomly for each dataset. The models generated from the combined feature sets for 18 datasets from Table 2.1 were compared with the models generated from the constituent feature sets. For example, a model generated from the combination of CP and SUBDUE features was compared to a model generated from SUBDUE alone and a model generated from CP alone. Table 4.11 and Table 4.12 show for each combined model and a model generated from one of its constituent feature sets, for how many datasets the combination resulted in an improvement in predictive performance for regression and classification models respectively. For example, the number 14 in the cell in row 2 and column 1 in Table 4.11 means that models using SELMA+ECFI resulted in a higher predictive accuracy than models generated from SELMA alone for 14 out of the 18 data sets, and the number 11 in the cell in row 1 column 2 means that the models using ECFI+SELMA resulted in a higher predictive accuracy than the models developed using ECFI alone for 11 out of the 18 datasets.

Adding the feature sets from the domain specific method ECFI, to the other feature sets helped increase the predictive performance of the resulting models for most of the datasets. Thus, the ECFI feature set can be used to enhance the predictive performance of regression and classification models.

4.2.5 EFFICIENT AND EFFECTIVE LEARNING

From the experiments presented in the previous section, it was shown that 1) by using a less complex representational form than graphs, the efficiency can be improved without affecting the predictive performance, and, 2) by

Table 4.11: The number of datasets which RMSE is reduced with the combined model (*Paper VIII*).

	ECFI	SELMA	*mfi*	CP	graph -Sig	SUB -DUE	MoFa
ECFI	-	11	5	7	4	3	4
SELMA	14	-	10	8	9	11	11
mfi	16	9	-	8	11	13	9
CP	16	10	7	-	8	11	6
graphSig	16	11	16	10	-	12	11
SUBDUE	17	10	14	11	9	-	14
MoFa	16	13	14	13	12	16	-

Table 4.12: The number of datasets which the model accuracy is increased with the combined classifier model (*Paper VIII*).

	ECFI	SELMA	MFI	CP	graph -Sig	SUB -DUE	MoFa
ECFI	-	10	7	9	2	8	4
SELMA	11	-	12	8	10	9	11
MFI	15	15	-	8	8	10	10
CP	10	14	2	-	4	10	6
graphSig	17	15	13	14	-	13	12
SUBDUE	14	12	13	14	10	-	10
MoFa	14	14	14	13	12	11	-

incorporating background knowledge the effectiveness can be improved without adding complexity to the pattern mining, thereby efficient and effective learning could be obtained.

4.3 PERFORMANCE ANALYSES OF THE PROPOSED METHODS

The proposed methods for pattern mining in this thesis have shown Pareto improvements in efficiency and effectiveness compared with several standard methods, as described in the previous sections and the papers included in this thesis. In this section the proposed methods are compared with each other to see which of these methods show Pareto improvements in efficiency and effectiveness.

4.3.1 PERFORMANCE COMPARISONS OF *mfi*, *Smfi* AND CP

Figure 4.5 and 4.6 present the comparisons of the performances of the classification and regression models for *mfi* and CP and *mfi* and *Smfi* respectively. The predictive performances of the models built using *mfi* and CP are not significantly different, leading to the conclusion that both *mfi* and CP lead to models that perform equally well. The regression models with *mfi* and *Smfi* are almost identical, but the classification models show that *Smfi* is slightly ahead of *mfi*.

The comparisons between *mfi*, *Smf*i and CP did not conclude in favour of any of the methods. Nevertheless one may select *mfi* over *Smfi* due to the extra work involved in the computation of the *Smfi*i's (i.e., applying *mfi* separately to data that is split into different classes). CP may be penalized compared to *mfi* sometimes by the large set of discovered patterns. However it is possible to find out whether smaller subsets bounded by top-k patterns of CP would provide similar (or enhanced) performance.

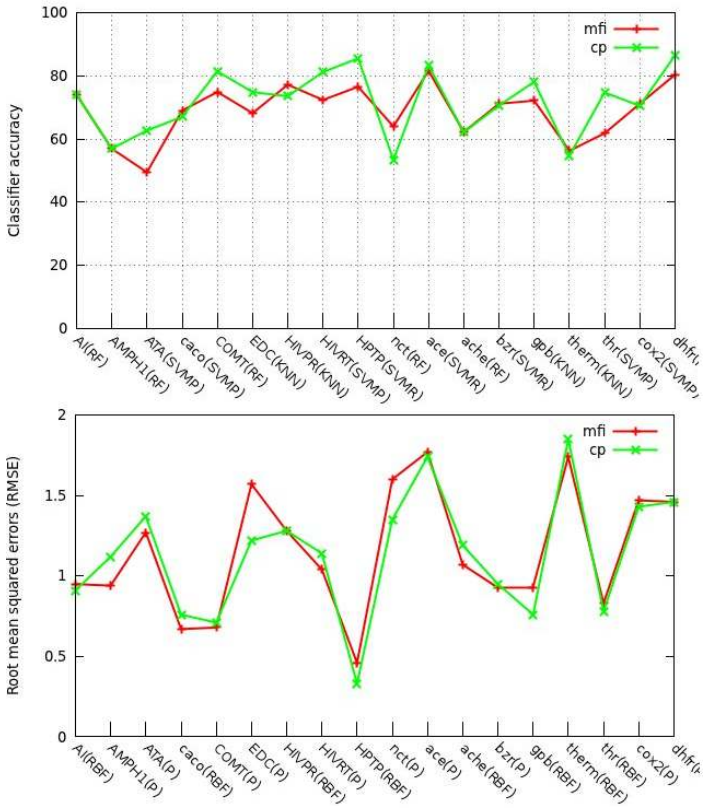

Figure 4.5: Comparison of classification (top) and regression (bottom) models of CP and *mfi*.

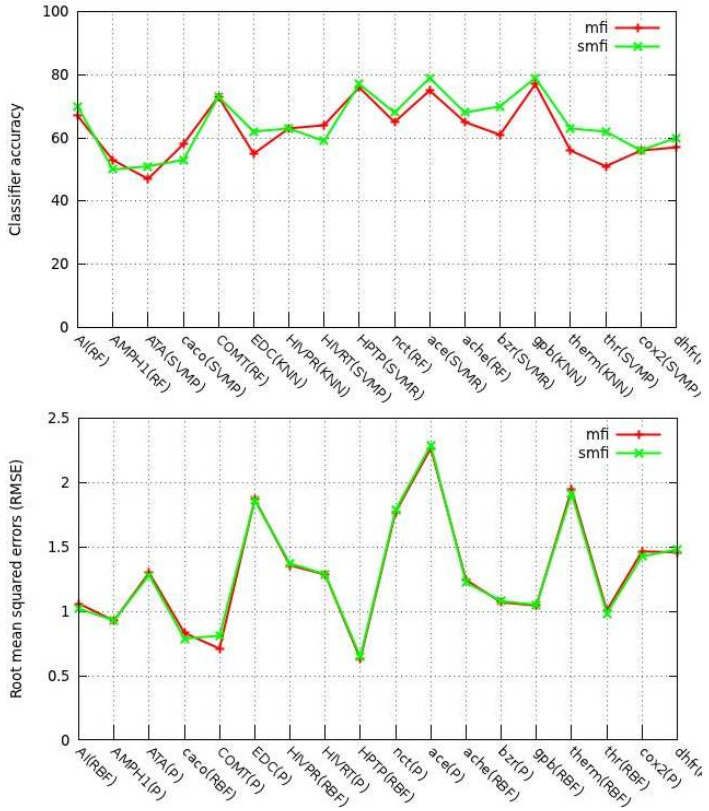

Figure 4.6: Comparison of classification (top) and regression (bottom) models of *Smfi* and *mfi*.

4.3.2 PERFORMANCE COMPARISONS OF EDGE LISTS AND EDGE SETS

Representing graphs as edge lists is less complex than representing them as edge sets, since an edge set requires the lexicographical order of the items. But the edge set contains more information than an edge list, as discussed in Chapter 3. The two graphs in Figure 4.7 show the performance levels of the method *mfi* when edge lists and edge sets are used. The models using an edge set provide a better performance for the majority of the datasets, having only one loss for classification models and no losses for regression models. The additional information in the edge set may be the reason for this enhanced performance. Figure 4.8 shows the same for models of CP. From these results it can be concluded that the models built using edge sets perform better than those with edge lists. In conclusion, *mfi* in conjunction with an edge set appears to be the best approach among the other proposed methods for efficient and effective learning from graph data.

4.3.3 SIZE OF THE FEATURE SET OF *mfi*

As stated in Chapter 1, the size of the feature set of frequency based methods can be very large when the graph size increases and/or the minimum support σ decreases. An analysis of the pattern set size of the proposed method *mfi* was therefore carried out. The results for some medicinal chemistry datasets, picked from Table 2.1, are presented in Figure 4.9. The sizes of the *mfi* of the datasets are small even for low values of σ, so that they could be effectively used in any learning algorithm.

Large graph databases

Figure 4.10 shows the pattern set sizes of *mfi* for some randomly picked datasets from Table 2.2. The curve corresponding to *Average* in Figure 4.10 represents the average of the sizes of the pattern sets discovered for the 60 datasets given in Table 2.2, with respect to the given support (σ). The variation of the sizes of the *mfi*'s are almost the same for all the datasets and are around 4500 for low support values such as 0.01 (which is not large for

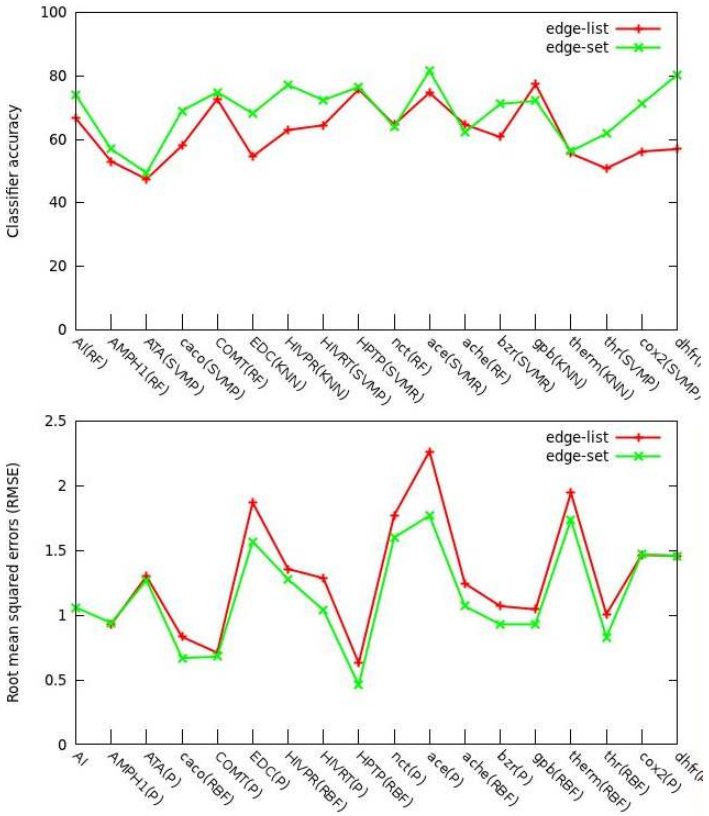

Figure 4.7: Comparison of models of *mfi* that use edge lists and edge sets.

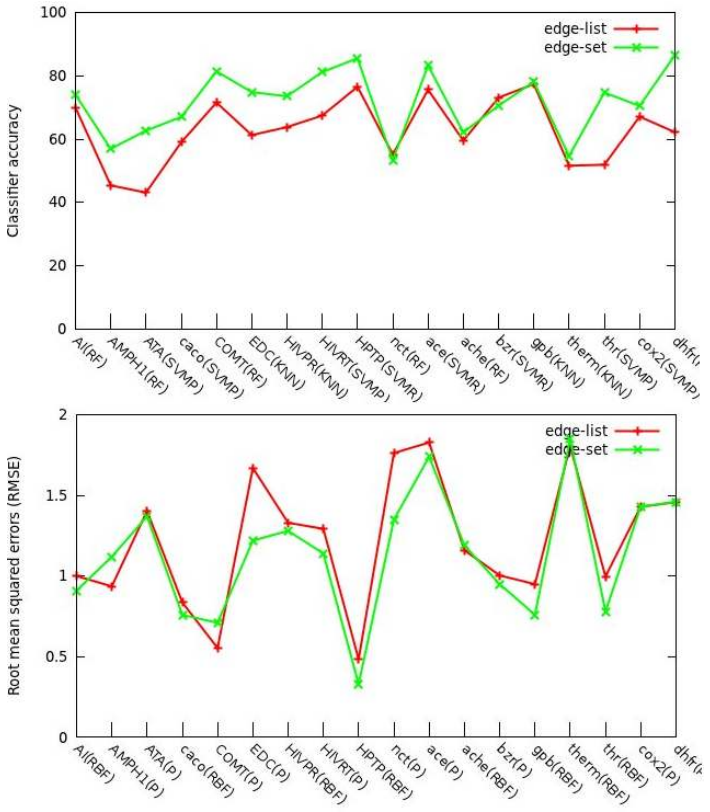

Figure 4.8: Comparison of models of CP that use edge lists and edge sets.

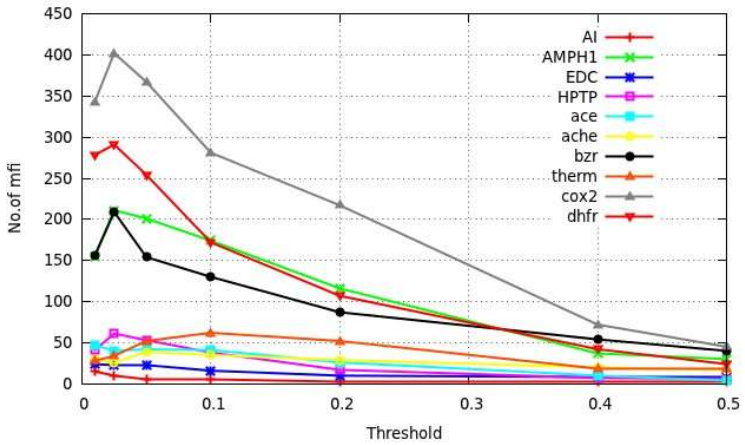

Figure 4.9: No. of *mfi* vs. threshold (support σ) for selected datasets from medicinal chemistry.

machine learning algorithms), and hence shows that *mfi* does not explode its pattern set for low support values when the dataset is large.

Datasets with large graphs

Figure 4.11 shows the size of the pattern sets discovered by *mfi* for the synthetic datasets g, h and I, with different graph sizes from 10 graphs to 10,000 graphs per dataset. Accordingly, the number of *mfi*'s generated for large graph databases such as $i10000$, are less than 9000. The execution times for the generation of the *mfi*'s for the datasets g, h and i are given in Figure 4.12. The execution times increase up to 7000 *ms* when the support value is decreased down to 0.01, but this becomes (almost) steady when the support is increased beyond 0.025. These results show that the efficiency of the method *mfi* does not get affected by reduced support σ, and/or increased database and graph size.

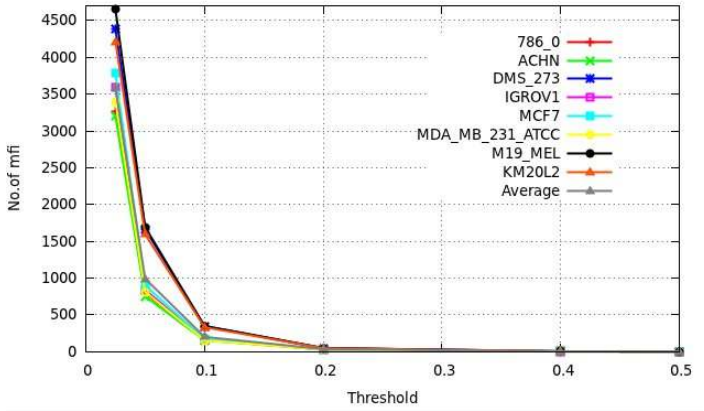

Figure 4.10: No. of *mfi* vs. threshold (support σ) of selected datasets from NCI repository.

mfi vs. database size

In this analysis we summarize the above results to establish any relation of the variation of the size of pattern sets with the support values. Figure 4.13 (top) shows the number of *mfi*'s vs. the size of the graph database. Figure 4.13 (bottom) shows the logarithmic values of *mfi*, which clearly shows the pattern of growth of *mfi*'s when the database size increases. The *mfi*'s corresponding to 5,000 and 10,000 are for the dataset g, and, as can be seen from the figure, are similar. This may be due to the similarity of the generated graphs when the graph size is increased to higher values.

Accordingly, we can see that the highest number of discovered *mfi*'s is approximately equal to the size of the dataset, except for $g1000$ that produce about 4500 patterns with 1000 graphs. These results show, contrary to the claim that the maximal set of the frequent patterns may not always be applicable in learning tasks (Hasan and Zaki, 2009), but that the *mfi* algorithm produce pattern sets that can still be applied for learning from datasets containing a set of (small) graphs.

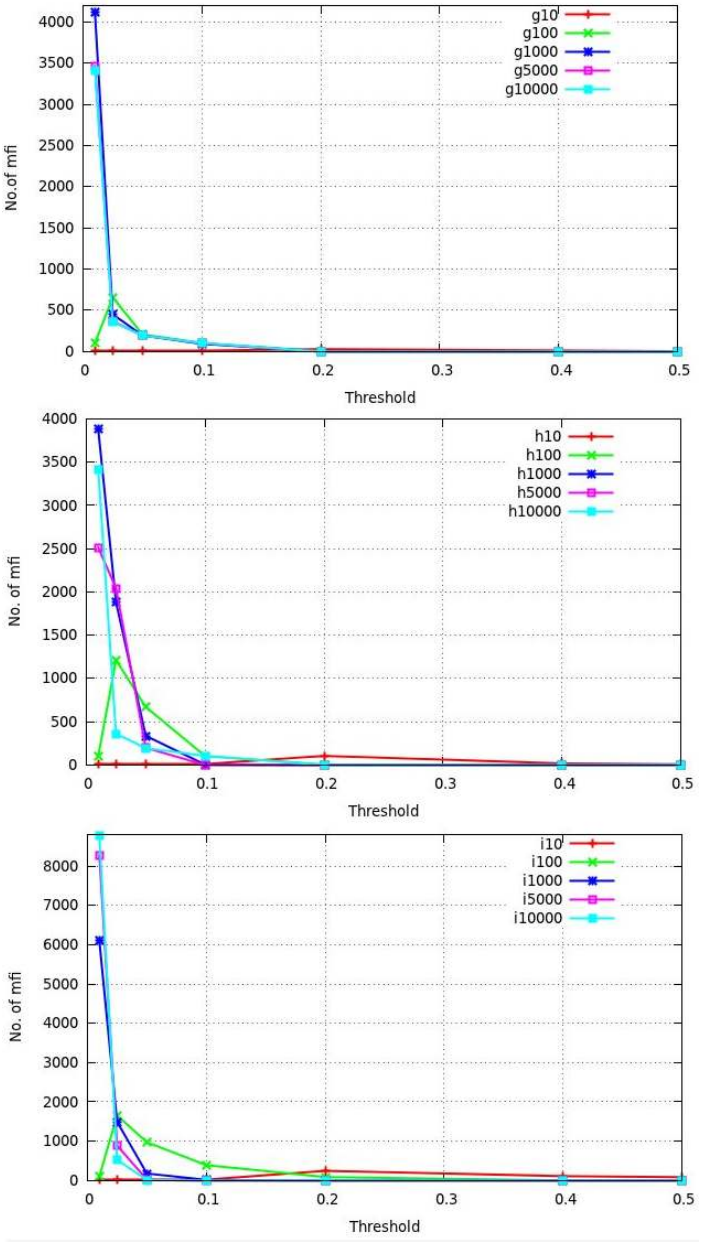

Figure 4.11: No. of maximal frequent items vs. threshold (support σ) for the synthetic graphs *g*, *h* and *i*.

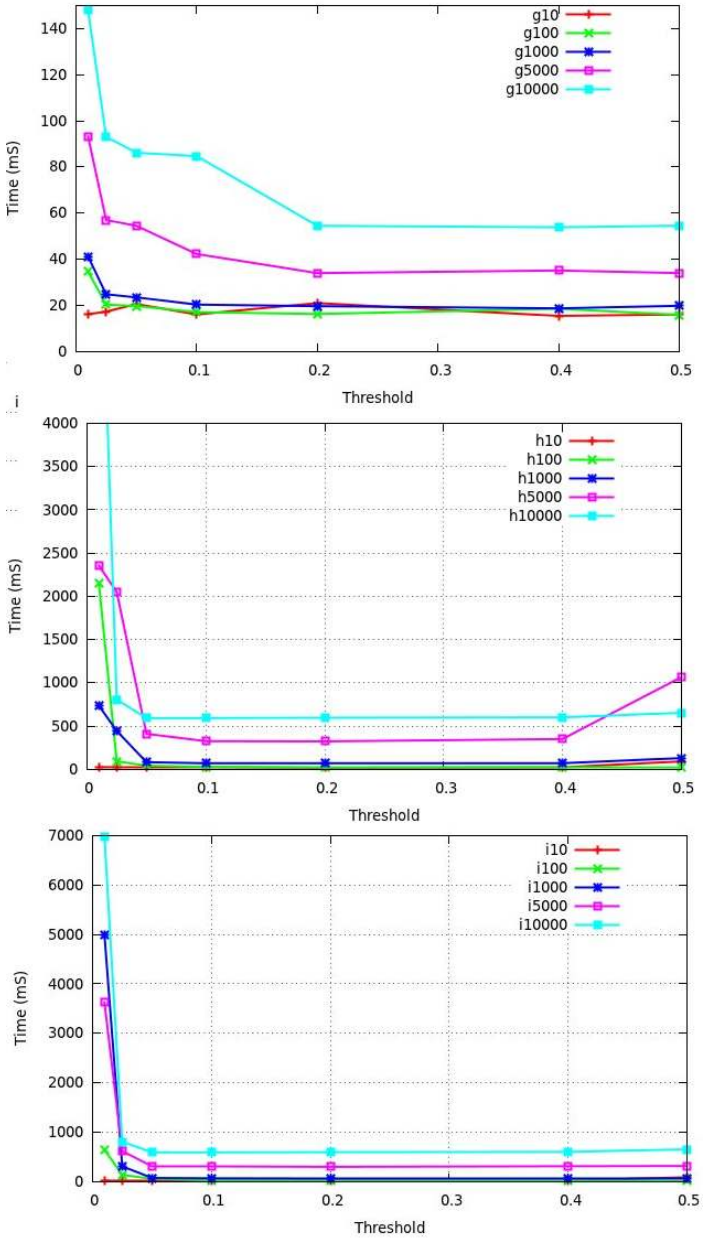

Figure 4.12: Execution times vs. threshold (support σ) for the synthetic graphs *g*, *h* and *i*.

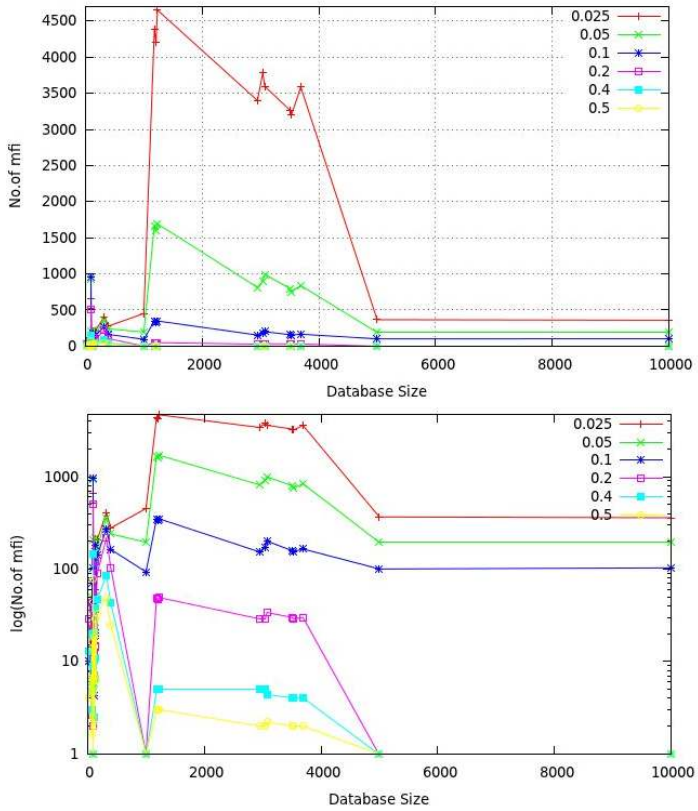

Figure 4.13: Database size vs. No of *mfi* (top) and log(No of *mfi*) (bottom).

4.4 SUMMARY

The results obtained from several experiments were discussed in this chapter. We have shown using various datasets that the methods *mfi* and CP are efficient compared to graph mining methods. Based on the predictive performance of the models built using the feature sets discovered by various pattern mining methods, it is concluded that the graph representation method edge sets in conjunction with the itemset mining methods resulted in feature sets that build predictive models efficiently without deteriorating the predictive performance. Inclusion of relevant background knowledge was shown to be useful in enhancing the predictive performance of any method, independently of the form in which they were used. We have also shown that some approaches such as using *mfi* and CP with edge sets are more successful than the other representations. The conclusions arrived at using these results will be compiled in the next chapter in order to examine whether we addressed the research question adequately.

CHAPTER 5

CONCLUDING REMARKS

The core of this thesis is the investigation of how to improve the efficiency and effectiveness of methods for learning predictive models from graph data using pattern mining. Due to the vast differences in the approaches taken to learn from graphs, we have selected a subset of methods to investigate, namely, learning from a set of small graphs using pattern mining methods. Several studies were conducted, each of which partially contributes to answering the research question. The conclusions drawn, based on the results of these studies, and a discussion of the future studies to which this research could be extended, are included in this chapter.

5.1 CONTRIBUTIONS

This thesis is based on nine publications describing the methods we proposed and evaluated for efficient and effective learning from graph data using pattern mining techniques. In Chapter 1, we pointed out that graph data are ubiquitous, but learning from graphs is a computationally expensive process, mainly due to the complexity of their representational form. On the other hand, due to the richness in the representation, predictive models built using graph data perform better than the models using other representational forms such as trees, sequences or itemsets. The majority

of the related approaches of learning from graphs take steps to improve the efficiency of the learning process by applying various constraints to the search space. Transforming the graphs into forms which are less complex to handle also reduce the cost of the mining process. Such steps however create different efficiency vs. effectiveness tradeoffs, turning the question of learning efficient and effective models, into an optimization problem with two objectives, where better solutions lie on the Pareto frontier of the solutions space. In such a situation, a method that has a higher efficiency and competitive predictive performance, or vice versa, constitutes a Pareto improvement (and thereby becomes a better method than the compared methods). However, which of the pattern mining methods used in feature construction to a propositional learner would result in Pareto improvements in efficiency vs. effectiveness was an open question that we addressed in this thesis. The efficiency of graph mining may be improved by using a less complex representational form than graphs. If such an approach results in models competitive in predictive performance, such a tradeoff would be near Pareto front. Further, the effectiveness of predictive models can be improved by the use of background knowledge, but, as pointed out in Chapter 1, the efficiency of the pattern mining methods decreases when the background knowledge is included in the graphs. However, it may be possible to incorporate background knowledge without increasing the complexity of the pattern mining process. The research presented in this thesis aimed at shedding some light along this line of research.

Our approach to answering the research question of efficient an effective learning from graphs is illustrated in Figure 2.1, where the graph representation and pattern mining methods are the contributions. Two graph representations named *edge list* and *edge set* have been used to transform graphs into forms that can be recognized by the pattern mining methods, maximal frequent itemset (*mfi*), maximal common substructures (*mcs*), and constraint programming based itemset mining (CP) and their variants defined in Section 3.5. Edge list and edge set are contributions of the thesis, whereas *mfi* and CP are itemset mining methods which are used

to discover patterns from graph data. The method *mcs* (and the system DIFFER) is also a new contribution.

In accomplishing the objective of improving the efficiency without depreciating the predictive performance, several graph encoding methods have been compared. The method *mcs* showed improved results compared to the state-of-the-art methods. However, the maximum itemset mining based method *mfi* has shown better performance than *mcs*. The methods *mfi* and CP demonstrated higher predictive performances than the comparison methods in many experiments. Further, both *mfi* and CP consumed only a few seconds for the discovery of their pattern sets for all the datasets, while the comparison methods required more time. Some of them did not complete the discovery of the pattern set within the stipulated time for some datasets. These outcomes lead to the conclusion that the proposed itemset mining methods speed up the learning process without decreasing the predictive performance. Hence, by representing graphs in the form of itemsets, which are computationally less expensive to handle than graphs, Pareto improvements in efficiency and effectiveness can be obtained, compared to the standard methods. Albeit the methods involving frequency are penalized by an unpredictable (and huge) size of the pattern set when the support values are low, we have shown using several datasets with different graph properties that the size of the pattern set of *mfi* for small graph datasets does not exceed the size of the dataset most of the time.

Ways of improving the predictive performance without increasing the complexity of the process of graph mining have also been examined along several lines. We have proposed two approaches to encoding background knowledge into graphs by 1) incorporating them into the graphs as a part of the node and edge definitions and 2) using them as additional features to the learning algorithm. The gain achieved by adding background knowledge into the node labels of graphs is significant. The amount of information contained in the node and edge labels is more important for better accuracy, than the form of representing the information inside the

node and edge labels. For example, in the chemoinformatics domain, 'two carbon atoms are connected by a bond labeled 7' is represented in two forms as given in Figure 2.1, but both encodings produced similar accuracy.

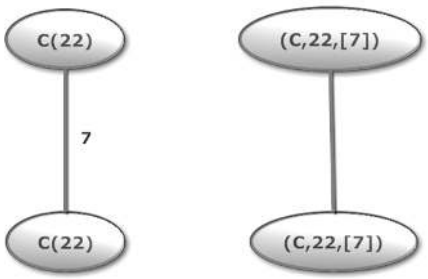

Figure 5.1: Two ways of incorporating the same amount of background knowledge into node labels–*Paper IV*.

The predictive performance of 75% of the regression models and 78% of classification models was improved by introducing domain specific background knowledge as additional features to the pattern sets of the graph and itemset mining methods. Also, the experiments using pattern sets from graph and itemset mining methods themselves as background knowledge concluded that 74% of the classification and 82% of the regression tasks on average had improved predictive performance when the feature set from itemset mining methods (*mfi* and CP) were used as background knowledge to the graph mining methods. The corresponding percentages, when graph mining methods were added to the proposed itemset mining methods, are 51% (classification) and 61% (regression).

These summaries of the outcomes of the experiments lead to the conclusion that the performance of the predictive models could be further improved by using background knowledge either in the form of extra features or as parts of the node and edge labels. In doing so, the size of the graph is not increased, thereby the complexity of mining algorithm is not increased,

and, therefore, any improvement to predictive performance is a Pareto improvement. Pareto improvements on pattern mining methods are more likely when the domain specific background knowledge is used as extra features than those from other pattern mining methods.

The empirical evaluations presented in this thesis lead to conclusions in favour of the proposed methods. However, we cannot rule out that not detecting any decreased performance might simply be due to using only a few datasets in the evaluation, and that the outcome of the empirical investigation could have been different if more datasets had been used.

5.2 FURTHER WORK

The presented experiments are focused on graph data in the form of a set of small graphs. An immediate question is whether or not the conclusions from this research carry over to other domains as well. Graphs such as social networks, web, and community networks, web link structures etc., contain huge structures of hundreds of thousands of nodes and even larger numbers of connections (edges) between the nodes, where nodes (and edges) are mostly labeled and the labels are almost unique. The present works could be extended to such domains. Similar studies could also be carried out for particular types of chemical compound datasets, e.g., very large or imbalanced datasets.

In optimization problems, finding the Pareto front is not an easy task (Konak et al., 2006). From our experiments we can only distinguish the methods that show Pareto improvements. Without conducting further experiments with many different approaches, concluding anything related to Pareto optimality is impossible. Even the proposed methods in this research could be subjected to improvements in their effectiveness and/or efficiency. For example, the predictive performance of the methods *mcs* and CP could be improved in many ways. The *mcs* algorithm could

possibly be extended to a graph kernel since this algorithm searches for common fragments in the edge sets of two graphs, which can be viewed as the similarity between the two graphs. This would be worthwhile to explore since *mcs* does not involve the graph isomorphism test during kernel computation (hence it is more efficient than graph kernels). The method CP could be further examined using the option of obtaining the top-k feature set, since the experiments we carried out show that the feature set of CP is comparatively large for some datasets. Further, the experiments involving CP could be extended with several different constraints such as information gain, the Gini index, convex hull, Fisher score, etc., to investigate any possibility of creating Pareto improvements in efficiency vs. effectiveness. More accurate predictive models could be obtained by further improving the feature selection in the itemset mining methods. The measures we used for feature selection for *mcs*, CP and *mfi* are currently somewhat naïve and straightforward. For *mcs*, the support values are purely user defined. The support values we used for itemset mining were selected from a limited set of discrete values that are not certainly optimized. It may be worthwhile to define a more compact and representative measure for the selection of features since selecting the most important set of features is the key factor for a better predictive performance. Using some coverage measure or a significance test, or a voting scheme using the ROC convex hull, might be promising approaches. The class labels of the training data could also be taken into account for a better measure for coverage or significance, as most of the existing methods does.

Alternative ways of utilizing background knowledge, such as, using the features of the domain specific method ECFI (which correspond to subgraphs), as a basis for compressing data, similar to replacing the molecular ring structures in MoFa (Borgelt, 2002), or replacing the subgraph fragments proposed by Cook and Holder (1994), could be investigated since such node labeling provide more structural information in the label, resulting in more accurate models.

REFERENCES

Charu Aggarwal and Haixun Wang. Graph data management and mining: A survey of algorithms and applications. In Charu Aggarwal and Haixun Wang, editors, *Managing and Mining Graph Data*, volume 40 of *Advances in Database Systems*, pages 13–68. Springer US, 2010.

Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

David W. Aha, Dennis Kibler, and Marc K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.

Ethem Alpaydin. *Introduction To Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, USA, second edition, 2009.

K. Bache and M. Lichman. UCI Machine Learning Repository, 2013.

Norman Blaikie. *Designing Social Research: The Logic of Anticipation*. Wiley, 2000.

Christian Borgelt. Mining molecular fragments: Finding relevant substructures of molecules. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM)*, pages 51–58. IEEE Press, 2002.

Karsten M. Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Proceedings of the 5th IEEE International Conference on Data Mining*, ICDM '05, pages 74–81, Washington, DC, USA, 2005. IEEE Computer Society.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Björn Bringmann. *Mining Patterns in Structured data*. PhD thesis, Katholieke Universiteit Leuven, Belgium, 2009.

Björn Bringmann and Albrecht Zimmermann. $tree^2\chi^2$ - decision trees for tree structured data. In AlipioMario Jorge, Luis Torgo, Pavel Brazdil, Rui Camacho, and Joao Gama, editors, *Knowledge Discovery in Databases: PKDD 2005*, volume 3721 of *Lecture Notes in Computer Science*, pages 46–58. Springer Berlin Heidelberg, 2005.

Frank Brown. Editorial opinion: Chemoinformatics - a ten year update. *Current Opinion in Drug Discovery and Development*, 8:298–302, 2005.

Douglas Burdick, Manuel Calimlim, and Johannes Gehrke. Mafia: A maximal frequent itemset algorithm for transactional databases. In *Proceedings of the 17th International Conference on Data Engineering*, pages 443–452. IEEE, 2001.

Wilfred Carr. Philosophy, methodology and action research. *Journal of Philosophy of Education*, 40:421–435, 2006.

ChemDB. Repository of Bren School of Information and Computer Science, University of California, Irvine, 2008. URL `ftp://ftp.ics.uci.edu/pub/baldig/learning/`. accessed 17-June-2011.

James Cheng, Yiping Ke, and Wilfred Ng. Graphgen: A synthetic graph generator, 2006. URL `http://www.cse.ust.hk/graphgen/`. accessed 04-October-2011.

Deborah Cohen and Benjamin Crabtree. Qualitative research guidelines project, 2006. URL `http://www.qualres.org/HomePosi-3515.html`. accessed 28-February-2012.

Diane J. Cook and Lawrence B. Holder. Substructure discovery using minimum description length and background knowledge. *Journal of Artificial Intelligence Research*, 1:231–255, 1994.

Diane J Cook and Lawrence B Holder. *Mining Graph Data*. John Wiley & Sons, USA, 2006.

Michael Crotty. *The Foundations of Social Research: Meaning and Perspective in the Research Process*. Sage Publications, London, 1998.

David Danks. Learning. *To appear in: K. Frankish & W. Ramsey (Eds.), Cambridge Handbook to Artificial Intelligence*.

Luc De Raedt and Stefan Kramer. The levelwise version space algorithm and its application to molecular fragment finding. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, volume 2 of *IJCAI'01*, pages 853–859, San Francisco, USA, 2001. Morgan Kaufmann Publishers Inc.

Asim Kumar Debnath, Rosa L. Lopez de Compadre, Gargi Debnath, Alan J. Shusterman, and Corwin Hansch. Structure activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital

energies and hydrophobicity. *Journal of Medicinal Chemistry*, 34:786–797, 1991.

Janez Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

Mukund Deshpande, Michihiro Kuramochi, and George Karypis. Frequent substructure-based approaches for classifying chemical compounds. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, pages 35–42, 2003.

Mukund Deshpande, Michihiro Kuramochi, Nikil Wale, and George Karypis. Frequent substructure-based approaches for classifying chemical compounds. *IEEE Trans. on Knowl. and Data Eng.*, 17(8):1036–1050, 2005.

Thomas G. Dietterich, Pedro Domingos, Lise Getoor, Stephen Muggleton, and Prasad Tadepalli. Structured machine learning: the next ten years. *Machine Learning*, 73(1):3–23, 2008.

Sean Ekins, Jordi Mestres, and Bernard Testa. In silico pharmacology for drug discovery: methods for virtual ligand screening and profiling. *British Journal of Pharmacology*, 152:9–20, 2007.

Peter Flach. The geometry of ROC space: Understanding machine learning metrics through ROC isometrics. In *Proceedings of the 20th International Conference on Machine Learning*, pages 194–201. AAAI Press, 2003.

Linton Freeman. *The Development of Social Network Analysis: A Study in the Sociology of Science*. BookSurge LLC, USA, 2004.

Robert D. Galliers. Choosing appropriate information systems research approaches: A revised taxonomy. In *Information Systems Research: Issues, Methods and Practical Guidelines*, pages 144–162, Oxford, U.K, 1991. Blackwell.

Salvador Garcia and Francisco Herrera. An extension on - statistical comparisons of classifiers over multiple data sets - for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008.

Thomas Gärtner. A survey of kernels for structured data. *ACM SIGKDD Explorations Newsletter*, 5(1):49–58, 2003.

Jesus A. Gonzalez, Lawrence B. Holder, and Diane J. Cook. Experimental comparison of graph-based relational concept learning with inductive logic programming systems. In *Proceedings of the 12th International Conference on Inductive Logic Programming*, ILP'02, pages 84–100, Berlin, Heidelberg, 2003. Springer.

Egon Guba. *The Paradigm Dialog*. Theory, Culture and Society. SAGE Publications, 1990.

Tias Guns, Siegfried Nijssen, and Luc De Raedt. Itemset mining: A constraint programming perspective. *Artif. Intell.*, 175(12-13):1951–1983, 2011.

Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques (The Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, 1st edition, 2000.

Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. Frequent pattern mining: Current status and future directions. *Data Min. Knowl. Discov.*, 15(1):55–86, 2007.

Mohammad Al Hasan and Mohammed Javeed Zaki. MUSK: Uniform sampling of k maximal patterns. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 650–661, 2009.

Christoph Helma, Stefan Kramer, and Luc De Raedt. The molecular feature miner MOLFEA. In *Proceedings of the Beilstein Workshop 2002: Molecular Informatics: Confronting Complexity*, 2003.

Alan Hevner and Samir Chatterjee. *Design Research in Information Systems: Theory and Practice*. Springer, USA, 2010.

Alan R Hevner, Salvatore T March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS quarterly*, 28(1):75–105, 2004.

Klaus Hinkelmann and Oscar Kempthorne. *Introduction to Experimental Design*. Design and Analysis of Experiments. Wiley, 2007.

Petteri Hintsanen and Hannu Toivonen. Finding reliable subgraphs from large probabilistic graphs. *Data Min. Knowl. Discov.*, 17(1):3–23, 2008.

Tamás Horváth, Thomas Gärtner, and Stefan Wrobel. Cyclic pattern kernels for predictive graph mining. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 158–167, New York, USA, 2004. ACM.

Jun Huan, Wei Wang, and Jan Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. In *Proceedings of the Third IEEE International Conference on Data Mining*, ICDM '03, pages 549–552, Washington, DC, USA, 2003. IEEE Computer Society.

Rob J. Hyndman and Anne B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688, 2006.

Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. An Apriori-based algorithm for mining frequent substructures from graph data. In *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 13–23, UK, 2000. Springer.

Chuntao Jiang, Frans Coenen, and Michele Zito. Frequent sub-graph mining on edge weighted graphs. In *Proceedings of the 12th International Conference on Data Warehousing and Knowledge Discovery*, pages 77–88, Berlin, Heidelberg, 2010. Springer.

Ning Jin, Calvin Young, and Wei Wang. GAIA: Graph classification using evolutionary computation. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, pages 879–890, New York, USA, 2010. ACM.

Thashmee Karunaratne, Henrik Boström, and Ulf Norinder. Comparative analysis of the use of chemoinformatics-based and substructure-based descriptors for quantitative structure-activity relationship (QSAR) modeling. *Intelligent Data Analysis*, 17(2):327–341, 2013.

Yiping Ke, James Cheng, and Jeffrey Xu Yu. Top-k correlative graph mining. *9th SIAM International Conference on Data Mining*, 2(4):150–163, 2009.

Nikhil S. Ketkar, Lawrence B. Holder, and Diane J. Cook. SUBDUE: Compression-based frequent pattern discovery in graph data. In *Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations*, OSDM '05, pages 71–76, New York, USA, 2005. ACM.

Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1137–1143. Morgan Kaufmann, 1995.

Abdullah Konak, David W. Coit, and Alice E. Smith. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering & System Safety*, 91(9):992–1007, 2006.

Mark A. Krogel, Simon Rawles, Filip Železný, Peter A. Flach, Nada Lavrac, and Stefan Wrobel. Comparative evaluation of approaches to propositionalization.

In *13th International Conference on Inductive Logic Programming*, pages 197–214, 2003.

Thomas Kuhn. *The Structure of Scientific Revolutions*. University of Chicago Press, Chicago, USA, 1970.

Michihiro Kuramochi and George Karypis. Frequent subgraph discovery. In *Proceedings of the 1st IEEE International Conference on Data Mining*, ICDM '01, pages 313–320, Washington, DC, USA, 2001. IEEE Computer Society.

Niels Landwehr. Trading expressivity for efficiency in statistical relational learning. Katholieke Universiteit Leuven, 2009.

Nada Lavrač, Filip Zelezny, and Peter Flach. RSD: Relational subgroup discovery through first-order feature construction. In *12th International Conference on Inductive Logic Programming*, pages 149–165. Springer, 2002.

George F. Luger. *Artificial Intelligence, Structures and Strategies for Complex Problem Solving*. Addison Wesley, USA, 4th edition, 2002.

John McCarthy. Challenges to machine learning: Relations between reality and appearance. In Stephen Muggleton, Ramon Otero, and Alireza Tamaddoni-Nezhad, editors, *Proceedings of the 17th International Conference on Inductive Logic Programming*, volume 4455, pages 2–9. Springer, 2007.

Donald Michie, Stephan Muggleton, David page, and Ashvin Srinivasan. To the international computing community: A new East-West challenge. Technical report, Oxford University Computing laboratory, Oxford, UK, 1994.

Tom Michael Mitchell. *The Discipline of Machine Learning*. Carnegie Mellon University, School of Computer Science, Machine Learning Department, 2006.

Ruchi R. Mittal, Ross A. McKinnon, and Michael J. Sorich. Comparison data sets for benchmarking QSAR methodologies in lead optimization. *Journal of Chemical Information and Modeling*, 49(7):1810–1820, 2009.

David Morgan. Paradigms lost and pragmatism regained: Methodological implications of combining qualitative and quantitative methods. *Journal of Mixed Methods Research*, 1(1):48–76, 2007.

Stephen Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295–318, 1991.

Stephen Muggleton. Inverse entailment and progol. *New generation computing*, 13(3-4):245–286, 1995.

Stephen Muggleton and Wray L. Buntine. Machineinvention of first order predicates by inverting resolution. In *Proceedings of the 5th International Conference on Machine Learning*, pages 339–352. Morgan Kaufman, 1988.

Stephen Muggleton and Cao Feng. Efficient induction of logic programs. In *Inductive Logic Programming*, volume 38, pages 281–298. Academic Press, 1992.

Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *The Jorunal of Logic Programming*, 19(20):629–679, 1994.

Claire Nédellec, Céline Rouveirol, Hilde Adé, Francesco Bergadano, and Birgit Tausend. Declarative bias in inductive logic programming. *Advances in Inductive Logic Programming, Frontiers in Artificial Intelligence and Applications*, pages 82 –103, 1996.

Marc Nicklaus. Downloadable structure files of NCI open database compounds, 1996. URL `http://cactus.nci.nih.gov/download/nci/`.

Siegfried Nijssen and Joost N. Kok. A quick start in frequent structure mining can make a difference. In *Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 647–652, New York, USA, 2004. ACM.

Siegfried Nijssen, Tias Guns, and Luc De Raedt. Correlated itemset mining in ROC space: a constraint programming approach. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 647–656, New York, USA, 2009. ACM.

T Olsson and V Sherbukhin. Selma, synthesis and structure administration (SaSA). AstraZeneca R & D Mölndal, Sweden, 1999.

Zanifa Omary and Fredrick Mtenzi. Machine learning approach to identifying the dataset threshold for the performance estimators in supervised learning. In *International Conference for Internet Technology and Secured Transactions*, pages 1–8, 2009.

Oxford. "research". oxford dictionaries, 2010. URL `http://oxforddictionaries.com`. accessed 28-February-2012.

Marcello Pelillo, Joachim Buhmann, Tiberio Caetano, Bernhard Schölkopf, and Larry Wasserman. Philosophy and Machine Learning. In *Workshop Book of the 25th Neural Information Processing Systems*, NIPS'11, pages 27–29, Spain, 2011. Neural Information Processing System Foundation, Inc.

John C. Platt. Fast training of support vector machines using sequential minimal optimization. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods*, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

Liva Ralaivola, Sanjay J. Swamidass, Hiroto Saigo, and Pierre Baldi. Graph kernels for chemical informatics. *Neural Netw.*, 18(8):1093–1110, October 2005.

Sayan Ranu and Ambuj K. Singh. GraphSig: A scalable approach to mining significant subgraphs in large graph databases. *25th International Conference on Data Engineering*, pages 844–855, 2009.

David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754, 2010.

Lorenza Saitta and Michèle Sebag. Phase transitions in machine learning. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning*, pages 767–773. Springer, 2010.

Ashvin Srinivasan. The Aleph manual, 2004. URL `http://www.cs.ox.ac.uk/activities/machlearn/Aleph/aleph.html`. accessed 24-February-2012.

Ashwin Srinivasan, Ross D. King, Stephen Muggleton, and Michael J. E. Sternberg. Carcinogenesis predictions using ILP. In *Proceedings of the Seventh International Conference on Inductive Logic Programming*, pages 273–287. Springer, 1997.

Ashwin Srinivasan, Ross D. King, and Stephen H. Muggleton. The role of background knowledge: Using a problem from chemistry to examine the performance of an ILP program. Technical Report PRG-TR-08-99, Oxford University Computing Laboratory, Oxford, 1999.

Marisa Thoma, Hong Cheng, Arthur Gretton, Jiawei Han, Hans Peter Kriegel, Alex Smola, Le Song, Philip S. Yu, Xifeng Yan, and Karsten Borgwardt. Near-optimal supervised feature selection among frequent subgraphs. In *Proceedings of the 9th SIAM International Conference on Data Mining*, pages 1075–1086, 2009.

Lini Thomas. *Maximal frequent subgraph mining*. PhD thesis, International Institute of Information Technology, Hyderabad, India, 2010.

Koji Tsuda and Hiroto Saigo. Graph classification. In Charu C. Aggarwal and Haixun Wang, editors, *Managing and Mining Graph Data*, volume 40 of *Advances in Database Systems*, pages 337–363. Springer, 2010.

Vladimir N Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999.

S. V. N. Vishwanathan, Nicol N. Schraudolph, Risi Kondor, and Karsten M. Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11:1201–1242, 2010.

Nikil Wale. Machine learning in drug discovery and development. *Drug Development Research*, 72(1):112–119, 2011.

Wei Wang and Jiong Yang. Mining high-dimensional data. In Oded Maimon and Lior Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, pages 793–799. Springer, 2005.

Takashi Washio and Hiroshi Motoda. State of the art of graph-based data mining. *SIGKDD Explor. Newsl.*, 5(1):59–68, 2003.

Xifeng Yan and Jiawei Han. gspan: Graph-based substructure pattern mining. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, ICDM '02, pages 721–724, Washington, DC, USA, 2002. IEEE Computer Society.

Xifeng Yan and Jiawei Han. CloseGraph: mining closed frequent graph patterns. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 286–295, New York, USA, 2003. ACM.

Xifeng Yan, X. Jasmine Zhou, and Jiawei Han. Mining closed relational graphs with connectivity constraints. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '05, pages 324–333, New York, USA, 2005. ACM.

Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca, and Viviane Grunert Da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7:117–132, 2003.