



TECHNISCHE UNIVERSITÄT MÜNCHEN

Fakultät für Informatik  
Lehrstuhl für Bioinformatik

# Structural Graph Clustering: Scalable Methods and Applications for Graph Classification and Regression

Dipl. Wirtsch.-Inf. Madeleine Seeland

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. Ernst W. Mayr

Prüfer der Dissertation:

1. Univ.-Prof. Dr. Burkhard Rost
2. Univ.-Prof. Dr. Stefan Kramer  
Johannes Gutenberg Universität Mainz

Die Dissertation wurde am 19.05.2014 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 04.08.2014 angenommen.



*In loving memory of my father, Alfred Seeland.*



---

## Abstract

---

In recent years, the collection of complex data from various domains, such as bio- and cheminformatics, has been rapidly increasing due to advances in technology for generating and collecting data. The explosive growth in data has generated an urgent need for new technologies and automated tools capable of efficiently extracting useful hidden information from these data. Data mining, therefore, has become a research area with increasing importance. As much of the data collected is structural in nature, graph-based representations have been intensively employed for modeling these data, making graph mining an important topic of research. This thesis focuses on the graph mining task of clustering objects in a graph database based on graph structure and studies several applications for classification and regression.

The first part of the thesis introduces scalable methods for clustering large databases of small graphs by common scaffolds, i.e., the existence of one sufficiently large subgraph shared by all cluster elements. In contrast to many related approaches, these methods are not based on computationally expensive maximum common subgraph operations or variants thereof, but on frequent subgraph mining. First, an incremental online approach for this task is presented that produces non-disjoint and non-exhaustive clusterings. A major challenge in this endeavor is the scalability of the approach to large data sets. Therefore, a parallelized structural graph clustering approach, called PSCG, is presented that takes advantage of high-performance parallel hardware and further employs several cluster exclusion criteria to reduce the number of expensive subgraph search computations. Subsequently, a scalable graph clustering approach, called SCAP (Structural Clustering by Abstract Pre-clustering), designed for the efficient clustering of very large graph databases containing millions of graphs is proposed.

The second part of this thesis investigates the applicability of graph clustering or, more generally speaking, local structural graph (similarity) neighborhoods for building models for classification and regression. The first approach, called LWL-MCS (Locally Weighted Linear Regression based on Maximum Common Subgraph), surveys locally weighted learning on graphs using a distance measure based on the maximum common subgraph to determine the neighborhood of a test instance. The second approach, called SCK (Structural Cluster Kernel), enables kernel methods to utilize additional information hidden in the structural neighborhood of the graphs under consideration. To this end, it exploits the clusters produced by PSCG to improve state-of-the-art graph kernels. Last, an approach

---

is presented that aims at extracting knowledge from support vector machines trained on the SCK to obtain more compact pattern-based classification models.

In extensive experiments, the effectiveness and efficiency of the proposed approaches have been proved on various real world data sets of molecular graphs. The results show that it is for the first time possible to cluster millions of graphs within a reasonable time using an accurate scaffold-based similarity measure. In the domain of cheminformatics, for instance, this represents a step towards structuring the vast chemical space. In addition, structural graph neighborhoods have proven their applicability in various applications for classification and regression.

The graph clustering approaches presented in this thesis pave the way for new research challenges in various areas such as bio- and cheminformatics and consequently have the potential to play a major role in domains involving the analysis of large volumes of structured data.

---

## Zusammenfassung

---

Die Menge komplexer Daten aus unterschiedlichen Bereichen wie der Bio- oder Chemieinformatik ist in den vergangenen Jahren infolge des technologischen Fortschritts bei der Datengenerierung und -sammlung stark angestiegen. Dieser explosionsartige Anstieg erzeugte einen großen Bedarf nach neuen Methoden zur effizienten Extrahierung von verborgenen Informationen aus großen Datenbeständen. Data Mining hat sich aus dieser Motivation heraus zu einem Forschungsgebiet mit zunehmender Bedeutung entwickelt. Da viele der gesammelten Daten in struktureller Form vorliegen, werden verstärkt graphbasierte Repräsentationen zur Modellierung der Daten eingesetzt, mit der Folge, dass sich Graph Mining zu einem wichtigen Forschungsthema entwickelt hat. Aufsetzend auf dieser Thematik dokumentiert diese Dissertation neue Ansätze zu einem Teilgebiet des Graph Minings, dem Graph Clustering. Das Ziel von Graph Clustering ist die Gruppierung von Objekten in einer Graphdatenbank basierend auf ihrer Graphstruktur. Weiterhin werden Anwendungen für die Klassifikation und Regression beschrieben.

Der erste Teil der Arbeit führt skalierbare Verfahren zum Clustern von großen Graphdatenbanken auf Basis eines den Clusterelementen gemeinsamen, ausreichend großen Subgraphen ein. Im Unterschied zu vielen verwandten Ansätzen basieren die Methoden nicht auf rechenintensiven maximalen gemeinsamen Subgraph Operationen oder Varianten davon, sondern auf dem Finden von häufigen Subgraphen. Zunächst wird ein inkrementelles online Verfahren vorgestellt, das überlappende (nicht-disjunkte) und nicht-vollständige Clusterings erzeugt. Eine große Herausforderung ist dabei die Skalierbarkeit auf großen Datensätzen. Zur Lösung dieser Herausforderung wurde ein parallelisierter struktureller Graph-Clustering-Ansatz namens PSCG entwickelt, der die Vorteile hochleistungsfähiger parallel arbeitender Rechenkerne nutzt und gleichzeitig Gebrauch von mehreren Ausschlusskriterien macht, um die Anzahl an teuren Berechnungen von häufigen Subgraphen zu reduzieren. Ergänzend wurde ein skalierbarer Graph Clustering Ansatz namens SCAP (Structural Clustering by Abstract Pre-clustering) entwickelt, der speziell für das effiziente Clustering von sehr großen Graphdatenbanken, die mehrere Millionen von Graphen enthalten, konzipiert wurde.

Der zweite Teil der Dissertation beschäftigt sich mit der Anwendbarkeit von Graph-Clustering-Verfahren oder, allgemeiner, von lokalen strukturellen Graphnachbarschaften zum Erlernen von Modellen zur Klassifikation und Regression. Der erste Ansatz namens LWL-MCS (Locally Weighted Linear Regression based on Maximum Common Subgraph)

---

erzeugt lokale Modelle zur Vorhersagezeit unter Verwendung eines Distanzmaßes basierend auf dem maximalen gemeinsamen Subgraphen zur Bestimmung und Gewichtung der Nachbarschaft einer Testinstanz. Der zweite Ansatz, SCK (Structural Cluster Kernel), erweitert Kern-Methoden, um Informationen nutzen zu können, die in strukturellen Graphnachbarschaften verborgen sind. Hierfür verwendet die Methode die aus PSCG gewonnenen Ähnlichkeiten, um die im Stand der Wissenschaft und Technik bekannten Graph Kernels zu verbessern. Zuletzt wird ein Ansatz präsentiert, der zum Ziel hat, Wissen aus (auf dem SCK) trainierten Support Vektor Maschinen Stützvektormaschinen zu extrahieren, um kompaktere musterbasierte Klassifikationsmodelle zu erhalten.

In umfangreichen Experimenten wurde die Effektivität und Effizienz der vorgestellten Ansätze auf zahlreichen molekularen Graphdatensätzen bewiesen. Die Ergebnisse zeigen, dass es zum ersten Mal möglich ist, Millionen von Graphen unter Verwendung eines akkuraten strukturbasierten Ähnlichkeitsmaßes innerhalb eines annehmbaren Zeitrahmens zu clustern. Auf dem Gebiet der Chemieinformatik stellt dies beispielsweise einen Schritt in Richtung Strukturierung des riesigen chemischen Strukturraums dar. Weiterhin wurde die Anwendbarkeit von strukturellen Graphnachbarschaften für verschiedenen Anwendungen zur Klassifikation und Regression gezeigt.

Die in dieser Dissertation vorgestellten Graph-Clustering-Ansätze ebnet den Weg für neue wissenschaftliche Herausforderungen in unterschiedlichen Gebieten wie beispielsweise die Bio- oder die Chemieinformatik und beinhalten somit das Potenzial, zukünftig in der Analyse von großen, strukturierten Datenmengen eine entscheidende Rolle zu spielen.



---

## Acknowledgements

---

Several individuals and institutions contributed in various ways to the completion of this dissertation. I am very thankful for their support and advice, and grateful for the unique chances this support offered me. Without their support, this work would not have been possible.

First of all, I want to express my sincere gratitude to my supervisor, Stefan Kramer, for giving me the opportunity to pursue my doctoral studies in his research group. Throughout my doctoral studies, he provided me with invaluable suggestions and comments, encouraged me to develop independent research skills, pushed me towards publishing results at highly reputable venues and provided all the assistance I needed to conduct my research. Thus, he significantly improved this thesis both regarding content and presentation. I am also grateful that Stefan gave me the possibility to finish my doctoral studies at the Technische Universität München when he moved to the Johannes Gutenberg-Universität Mainz. In this context, I would also like to thank Burkhard Rost for hosting me at the Technische Universität München and for acting as the first reviewer of this thesis.

During the past years, I had the chance to collaborate with many interesting people who provided valuable input on various parts of my research. In particular, I am very grateful to Bernhard Pfahringer for his support and valuable discussions. It has been a unique chance for me to work with him and to learn from his scientific experience. Special thanks go to Andreas Karwath, with whom I worked together on various research projects. In all of those he has made substantial contributions. Besides, he always had an open ear for questions and discussions. Further, I am very grateful to Tobias Girschick for his great support especially at the beginning of my doctoral studies and for numerous fruitful discussions. Thanks also to Simon Berger for helping me to implement the parallel version of the clustering approach and for valuable discussions regarding high performance computing related issues. I would also like to thank all my colleagues at the Technische Universität München who provided an excellent working atmosphere and significantly influenced my work and life. Special thanks go to our great system administrator, Timothy Karl, not only for his continuous technical support and assistance with the computational resources, but also for his constant positive attitude. I am also deeply grateful to Marlena Drabik for all her help regarding administrative issues and for integrating me so warmly into the lab. Likewise, I would like to thank my former office mates, Jana Schmidt and Andreas Hapfelmeier, for making my work place such a pleasant to be.

---

Last but not least, I want to thank the most important people in my life, my family, for their continuing support and believe in me. Thanks to my parents, Jutta Seeland and Alfred Seeland, for their love and support throughout my life. I dedicate this dissertation to the loving memory of my father, Alfred Seeland, who I miss dearly as I finish this part of my life in his absence. He would have been very proud of me.

The last credits I would like to dedicate to a very special person, my husband Jan Gumprecht, for his constant source of support and love in my life. He was always there for me, encouraging me and giving me the necessary strengths in difficult times. I would never have reached this point without his continuous support and love.

Finally, I acknowledge financial support of the German Bundesministerium für Bildung und Forschung (BMBF), under the project REACH (FKZ 0315546C).

---

# Contents

---

Acronyms	xiii
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Contributions . . . . .	4
1.3 Applications . . . . .	8
1.3.1 Definition of Chemical Categories . . . . .	8
1.3.2 Predictive Toxicology . . . . .	9
1.3.3 Virtual Screening . . . . .	9
1.4 Outline of the Thesis . . . . .	9
<b>2 Related Work</b>	<b>11</b>
2.1 Introduction to Graph Theory . . . . .	11
2.1.1 Graph . . . . .	11
2.1.2 Graph Isomorphism and Subgraph Isomorphism . . . . .	12
2.2 Graph Clustering . . . . .	15
2.2.1 Clustering Vectorial Data . . . . .	17
2.2.2 Graph-based Clustering . . . . .	19
2.3 Kernel Methods . . . . .	22
2.3.1 Kernels . . . . .	23
2.3.2 Graph Kernels . . . . .	26
2.3.3 Cluster Kernels . . . . .	30
2.3.4 Support Vector Machines . . . . .	31
2.4 Knowledge Extraction from SVMs . . . . .	37
2.4.1 Rule Extraction from SVMs . . . . .	38
2.4.2 Other Knowledge Extraction Methods from SVMs . . . . .	46
<b>3 Graph Clustering</b>	<b>49</b>
3.1 Structural Graph Clustering . . . . .	50
3.1.1 Problem Definition . . . . .	52
3.1.2 Method . . . . .	54
3.1.3 Experiments . . . . .	56

3.1.4	Conclusion . . . . .	62
3.2	Parallel Structural Graph Clustering . . . . .	64
3.2.1	Method . . . . .	64
3.2.2	Experimental Results . . . . .	75
3.2.3	Conclusion . . . . .	80
3.3	Structural Clustering by Abstract Pre-clustering . . . . .	81
3.3.1	Method . . . . .	82
3.3.2	Experimental Results . . . . .	90
3.3.3	Conclusion . . . . .	97
4	Maximum Common Subgraph Based Locally Weighted Regression . . . . .	99
4.1	Introduction . . . . .	99
4.2	Related Work . . . . .	100
4.3	Method . . . . .	102
4.3.1	Notation and Definitions . . . . .	102
4.3.2	MCS Algorithm . . . . .	102
4.3.3	MCS-based Distance Measure . . . . .	103
4.3.4	MCS-based Locally Weighted Regression . . . . .	104
4.4	Experimental Results . . . . .	108
4.5	Discussion and Conclusion . . . . .	115
5	The Structural Cluster Kernel . . . . .	117
5.1	Introduction . . . . .	117
5.2	Method . . . . .	118
5.2.1	Structural Cluster Kernel . . . . .	118
5.2.2	Semi-Supervised Setting . . . . .	121
5.3	Experimental Results . . . . .	122
5.3.1	Supervised Setting . . . . .	123
5.3.2	Semi-Supervised Setting . . . . .	127
5.3.3	Comparison to Locally Weighted Learning . . . . .	129
5.4	Conclusion . . . . .	132
6	Mining Support Vectors of the Structural Cluster Kernel . . . . .	133
6.1	Introduction . . . . .	133
6.2	Problem Definition . . . . .	135
6.3	Method . . . . .	136
6.3.1	Backbone Refinement Class Mining . . . . .	136
6.3.2	Graph Mining On Support Vectors . . . . .	138
6.4	Experiments . . . . .	141
6.4.1	Baseline Methods . . . . .	141
6.4.2	Data Sets . . . . .	142
6.4.3	Experimental Setup . . . . .	142

6.4.4 Results . . . . .	143
6.5 Conclusion . . . . .	149
7 Conclusion and Outlook	151
7.1 Conclusion . . . . .	152
7.2 Outlook . . . . .	155
List of Figures	159
List of Tables	165
List of Algorithms	167
Bibliography	169



---

## Acronyms

---

APreClus	Abstract Pre-Clustering
DySC	Dynamic Seed-Based Clustering
EM	Expectation Maximization
ICA	Independent Component Analysis
LoMoGraph	Local Models for Graph Classification and Regression
LWL	Locally Weighted Learning
LWL-MCS	Locally Weighted Linear Regression based on Maximum Common Subgraph
MCES	Maximum Common Edge Subgraph
MCS	Maximum Common Subgraph
NSPDK	Neighborhood Subgraph Pairwise Distance Kernel
PCA	Principal Component Analysis
PSCG	Parallel Structural Clustering of Graphs
QSAR	Quantitative Structure-Activity Relationship
SCAP	Structural Clustering by Abstract Pre-clustering
SCK	Structural Cluster Kernel
SOM	Self-Organizing Map
SVM	Support Vector Machine
WDK	Weighted Decomposition Kernel





# CHAPTER 1

---

## Introduction

---

### 1.1 Motivation

In recent years, the collection of data from various domains, such as bio- and cheminformatics, has been rapidly increasing due to improvements of existing technologies as well as the introduction of new technologies allowing to conduct many large scale experiments. For example, in the domain of cheminformatics there has been an explosion in the type and amount of data that is available for analysis due to the advent of large public repositories of chemical and biological data. An example for such big data sources is the PubChem database [176] that contains nearly 49.5 million chemical compounds. This vast amount of information generates new opportunities for extracting and understanding the underlying relationship between chemical structures and biological activity. The extraction of useful knowledge from these large amounts of data is only possible with the help of complex computational tools. In the domain of cheminformatics, this new insight may provide the potential for supporting the drug discovery process and the development of safer chemicals [29, 108, 150, 235, 242]. Hence, there is a pressing need for the development of techniques and tools capable of extracting useful hidden information from these data. With the explosive growth of data, the demand for analysis tools and techniques even increases.

Data mining, an important step in this process of knowledge discovery, provides methods that discover interesting, non-trivial, and useful patterns hidden in the data [5, 50, 98]. Most classical data mining approaches are dealing with data that is represented by itemsets or attribute-value encodings, where each instance is described by a set of properties. However, such fixed types of representation cannot always be used to model the data to be analyzed in an adequate manner. Many collections of data from various domains such as bioinformatics, cheminformatics, social network analysis etc., contain information that is structural in nature. This poses a critical problem to the traditional itemset and attribute-value oriented modeling approaches as they are unable to preserve the topological structure of the data in the representation.

This limitation has triggered the data mining research community to encourage mining and learning within alternative and more expressive representations such as sequences,

trees and graphs [90, 132, 137, 209, 251]. Among them, graphs – consisting of a set of nodes connected by edges – are one of the most general data structures in computer science, as sequences and trees are special cases of graphs. The incentive for using graph representations is that they provide a natural representation of structured data and are more expressive in comparison to flat representations. It makes them more broadly and extensively applicable. As a result, graphs have become increasingly important in modeling complicated structures, such as chemical compounds, protein structures, biological networks, social networks, the web, workflows, circuits, images, and XML documents, with broad applications in various areas including bio- and cheminformatics, image classification, web analysis and computer network analysis [8, 31, 32, 33, 70, 71, 100, 143, 149, 154, 152, 179, 196, 197].

With the increasing demand on the analysis of large amounts of structured data, graph mining has become an active and important topic in data mining [58, 113, 141, 161, 245, 249]. Graph mining aims at extracting useful knowledge from a large amount of structured data modeled as graphs. This discipline has become an important topic of research because of a wide variety of data mining problems in computational biology, chemical data analysis, drug discovery and communication networking. The problem of graph mining arises in two different contexts: mining large networks and mining large databases of small graphs. The first setting, mining large networks, considers data represented in one large, connected network [91, 140, 178]. In this domain, a network (graph) is composed of a large number vertices with distinct labels. Examples of such networks include the World Wide Web [37], social networks [74], citation networks [36], biological networks (e.g., protein interaction networks [11, 99, 110] and gene regulatory networks [155, 169]), and computer networks [76]. The second setting, mining of large databases of small graphs, considers large databases of small, separate, independent graphs, such as databases of molecules or databases of images. In this domain, the graphs are relatively small, but the labels on different nodes, which are drawn from a limited set of elements, may be repeated many times in a single graph. This thesis focuses on the latter graph mining setting, namely mining large databases of small graphs. In recent years, this topic has become a very active research area and several techniques have been designed covering the whole range of methods from data mining and machine learning. Among them are techniques that deal with problems such as frequent pattern mining, clustering and classification or regression [3, 30, 59, 109, 127, 139, 173, 245, 246, 244, 251].

Clustering is a fundamental task and one of the most studied topics in data mining [116, 117, 128, 243]. Given a set of data instances, the general goal of clustering is to group the instances into a set of well separated groups that share common characteristics based on similarity. Intuitively, instances within a cluster are more similar to each other than they are to an instance belonging to different clusters. Clustering algorithms are particularly suitable for the exploration of interrelationships among individual objects, i.e., they are mainly used as exploratory data analysis tool.

In recent years, clustering has received a lot of attention in the domain of graphs [4, 79, 181]. The problem of graph clustering has traditionally been studied in the context

of node clustering of individual graphs, in which one attempts to cluster the vertices of a given input graph into groups of densely connected vertices. This kind of clustering is sometimes also referred to as community detection [81, 195] and has traditionally been studied in the context of graph-partitioning [1, 126, 131], minimum-cut determination [125], network structure clustering [181] and dense subgraph determination [91, 252]. In recent years, the problem of clustering graphs has also been investigated in the context of clustering large databases of small graphs, where one attempts to cluster many different individual graphs (as objects) based on their structural similarity [3, 67]. This thesis focuses on the latter class of clustering problems, i.e., on clustering large databases of small graphs (also called graph-based clustering). In the following, the terms graph-based clustering and graph clustering are used interchangeably to refer to the problem of clustering data sets of individual graphs. Graph clustering techniques are very useful for detecting groups of structurally similar graphs while at the same time highlighting differences between dissimilar ones. As a result, clustering large databases of small graphs has emerged as a challenging research area with a large variety of applications, such as in the field of virtual screening, where the task is to analyze large databases of chemical compounds to identify possible drug candidates [233]. By applying clustering techniques it is possible to prestructure the chemical space, e.g., for local modeling to capture the multi-mechanistic nature of many endpoints, the rediscovery of analog series or visualization. Traditional graph clustering approaches ignore the topological structure in the graph data and transform the graphs into a feature vector-based representation [164, 250]. These techniques have the advantage of being highly efficient, but at the same time imply a loss of information with respect to the graph topology. On the other hand, more sophisticated graph clustering approaches are directly based on the structure of the graphs. These techniques have the desirable property that the calculated similarity measure is intuitive and can be visualized easily. However, they suffer from efficiency and scalability problems with respect to large graph data sets. Most of the proposed structure-based clustering approaches are quadratic in runtime, as every new instance has to be compared to every other instance. As graph data grows in scale, it becomes increasingly more challenging to identify clusters. Thus, there is a pressing need for graph clustering algorithms that are able to handle large databases of graph.

To address this challenge, this thesis presents contributions in the field of graph clustering with a special focus on scalability. In particular, the thesis introduces scalable approaches for clustering large data sets of graphs by common scaffolds, i.e., the existence of one sufficiently large substructure shared by all cluster elements. Graph clustering or, in more general, the definition of local structural graph (similarity) neighborhoods can be useful for a variety of purposes, e.g., to build local models for classification or regression [38]. To illustrate the usefulness of local structural graph neighborhoods, this thesis further investigates applications that exploit the structural neighborhood of graphs to build models for classification and/or regression.

## 1.2 Contributions

There are six main contributions presented in this thesis:

1. An online algorithm based on frequent subgraph mining for clustering graph databases in terms of structural similarity,
2. A parallelized structural graph clustering algorithm that can handle large databases of graphs,
3. A scaffold-based graph clustering algorithm based on abstract pre-clustering capable of clustering large databases containing millions of graphs,
4. A locally weighted learning approach for regression on graphs that defines local structural neighborhoods of test instances by the size of common subgraphs,
5. A novel approach enabling kernel methods to utilize additional information hidden in the structural neighborhood of the graphs, and
6. An approach for extracting information from support vector machines for pattern-based classification.

First of all, a novel structural graph clustering algorithm is presented that clusters large graph databases according to scaffolds, i.e., large structural overlaps that are shared among all cluster members. To do so, the problem formulation takes advantage of a frequent subgraph mining algorithm without effectively generating thousands of subgraphs in the process. In clustering the data, the proposed clustering approach requires the cluster members to share at least one common subgraph that covers a specific fraction of the graphs in the cluster. The algorithm works in an online fashion, i.e., it processes one structure after the other and produces overlapping (non-disjoint) and non-exhaustive clusters.

An important challenge in this endeavor is the scalability to large graph data sets. Graph databases such as the ones representing chemical compounds routinely encompass several hundred thousand graphs; thus, clustering methods that are able to explore and structure the vast graph space are highly desirable. However, the majority of structural, i.e., scaffold-based, graph-based clustering algorithms, involving, e.g., the computation of the Maximum Common Subgraph (MCS) which is known to be NP-hard, is hardly suitable for such data sets. Graph data sets covered in related papers typically contain only several hundred graphs [3, 107, 182], and hardly any effort has been spent on characterizing the performance of the clustering algorithms. To address this shortcoming, the aforementioned structural clustering algorithm has been extended to handle larger graph databases. The novel structural graph clustering algorithm, called PSCG (Parallel Structural Clustering of Graph), is based on the idea of task partitioning in conjunction with refined cluster membership tests. More precisely, a set abstraction of graphs and a size-based clustering

criterion are used to reduce the number of expensive subgraph search computations, which are not affordable exhaustively on large databases. Moreover, to avoid cluster comparisons with all cluster members, which grow computationally more expensive with increasing cluster size, a cluster representative is defined for each cluster once a unique cluster scaffold is found. PSCG is able to handle graph data sets of at least 300,000 graphs. Still, common graph databases, such as real-world compound libraries employed for virtual screening, contain millions of molecular graph structures, and clustering algorithms to structure these libraries are needed. To address this challenge, a scaffold-based algorithm for clustering such very large molecular graph databases is presented. The approach, named SCAP (Structural Clustering by Abstract Pre-clustering), employs two clustering stages. It first partitions the original data set into several smaller data sets using a greedy clustering approach inspired by Dynamic Seed-Based Clustering (DySC) [257] for RNA reads and a similarity measure based on an abstraction from the actual structural similarity measure. The pre-clustering approach is referred to as APreClus (Abstract Pre-Clustering). The motivation behind the pre-clustering step is that by using this pre-clustering approach and the abstraction-based similarity measure, dissimilar partitions of the original data set are generated, without the loss of information. The similarity measure ensures that only graphs which have the potential of being structurally similar are assigned to the same partition. Overall, this leads to a reduction in the number of expensive subgraph search computations performed in the second clustering stage, which are now not required as the partitions are dissimilar from each other. The resulting partitions are further clustered into a finer level of granularity using the highly parallelized scaffold-based clustering approach PSCG that produces overlapping (non-disjoint) and non-exhaustive clusters. The second-stage clustering avoids cluster comparisons with all cluster members, which grow computationally more expensive with increasing cluster size, by defining a cluster representative for each cluster.

Graph clustering or, more generally, the definition of local structural graph (similarity) neighborhoods can be useful for a variety of purposes. Graph clustering or, more generally, the definition of local structural graph (similarity) neighborhoods can be useful for a variety of purposes. One family of methods making use of local neighborhoods are local learning methods. In recent years, local learning methods have experienced renewed interest in the form of local models, i.e., high-quality models of small regions of the input space that often have the advantage of being easier to predict and easier to interpret by domain experts [192]. Several local models together can make up a global model, or global models are the fallback solution (default) when no local model is applicable. One way of defining local models is in terms of clusters. Such an approach was followed by Buchwald *et al.* [38] who defined local models in terms of structural graph clusters obtained by the structural graph clustering approach PSCG. More precisely, the approach called LoMo-Graph (Local Models for Graph Classification and Regression) exploits structural graph neighborhoods in a static way by pre-computing local graph neighborhoods and using them as a basis for building models for classification or regression (e.g., one per cluster).

Local structural neighborhoods may also be used in a dynamic way by determining them for a given test instance individually on demand, at testing time. Such a lazy learning technique is chosen by approaches based on locally weighted learning [10], for instance. Locally weighted learning has been known to be highly effective for regression for a long time, but has not been studied yet for structured data like graphs, although the approach makes sense for at least two reasons. First, it is intuitive to structure the input space before inferring any predictive model, as the structural composition and diversity of typical data sets for graph classification and regression have a serious impact on the predictive performance of methods. A closer look reveals that there exist “structural islands” in many data sets, i.e., subsets of instances that share a large common structural scaffold. Second, for structured objects, local models or local neighborhoods are an opportunity, because the wealth of possible descriptors and similarity/dissimilarity measures enable the use of one view for determining the neighborhood, and a different view for actually making the prediction. For this reason, this thesis studies locally weighted regression on graphs, particularly in the context of so-called Quantitative Structure-Activity Relationships (QSARs), which are models relating chemical structure to biological activity. The approach referred to as LWL-MCS (Locally Weighted Linear Regression based on Maximum Common Subgraph) defines local neighborhoods of test instances by the size of common subgraphs. More specifically, an MCS-based distance measure is employed for locally weighted learning. The actual predictive models are then built using a feature-vector representation of graphs. In an empirical evaluation, the presented approach is compared to other methods using local neighborhoods.

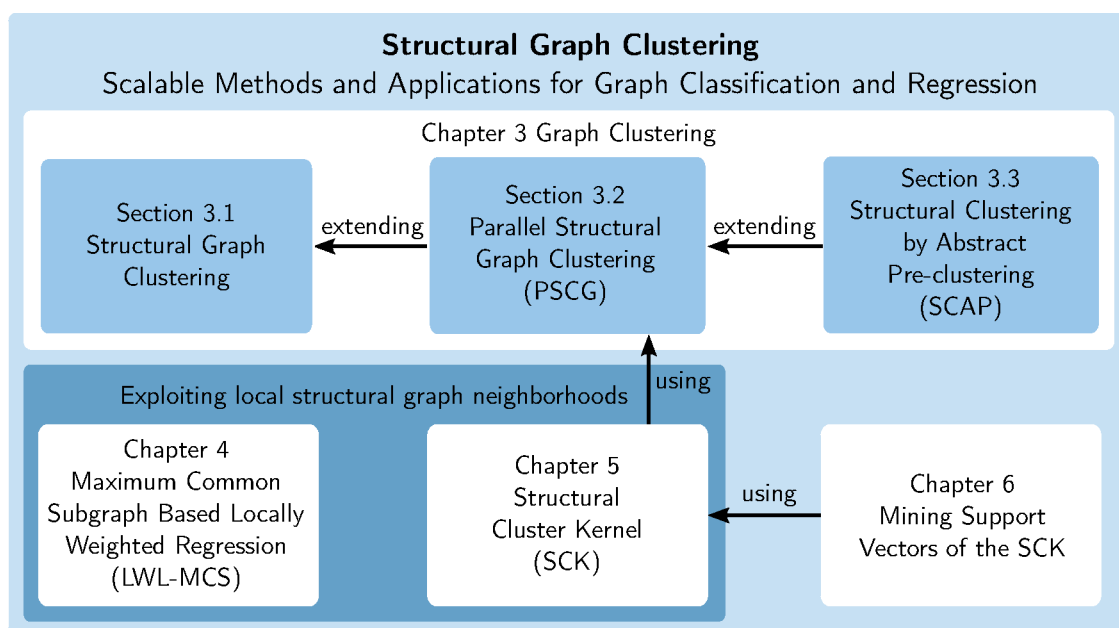
Another family of methods making use of local neighborhoods are kernel methods. Kernels defined on graphs [87] typically compare sets of common graph elements like chains, trees or subgraphs. This thesis introduces a novel approach enabling kernel methods to utilize additional information hidden in the structural neighborhood of the graphs under consideration. More specifically, the novel kernel, called SCK (Structural Cluster Kernel), incorporates similarities induced by the structural clustering algorithm PSCG to improve state-of-the-art graph kernels. The approach taken is based on the idea that graph similarity can not only be described by the similarity between the graphs themselves, but also by the similarity they possess with respect to their structural neighborhood. When this novel graph kernel is used in a Support Vector Machine (SVM) algorithm, a graph prediction system emerges. The approach is applied to a challenging problem in cheminformatics, i.e., the prediction of toxicity and biological activity of chemical compounds.

Besides pattern-based methods, kernel-based methods are the most important classification methods for structured data such as graphs, trees, and sequences. In combination with SVMs, kernel-based methods are often considered as state-of-the-art classification methods in machine learning. An important advantage of SVMs is that their classification decision is based on a subset of the training examples, referred to as the *support vectors*. However, an important drawback of SVMs is typically their black box character. The generated non-linear models frequently lack interpretability, i.e., they do not naturally

provide an explanation of the classification decisions being made. In the light of this, the thesis addresses the following question: Can we make use of trained SVM models to obtain more compact pattern-based classification models? There are at least two possible ways of doing so: by analyzing a trained SVM model together with the training set, or by using the models as oracles to label instances [62]. In this thesis, the former of the two approaches is studied. The proposed approach extracts information from trained support vector machines, in particular their support vectors and their relevance according to their coefficients. It uses the support vectors along with their coefficients as input to pattern mining algorithms able to handle weighted instances. The experiments in the domain of graph mining and molecular graphs show that the resulting models are not significantly less accurate than models trained on the full data sets, yet require only a fraction of the time using much smaller sets of patterns.

Figure 1.1 illustrates the structure of this thesis.

The scientific work of this thesis has been conducted over the course of four years. The algorithms and results presented here have been published in six peer-reviewed conference papers [202, 203, 204, 205, 206, 208]. Research on other topics related to not covered by this thesis was published in one journal article [38] and one peer-reviewed conference paper [207].



**Figure 1.1:** Overview of the contributions of the thesis.

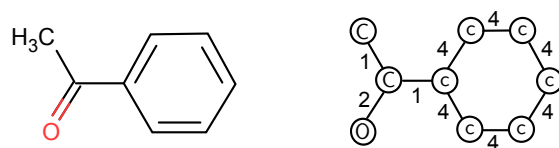
## 1.3 Applications

While data mining and machine learning became popular techniques for a wide range of problems, graph clustering approaches are still rarely used in potential applications. This might be caused by the computational complexity of graph-based approaches. Nevertheless, there are many potential areas such as bio- and cheminformatics, the web, social networking and community detection, which could benefit from the use of graph-based clustering methods. This thesis focuses on applications in the domain of cheminformatics.

In the domain of cheminformatics, graphs are one commonly used representation to model molecular compounds in chemistry [89]. In a vertex- and edge-labeled undirected graph, the vertices and edges correspond to atoms and chemical bonds, respectively. The vertex labels identify symbols of chemical elements, whereas the edge labels characterize the bond type. For example a carbon atom is labeled by  $C$  and an oxygen atom is labeled by  $O$ , whereas a single bond is labeled by 1, a double bond is labeled by 2 and a triple bond is labeled by 3. Most sophisticated representations for molecular graphs further employ special node and bond labels for aromatic bindings, for example inside aromatic rings. An example for a molecular graph representation is depicted in Figure 1.2. In this representation only heavy atoms are taken into account, i.e., hydrogen atoms ( $H$ ) are ignored. Note the special node label (lowercase  $c$ ) and edge label (4) in the ring structure. In cheminformatics, graph clustering algorithms may be useful for a variety of purposes, e.g., to structure the chemical space, for drug design or for predicting characteristics of molecules from their graph structures, e.g., toxicity, or effectiveness as a drug. In the following, some interesting applications in this domain are presented.

### 1.3.1 Definition of Chemical Categories

Graph clustering may be useful for areas such as QSARs and predictive toxicology, where the task is not only prediction, but also the formation of *categories* homogeneous in the structure. Such categories are urgently required, for instance, in the context of REACH (Registration, Evaluation, Authorisation and Restriction of Chemical substances, the European Community Regulation on chemicals and their safe use) [186]. The central axiom of QSARs is that the activity of molecules is reflected in their structure, i.e., structurally similar compounds should also have similar biological activity. Hence, by exploiting structural information in the clustering process, one may expect to find a set of structurally homogeneous clusters reflecting similar biological or toxicological profiles.



**Figure 1.2:** A 2D graph representation of a molecular compound (1-Phenylethanone).



### 1.3.2 Predictive Toxicology

Graph clustering may also be useful for building (local) models for classification or regression, e.g., for predicting toxicology. The learning task in the predictive toxicology setting is as follows: Given data about the molecular structure of some compounds and the compounds' toxic endpoints, learn a model that predicts the toxic endpoints of new compounds. The endpoints represent toxic effects, for instance, brain tumor or kidney failure. Learning tasks of this form are usually known as QSAR. QSAR learning is highly relevant in modern medicinal chemistry and drug design, where automated experiments, for instance from combinatorial chemistry, have led to vast amounts of data that is almost impossible to analyze without the help of computers, data mining and machine learning methods.

### 1.3.3 Virtual Screening

Moreover, graph clustering approaches offer a benefit in the field of virtual screening which involves the use of high-performance computing to analyze large databases of chemical compounds in order to identify possible drug candidates [233]. Virtual Screening has become an integral part of today's drug discovery and drug design process. By applying clustering techniques it is possible to prestructure the chemical space which may provide a means for local modeling to capture the multi-mechanistic nature of many endpoints, the rediscovery and explicit representation of analog series or visualization. Thus, clustering techniques may be useful to get a better understanding of the chemical space and may serve as a supporting aid in virtual screening campaigns. However, the majority of structural graph-based clustering algorithms that are, for example, based on the computation of the MCS, is hardly applicable for such tasks. Hence, there is a pressing need for clustering approaches that are able to handle large databases of chemical compounds.

## 1.4 Outline of the Thesis

This thesis basically consists of two parts. The first part (Chapter 3) covers scalable methods for clustering large databases of graphs that are directly based on the structure of the graphs. The second part (Chapters 4-6) deals with applicative aspects of graph clustering and structural graph neighborhoods for graph classification and regression. In the following, a brief description of the content of each chapter is reported.

Chapter 2 presents the start of the art relevant to this thesis and starts by recalling the basis concepts of graph theory. The notation used in this thesis is introduced, and standard graph clustering algorithms are described. The main focus lies on two types of graph clustering approaches, vector-based and graph-based graph clustering approaches. The chapter continues with an overview of kernels and kernel methods with a particular focus on graph kernels. This part of the chapter ends with the description of support vector machines, the most widespread kernel-based machine learning algorithm nowadays.

The final section in this chapter provides an in-depth review of related work in knowledge extraction, with a particular focus on knowledge extraction methods for SVMs. This section provides context for understanding the contributions of the novel work presented in chapter 6. The chapter contains no original contribution by the author; its sole function is to introduce the definitions and notation that will be used throughout this thesis.

Subsequently, Chapter 3 covers the topic of structural graph clustering and introduces three approaches therefore. First, Section 3.1 presents an approach that clusters databases of graphs according to scaffolds (i.e., large structural overlaps) that are common between cluster members and that produces overlapping (non-disjoint) and non-exhaustive clusters. This approach provides the basis for the following graph clustering approaches. The section introduces the problem of structural graph clustering and presents detailed experimental results, both quantitatively and qualitatively. Section 3.2 is dedicated to a highly parallelized extension of the previously presented structural graph clustering approach for clustering even larger databases of graphs. Finally, Section 3.3 presents a graph clustering approach named SCAP (Structural Clustering by Abstract Pre-clustering) that first partitions the original data set into several smaller data sets using a greedy clustering approach based on dynamic seed clustering. The resulting clusters are further partitioned into a finer level of granularity using an extension of PSCG. The following two chapters present two approaches that exploit the structural neighborhood of graphs for model building.

Chapter 4 presents LWL-MCS, an approach that combines locally weighted learning with graph distances based on the maximum common subgraph. The approach is investigated in the context of regression on graphs, in particular for applications in cheminformatics and for QSARs.

Chapter 5 introduces the structural cluster kernel named SCK, a new family of efficient kernels on graph data structures. It is computed by not only taking into account the similarity between the graphs themselves, but also the similarity they possess with respect to their structural graph neighborhood. The approach is applied to a challenging problem in cheminformatics, i.e., the prediction of toxicity and biological activity of chemical compounds.

Chapter 6 covers the topic of knowledge extraction from SVMs. The proposed approach employs the SVMs on the SCK to investigate the question whether one may make use of these models to obtain more compact pattern-based classification models. To do so, the approach extracts information from trained support vector machines, in particular their support vectors and their relevance according to their coefficients and uses this information as input to pattern mining algorithms able to handle weighted instances.

Finally, Chapter 7 summarizes the contributions of the dissertation and discusses some promising directions for further research and open problems.

# CHAPTER 2

---

## Related Work

---

This chapter provides background material for the remainder of the thesis. Since the thesis focuses on methods for graph data, the chapter starts by giving an introduction to the basics of graph theory in Section 2.1. It gives the notation used throughout the thesis. Subsequently, Section 2.2 provides a review on graph clustering and an overview of the most important algorithms in this area. Next, Section 2.3 gives a brief introduction to kernels and kernel methods with a particular focus on graph kernels. The final section, Section 2.4, surveys related work that has been conducted in the area of knowledge extraction from Support Vector Machines (SVMs).

### 2.1 Introduction to Graph Theory

The purpose of this section is to define terminology and notation for the remainder of this thesis, and to provide the definitions from graph theory that are necessary to understand the graph-based algorithms proposed in this work.

#### 2.1.1 Graph

In its most general form, a graph is composed of a set of nodes connected by edges.

**Definition 2.1 (Graph):**

A *graph* is a pair  $g = (V, E)$  composed of a set of vertices (nodes)  $V$  and a set of edges  $E \subseteq \{(u, v) | u, v \in V\}$ .

Depending on whether directions are assigned to edges, the resulting graph is directed

or undirected.

**Definition 2.2 (Directed and Undirected Graph):**

A graph  $g = (V, E)$  is a *directed* graph if the pairs in  $E$  are ordered pairs, i.e., for every  $(u, v) \in E$ ,  $(u, v) \neq (v, u)$ , otherwise  $g$  is said to be an *undirected* graph.

Figure 2.1 (left) gives an example of an undirected graph, Figure 2.1 (center) an example of a directed graph. Assigning labels to nodes and edges in a graph, we obtain a labeled graph.

**Definition 2.3 (Labeled Graph):**

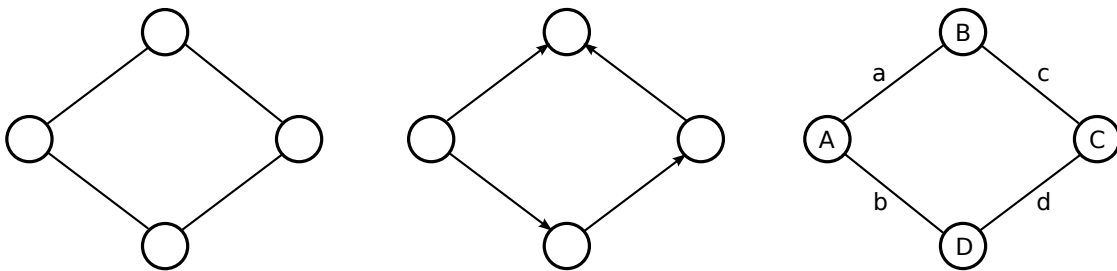
A *labeled* graph is represented as a 4-tuple  $g = (V, E, \alpha, \beta)$ , where  $(V, E)$  is a graph,  $\alpha : V \rightarrow L$  is a mapping that assigns labels to the vertices, and  $\beta : V \times V \rightarrow L$  is a mapping that assigns labels to the edges.

An example of a labeled graph is depicted in Figure 2.1 (right). Throughout this thesis, we are dealing with labeled, undirected graphs. Further, in this thesis, no particular restriction are assumed about graph topologies. In particular, we allow the presence of cycles.

The order (or size) of a graph  $g$  is defined as the number of vertices of  $g$  and is represented as  $|V|$ . The number of edges of  $g$  is denoted by  $|E|$ .

### 2.1.2 Graph Isomorphism and Subgraph Isomorphism

To check if two graphs are equivalent, a concept, namely isomorphism, is required. In general terms, an isomorphism is defined as a map between objects that preserves structure. Two objects are called isomorphic, if an isomorphism exists between them. The problem



**Figure 2.1:** Examples for directed, undirected and labeled graphs. Left: Undirected graph. Center: Directed graph. Right: Labeled undirected graph.

of graph isomorphism is defined as follows:

**Definition 2.4 (Graph Isomorphism):**

Let  $g = (V, E, \alpha, \beta)$  and  $g' = (V', E', \alpha', \beta')$  be two graphs. A graph isomorphism is a bijective function  $f : V \rightarrow V'$  satisfying

1.  $\alpha(u) = \alpha'(f(u))$  for all nodes  $u \in V$
2. for each edge  $e = (u, v) \in E$ , there exists an edge  $e' = (f(u), f(v)) \in E'$  such that  $\beta(e) = \beta'(e')$
3. for each edge  $e' = (u, v) \in E'$ , there exists an edge  $e = (f^{-1}(u), f^{-1}(v)) \in E$  such that  $\beta(e) = \beta'(e')$

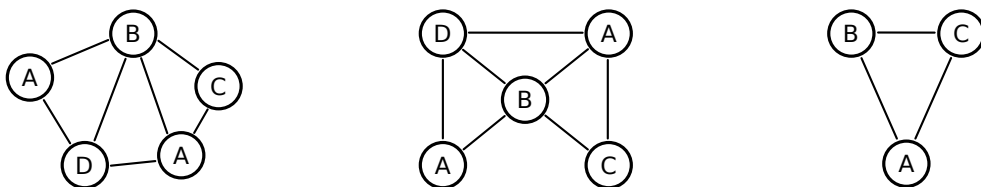
In Figure 2.2 two isomorphic graphs are shown (left and centered graph).

**Definition 2.5 (Subgraph):**

Given two labeled graphs  $g = (V, E, \alpha, \beta)$  and  $g' = (V', E', \alpha', \beta')$ ,  $g'$  is a *subgraph* of  $g$ , denoted by  $(g' \subseteq g)$  if:

1.  $V' \subseteq V$
2.  $E' \subseteq E$
3.  $\forall u \in V' : \alpha'(u) = \alpha(u)$
4.  $\forall (u, v) \in V' \times V' : \beta'(u, v) = \beta(u, v)$

Closely related to graph isomorphism is subgraph isomorphism, which can be seen as a concept describing subgraph equality. A subgraph isomorphism is a weaker form of matching in terms of only requiring that an isomorphism holds between a graph  $g$  and a subgraph of  $g$ . Intuitively, subgraph isomorphism is the problem of detecting whether a smaller graph is identically present in a larger graph. In Figure 2.2, an example of



**Figure 2.2:** Examples for graph isomorphism and subgraph isomorphism. The centered graph is isomorphic to the left graph, and the right graph is isomorphic to a subgraph of the left graph. The node labels are indicated by different letters. All edge are assumed to have identical edge labels.

subgraph isomorphism is given. The right graph is a subgraph of the left graph.

**Definition 2.6 (Subgraph Isomorphism):**

Let  $g = (V, E, \alpha, \beta)$  and  $g' = (V', E', \alpha', \beta')$  be graphs. An injective function  $f : V \rightarrow V'$  from  $g$  to  $g'$  is a subgraph isomorphism if there exists a subgraph  $g'' \subseteq g'$  such that  $f$  is a graph isomorphism between  $g$  and  $g''$ .

In this context, we are often interested in common subgraphs or maximum common subgraphs.

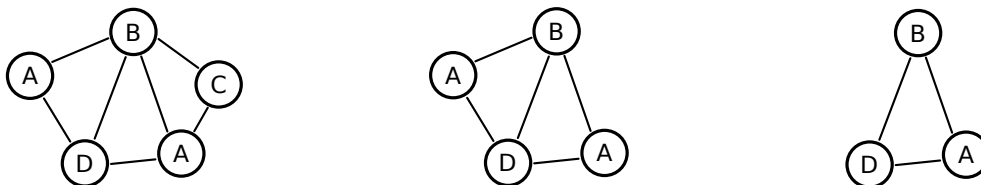
**Definition 2.7 (Common Subgraph):**

Given two arbitrary labeled graphs  $g = (V, E, \alpha, \beta)$  and  $g' = (V', E', \alpha', \beta')$ , a *common subgraph* of  $g$  and  $g'$ ,  $cs(g, g')$ , is a graph  $g'' = (V'', E'', \alpha, \beta)$  such that there exists a *subgraph isomorphism* from  $g''$  to  $g$  and from  $g''$  to  $g'$ . This can be generalized to sets of graphs. The set of common subgraphs of a set of graphs  $\{g_1, \dots, g_n\}$  is then denoted by  $cs(\{g_1, \dots, g_n\})$ .

**Definition 2.8 (Maximum Common Subgraph):**

Given two graphs  $g$  and  $g'$ , a graph  $g''$  is called a Maximum Common Subgraph (MCS) of  $g$  and  $g'$  if  $g''$  is a common subgraph of  $g$  and  $g'$  and there exists no other common subgraph of  $g$  and  $g'$  that has more vertices than  $g''$ .

The maximum common subgraph of two graphs  $g$  and  $g'$  can be seen as the intersection of  $g$  and  $g'$ . In other words, the maximum common subgraph refers to the largest part of two graphs that is identical in terms of structure and labels. Note that in general the maximum common subgraphs needs not to be unique, i.e., there might be more than one maximum common subgraph of identical size for two given graphs  $g$  and  $g'$ . In Figure 2.3 the maximum common subgraph (right graph) is shown for two graphs (the left and the centered graph).



**Figure 2.3:** Maximum common subgraph example. The right graph is a maximum common subgraph of the left graph and centered graph.

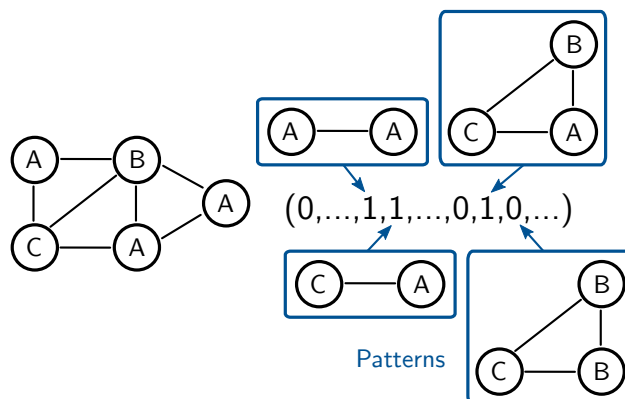
## 2.2 Graph Clustering

Any nonuniform data contains an underlying structure due to the heterogeneity of the data. The process of identifying this structure in terms of grouping the data elements is called clustering [2]. The resulting groups are called clusters. The grouping is usually based on some similarity measure defined for the data elements. Clustering is closely related to unsupervised learning. A basic task in unsupervised learning is to classify a data set into two or more classes based on a similarity measure over the data, without resorting to any a priori information on how the classification should be done. Clustering is a well studied topic in the literature, and various approaches have been proposed up to now [116, 128, 253]. In recent years, the clustering problem has also been increasingly investigated in the context of graph data [4, 79, 181]. Graphs are structures formed by a set of vertices (also called nodes) and a set of edges that are connections between pairs of vertices. Graph clustering is an important graph mining task that aims at grouping the graphs in a data set into clusters taking into consideration the structure of the graph. The identification of clusters in graph data is useful in many applications involving bio- and cheminformatics and the web. The problem of clustering in the graph domain has traditionally been studied in the context of node clustering of individual graphs, in which one attempts to decompose a single graph in the sense of grouping the vertices of a given input graph into clusters. This kind of clustering, sometimes also referred to as community detection [81, 195], has traditionally been studied in the context of graph-partitioning [1, 126, 131], minimum-cut determination [125], network structure clustering [181] and dense subgraph determination [91, 252]. In recent years, the problem of clustering graphs has also been investigated in the context of clustering graph-based data, where one attempts to cluster many different individual graphs (as objects), which are defined over a particular domain of vertices, based on their structural similarity [3, 67]. This thesis focuses on the latter class of clustering problems, i.e., on clustering of graph-based data.

Graph-based clustering algorithms can be further divided into two complementary graph clustering approaches [185]. Traditional graph clustering approaches ignore the topological structure in the graph data, establish a set of features or invariants from a structural description of a graph, and use these features in a vector representation to which various similarity or distance measures can be applied [68, 164, 194, 241, 250]. The feature vector can be composed of properties of the graph and/or of subgraph occurrences. Figure 2.4 shows an example of a graph represented as a binary feature vector of subgraph pattern indicators. Feature vectors are one of the most common and widely used data representations offering a number of useful properties, in particular, the mathematical wealth of operations available in a vector space. For instance, the sum, the product or the mean between two objects is well defined in vector spaces, and moreover, can be efficiently computed. Computing the similarity/distance between two objects represented by vectors is straightforward, too. The convenience and low computational complexity of algorithms that employ feature vectors as their input have resulted in a rich repository of algorithm-

mic tools for clustering and classification, such as k-means clustering, Bayesian classifiers, Neural Networks, SVMs, and many more [24, 72, 210]. However, despite the fact that these representations have the advantage of being highly efficient, they are at the same time associated with representational limitations. More specifically, feature vector-based representations imply a loss of information with respect to the graph topology. Further, they are constrained to a predefined length, which has to be preserved for all objects encountered in a particular application.

The second class of graph clustering approaches use the structure of the graphs directly. Compared to feature vectors, graphs are a much more powerful and flexible tool to represent structured objects and have a higher representational power than feature vectors. Whereas feature vectors provide no direct possibility to describe structural relations in the objects under consideration, graphs are not only able to describe properties of an object, but can also explicitly model the structural relationship between objects. Moreover, in contrast to feature vectors, graphs are not constrained to a fixed size, i.e., the number of nodes and edges is not limited a priori and can be adapted to the size or the complexity of each individual object under consideration. Thus, the more complex an object is, the larger the number of nodes and edges can be. In recent years, an increasing interest in graph-based object representations has been observed, due to the ability of graphs to represent properties of entities and binary relations at the same time [57]. For instance, graph-based representations have been intensively employed in the areas of bio- and cheminformatics [31, 32, 33, 152, 179]. Web content mining represents another area of research that investigated graphs with an emerging interest [196, 197]. While the majority of work in the area of graph clustering is based on feature vector representations, relatively little attention has been paid to the clustering of symbolic structures, such as graphs. This section gives an overview of standard techniques for clustering databases of graphs. Section 2.2.1 reviews work on graph clustering methods based on feature vector representations, while Section 2.2.2 focuses on methods that exploit the structure of the graphs directly.



**Figure 2.4:** Example of a graph represented by a binary feature vector indicating the presence or absence of subgraph occurrences.



### 2.2.1 Clustering Vectorial Data

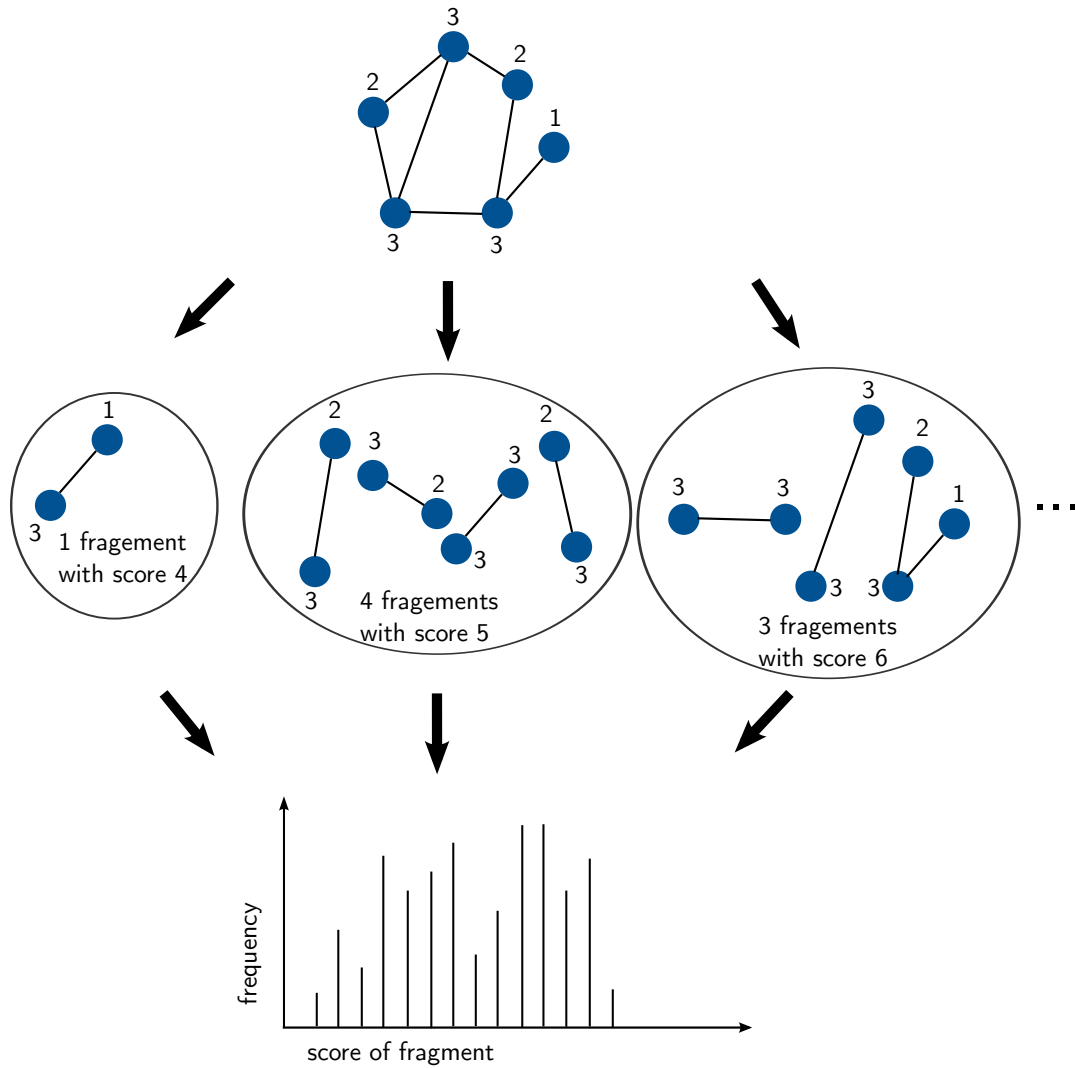
This section gives a brief survey on graph clustering approaches characterized by the use of feature vectors.

Yoshida *et al.* [250] introduced a graph clustering method based on structural similarity of fragments in graph-structured data considering connected subgraphs only. The proposed approach characterizes a fragment based on the connectivity (degree) of a node in the fragment. The representation of a graph is transformed into a fragment spectrum which represents the frequency distribution of fragments in terms of the connectivity of a node within the fragment. Figure 2.5 shows an example of the construction of a fragment spectrum using their approach. To extract connected subgraphs from graph-structured data, the authors employ the graph mining method called Graph-Based Induction (GBI) [161]. The graphs are then clustered with respect to the transformed fragment spectra by applying the standard clustering method k-means. To determine an appropriate number of clusters, the authors estimate the quality of clustering based on a pseudo-entropy for a cluster. The approach is experimentally evaluated on synthetic data only and does not consider edge and node labels.

In work by De Mauro *et al.* [68], a mapping from the domain of graphs to feature vectors is realized by means of a neural network. Further, the authors proposed a subsequent clustering procedure in the feature space, which is based on topological constraints defining the similarity or the dissimilarity between two input graphs.

Another indirect approach to graph clustering was introduced by Luo *et al.* [148] employing vectors of graph-spectral features. For each graph, the authors compute the adjacency matrix and use the leading eigenvectors of this matrix to define clusters of nodes. More precisely, each of the leading eigenvectors represents a cluster of nodes and is mapped to a component of a feature vector using the eigenvalue order to index the components. The length of the vectors are determined by the number of leading eigenvalues. Different graph spectral features are used as components of the feature vectors, i.e., the cluster volume, the cluster perimeter, the cluster Cheeger constant, the inter-cluster edge distance, and the shared perimeter length. Aiming to explore whether these vectors can be used for the purposes of graph-clustering, the authors investigate the use of both central and pairwise clustering methods.

In subsequent work, Luo *et al.* [147] investigated whether the independent or principal components of the spectral feature vectors can be used to embed graphs in a pattern space suitable for clustering. More specifically, similar to previous work [148] the authors use the leading eigenvalues and eigenvectors of the unweighted graph adjacency matrix to calculate graph spectral feature vectors. Next, these vectors are embedded in a lower dimensional pattern space using both Principal Component Analysis (PCA) and Independent Component Analysis (ICA). The authors further studied which of the spectral features results in the best clusters. The experimental results demonstrate that the ICA embedding is better than the PCA embedding for clustering graphs. Of the spectral features used in



**Figure 2.5:** Fragment spectrum of a graph adapted from Yoshida *et al.* [250].

the experiments, the eigenvalues of the adjacency matrix and the shared perimeters result in the pattern spaces with the best cluster structure.

Similar to Luo *et al.*, Hancock *et al.* [75, 149, 189, 221] use spectral theory to convert graphs into vectors by means of spectral decomposition into eigenvalues and eigenvectors of the adjacency (or Laplacian) matrix of a graph.

To keep the representational power of graphs while being able to operate in a vector space, Ferrer *et al.* [78] use an approach called graph embedding as a way to map graphs into a vector space [188] using the graph edit distance [39] to map each graph into a vector space. The median of the set of vectors obtained with this mapping can be easily computed in the vector space. Then, using the two closest points in the vector space and the weighted mean of a pair of graphs [41] they obtain an approximation of the median graph as the final result. In the experiment reported in their paper, the authors focus on employing the k-means algorithm [151, 214] using both the set median and the generalized median as the cluster representatives and comparing the two approaches to each other.

The results – evaluated through two standard clustering performance measures (the Rand index and the Dunn index) – demonstrate that the generalized median graph yields better performance than the set median graph.

### 2.2.2 Graph-based Clustering

This section gives a short survey of graph clustering approaches that operate directly on graphs. Clustering and classification of graphs have wide-spread applications in diverse fields such as pattern recognition, image analysis, drug discovery, or biometrics. The general objective of graph-based clustering approaches is to group graphs in a given data set based on structural similarity.

A graph clustering algorithm, which is an extension of Kohonen’s well-known Self-Organizing Map (SOM) algorithm [134], was introduced by Günter [94] and Günter and Bunke [95]. SOM has become an established tool in pattern recognition and related areas. Whereas the classical SOM approach is based on vectorial pattern representations, the clustering algorithm by the authors works in the domain of graphs. To cluster graph-based data under the extension of SOMs, the graph edit distance [168] is used as distance measure between graphs. The SOMs procedure for neuron updating in the graph domain is based on the concept of the weighted mean of a pair of graphs. The approach is experimentally evaluated on the graph representations of capital letters. However, in the paper only those characters from the alphabet were considered that consist of straight lines only. In subsequent work, Günter and Bunke [96] extended the algorithm by proposing a number of cluster validation indices from feature vector representations to the graph domain allowing the application of the graph clustering algorithm without any prior knowledge of the number of clusters. As in the previous paper, graph representations of capital letters composed of straight line segments only were used to show the feasibility of the proposed method.

Schenker *et al.* [196] introduced an extension of the k-means clustering algorithm to graph-based representations. To represent the center of a cluster, the set median graph [119] is used. The median of a set of graphs is defined as the graph that has the lowest average distance to all graphs in the set. In their work, the authors compare the representational power of feature vectors and graphs under the context of web content mining. The experimental results show better accuracies of the graph-based approaches over the comparable vector-based methods.

Jiang *et al.* [119] introduced the concept of the generalized median graph which generally provides a better cluster representation than the set median graph. Given a set of graphs, the generalized median graph is defined as the graph that has the minimum sum of distances to all graphs in the set. It can be seen as the representative of the set. While the generalized median graph has a large number of potential applications in many classical algorithms for learning, clustering and classification, its main disadvantage is that its computation is exponential both in the number of input graphs and their size [42]. A

number of algorithms for the generalized median graph computation have been reported in the past [77, 103, 119]. For instance, Hlaoui and Wang [103] proposed an approximate algorithm for computing the generalized median graph from a set of graphs which is used to extend the k-means-based algorithm to graph clustering. The experimental evaluation on random graphs and on a synthetic image database demonstrates the efficiency of the proposed algorithm in correctly classifying graphs into sets of clusters. However, in general these algorithms suffer from either a large complexity or are restricted to special types of graphs.

To make cluster algorithms based on centers such as k-means applicable to graphs, Bunke *et al.* [40] constructed a supergraph as cluster representation. More precisely, the authors introduced the notion of the weighted minimum common supergraph (WMCS) of a cluster, which is a graph summarizing all the properties of all the graphs belonging to the cluster. Further, they introduced a nearest neighbor clustering in which graphs are added to clusters such that the change in entropy within the cluster is minimized.

In work by Raymond *et al.* [182], the suitability of graph-based similarity measures for chemical clustering was evaluated and their effectiveness was compared with that of fingerprint-based measures. The authors employed several clustering methods (e.g., the Ward's method and the Jarvis-Patrick method) to process graph-based similarities, with the results from conventional fingerprint-based similarities providing a benchmark of comparison. More specifically, they applied their method for the identification of the Maximum Common Edge Subgraph (MCES) [183, 184] to the calculation of inter-graph similarities based on the graph similarity coefficient of Wallis. In their comparison, Raymond *et al.* [182] reported that no obvious advantage results from the use of the more sophisticated, graph-based similarity measures. They draw the conclusion, that although the results obtained from the use of graph-based similarities are different from fingerprint-based similarities, there is no evidence to suggest that one approach is consistently better than the other.

Dalamagas *et al.* [67] presented a methodology for clustering structurally similar Extensible Markup Language (XML) documents. Modeling XML documents as rooted ordered labeled trees, the authors face the problem of clustering XML documents by structure as a tree clustering problem. To estimate the structural similarity between XML documents, the authors defined a structural distance metric. Motivated by their claim that real XML documents tend to have many repeated nodes which affect the performance of the tree edit algorithms, they introduced a summary tree structure in which the repeated nesting nodes are reduced (or removed) from the rooted labeled trees. According to the authors, these tree structural summaries have minimal processing requirements while maintaining the structural relationships of the elements in an XML document. Further, the authors presented a new algorithm to calculate tree edit distances using a dynamic programming algorithm. Given two trees  $T_1$  and  $T_2$  representing two XML documents, a tree edit sequence is a sequence of tree edit operations (insert node, delete node, etc) to transform  $T_1$  to  $T_2$ . Assuming a cost model to assign costs for every tree edit operation, the tree edit

distance between  $T_1$  and  $T_2$  is defined as the minimum cost among the costs of all possible tree edit sequences that transform  $T_1$  to  $T_2$ . The experimental results demonstrate that the usage of structural summaries improves the performance of the clustering procedure without compromising cluster quality. However, due to complexity issues, edit distance based approaches are infeasible for large data collections.

A different approach for clustering XML data based on their structure was proposed by Aggarwal *et al.* [3]. Similar to Dalamagas *et al.* [67], the authors exploit a tree representation of XML documents in which a XML document is viewed as a rooted ordered labeled tree. However, instead of calculating the tree edit distance between any pair of XML documents, they compute XML similarity in terms of coverage of frequent substructures at a specified support level. The XProj algorithm uses a set of frequent substructures as a representative for a given cluster of XML documents. The clustering algorithm is a partition based algorithm, which constructs groups that maximize the structural similarities among the documents within a group. In order to make the structural similarity more comparable among different sets of representatives and the similarity calculation more efficient, the authors only consider frequent substructures of a specified size as representatives. Further, to speed up the frequent substructure representative mining, XProj adopts a set of high quality approximate structures, that is, sequences of tree edges. For mining frequent sequences, the sequential pattern mining algorithm BIDE [234] was revised in order to terminate search once a sequence reaches a specified size. The experimental results show that XProj produces clusters of higher quality and higher precision compared to the approach by Dalamagas *et al.* [67].

Hossain and Angryk [107] introduced a new technique for document clustering based on co-occurrence of frequent subgraphs in the documents. To discover frequent subgraphs, GDClust utilizes graph-based mining technology. More specifically, GDClust represents text documents as hierarchical document-graphs and utilizes an Apriori paradigm [6] for finding frequent subgraphs. Discovered frequent subgraphs are then utilized by a Hierarchical Agglomerative Clustering (HAC) [253] to cluster documents depending on the similarity of the subgraphs in the document-graphs. However, rather than focusing only on the co-occurrence of frequent terms in text documents, GDClust enables clustering of documents providing humanlike sense-based searching capabilities. In other words, GDClust is able to group documents in the same cluster even if they do not contain common keywords, but still possess the same sense. The approach is motivated by the way human beings process text data.

Tsuda and Kudo [223] proposed an Expectation Maximization (EM)-based method for clustering graphs based on weighted frequent pattern mining. In their approach, a set of informative patterns are efficiently collected based on latent cluster labels. The proposed method is probabilistic and adopts a binomial mixture model defined on a very high dimensional vector indicating the presence or absence of all possible subgraph patterns. The approach has several drawbacks. First, even though the method retrieves discriminant patterns that are useful for understanding the obtained clusters, the number of selected

patterns may be too many for interpretation. Second, the number of clusters has to be specified a priori. Third, the method cannot take into account the similarity of vertex and edge labels.

Tsuda and Kurihara [225] presented a graph clustering approach based on frequent pattern mining that addresses the problem of learning a Dirichlet process (DP) mixture model in the high dimensional feature space of graph data. Due to efficiency reasons variational inference [26, 142] is adopted. To limit the dimensionality of the feature space, the authors formulated a feature saliency criterion and developed a search algorithm to find best patterns. More specifically, an approach has been proposed that selects features by minimizing the variational free energy. To find the best patterns quickly, a depth-first search (DFS) code tree [245] has been adopted, where the generation of useless subgraphs is suppressed by a tree pruning. Despite the proposed feature selection method to obtain a reduced feature set, DP clustering, however, still outputs a large number of frequent subgraphs which make the graph clusters difficult to interpret.

In 2009, Jouili and Tabbone [123] introduced a hypergraph model to cluster a set of graphs that allows for overlapping clusters. A hypergraph consists of a set of vertices and a set of hyperedges where each hyperedge is a subset of vertices. In their model, each graph is represented by a vertex and each cluster by a hyperedge. The degree of a vertex is the number of hyperedges it belongs to, and the degree of a hyperedge is the number of vertices it contains. Establishing a hypergraph-based model for a graph database, the authors introduced a clustering technique based on the prototype selection to cluster the graph set into  $k$  independent clusters. Using the concept of the graph median and a given threshold, the proposed algorithm automatically detects the number of clusters. Further, the proposed method allows for overlapping clusters, i.e., a graph can be assigned to more than one cluster.

In subsequent work, Jouili *et al.* [124] proposed a graph clustering algorithm that is an adaptation of the mean-shift algorithm [56] into the domain of graphs. The notion of a set median and a generalized median graph is used to implement the shifting operation instead of the mean in the classical mean-shift clustering. The median graph shift clustering is a deterministic and non-parametric algorithm. It computes the number of clusters during execution. The authors performed a set of clustering experiments on three data sets using two validation indices. In addition, a comparison with k-means clustering [151, 214] is provided. For the k-means algorithm, the graph edit distance approximation and the set median graph are used to compute the centers and to perform the clustering.

## 2.3 Kernel Methods

Kernel methods are a powerful class of methods for pattern analysis and classification which are widely applicable and known for their state-of-the-art performance. In recent years, they have attracted considerable interest in the machine learning community and are increasingly used for solving various real-world problems, such as molecule classification

[152, 179], protein prediction [33, 144], image classification [100], text classification [120] and handwriting recognition [12]. Several reasons contribute to the increasing interest in and success of kernel methods. First of all, they attract a lot of attention due to their solid foundation built on mathematics and statistical learning. By means of kernel functions standard algorithms which were originally designed for feature vectors can be applied to more complex data structures such as strings, trees, or graphs. Thus, kernel methods are able to bridge the gap between statistical and structural pattern recognition. Further, kernel methods allow the extension of basic linear algorithms to complex non-linear methods in a simple and elegant way.

This section gives a brief introduction to kernel methods. For a complete and in-depth treatment of kernel methods, the reader is referred to the textbooks by Schölkopf and Smola [199] and Shawe-Taylor and Cristianini [210]. The section starts by introducing kernel functions and some basic properties for patterns given in terms of vectors. Next, the extension of kernel functions to structural data and in particular to graph-based representations is described. Later in this section, the concept of cluster kernels is introduced. Finally, the formulation of support vector machines is described in detail.

### 2.3.1 Kernels

In statistical pattern recognition the input patterns are traditionally given by vectors of real-valued numbers, while in structural pattern recognition, graphs can be employed for representing the available data. Algorithms for analysis and recognition are commonly designed such that they directly work with the actual data structures in the specific pattern space. In kernel methods, the underlying data is represented in an essentially different way [201]. Here, an explicit representation of the data is of secondary interest. In other words, kernel methods do not depend upon individual pattern representations, but instead rely on the notion of pairwise similarity between the data at hand. More formally, given a pattern space  $X$  with  $n$  patterns or objects  $\{x_1, \dots, x_n\} \subseteq X$ . A kernel is defined by a real-valued similarity function  $k : X \times X \rightarrow \mathbb{R}$ , referred to as *kernel function*, which represents the pattern space  $X$  in an implicit way by means of pairwise kernel values  $k_{ij} = k(x_i, x_j)$ . This implicit representation is used for the design of algorithms. That is, instead of working with individual data entities, the designed algorithms work with pairwise kernel values. Hence, in order for kernel based algorithms to be applicable to patterns from some pattern space  $X$ , a kernel function  $k : X \times X \rightarrow \mathbb{R}$  needs to be defined.

In contrast to kernel methods, kernel functions need an individual pattern representation meaning that the kernel function employed in the pattern space  $X$  is defined with respect to the patterns in  $X$ . For example, kernels exist that are exclusively designed for vectors or for graphs. In this section, the pattern space  $X$  will be some possibly infinite dimensional vector space  $\mathcal{H}$ , whereas in Section 2.3.2 the pattern space  $X$  will be given by the domain of graphs  $\mathcal{G}$ .

Kernel functions can be used to derive information from patterns that is useful for tasks

such as clustering, classification and regression. Generally, a function is a valid kernel function if it satisfies symmetry, i.e.,  $k(x_i, x_j) = k(x_j, x_i)$ , and positive definiteness in the following sense:

**Definition 2.9 (Positive Definite Kernel):**

A symmetric function  $k : X \times X \rightarrow \mathbb{R}$  is a positive definite<sup>1</sup> kernel if, for any  $n \in \mathbb{N}$ ,  $x_1, \dots, x_n \in X$ , and  $c_1, \dots, c_n \in \mathbb{R}$

$$\sum_{i,j=1}^n c_i c_j k(x_i, x_j) \geq 0. \quad (2.1)$$

**Definition 2.10 (Kernel Matrix):**

Given a positive definite kernel function  $k$  and patterns  $x_1, \dots, x_n \in X$ , a  $n \times n$  matrix  $K$

$$K = \begin{bmatrix} k_{11} & k_{12} & \dots & k_{1n} \\ k_{21} & k_{22} & \dots & k_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ k_{n1} & k_{n2} & \dots & k_{nn} \end{bmatrix} \quad (2.2)$$

can be formed such that  $K_{ij} = k(x_i, x_j)$  for  $i, j = 1, \dots, n$ . This matrix is called the kernel matrix (or Gram matrix) of  $k$  with respect to  $x_1, \dots, x_n$

To verify whether a kernel function is positive definite, one can check if the condition in Definition 2.9 is satisfied. This is equivalent for the kernel matrix  $K = k_{ij}$  to be positive definite.

Given these definitions, it follows that if  $k$  is a positive definite kernel function, a feature space can be constructed in which  $k$  is the dot product. More precisely, a *Hilbert space*  $\mathcal{H}$  can be constructed with

$$k(x, x') = \langle \varphi(x), \varphi(x') \rangle. \quad (2.3)$$

According to Meschkowski [167], a dot product space  $\mathcal{H}$  is called a Hilbert space if it is complete with respect to the metric  $d(x, y) = \|x - y\|$ . The Hilbert space associated with a kernel is referred to as a *reproducing kernel Hilbert space*. By means of functional analysis it can be shown that every kernel function is associated with a reproducing kernel Hilbert space and that every reproducing kernel Hilbert space is associated with a kernel function [43].

---

<sup>1</sup> Note that in mathematics, functions according to this definition are called positive semidefinite, since the sum  $\sum_{i,j=1}^n c_i c_j k(x_i, x_j)$  is not strictly positive, but can be zero. On the other hand, functions for which this sum is strictly positive are called positive definite functions. For brevity, the term “semi” is often omitted in machine learning.



### 2.3.1.1 Properties of Kernels and Examples

Positive definite kernel functions satisfy a number of closure properties that enable the construction of new kernel functions by combining known ones.

**Proposition 1 (Closure Properties):**

Let  $X$  be an input space,  $k_1$  and  $k_2$  arbitrary positive definite kernels defined over  $X \times X$ ,  $\alpha \in \mathbb{R}^+$ ,  $f : X \rightarrow \mathbb{R}$ ,  $k_3$  a valid kernel over  $\mathcal{H} \times \mathcal{H}$  and  $\varphi : X \rightarrow \mathcal{H}$ . Then the following kernel functions are also positive definite kernels:

1.  $k(x, x') = \alpha k_1(x, x')$
2.  $k(x, x') = k_1(x, x') + k_2(x, x')$ ,
3.  $k(x, x') = k_1(x, x')k_2(x, x')$ ,
4.  $k(x, x') = f(x)f(x')$ ,
5.  $k(x, x') = k_3(\varphi(x), \varphi(x'))$ .

These rules can be used to create new kernel functions by combining known ones. Among the most well-known positive definite kernel functions are the linear, the polynomial and the Gaussian radial basis function (RBF) kernel as shown in Table 2.1.

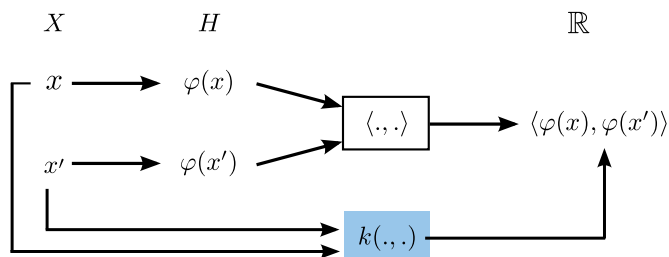
To summarize, two major benefits can be ascribed to the use of kernels. First, by means of kernel similarities between arbitrary objects can be defined. That is, kernels can be designed for any kind of data provided that they are valid kernels. As a matter of fact, several kernels for complex structures, such as sequences, trees, or graphs, have been proposed in the literature so far. Second, valid kernels can be adopted for several learning tasks, such as clustering, classification, regression, or feature extraction, as long as the respective algorithms are based on dot product calculations.

### 2.3.1.2 Kernel Trick

A key advantage of kernels is that they allow the extension of basic linear algorithms to complex non-linear methods. Consider a non-linear function  $\varphi : X \rightarrow \mathcal{H}$  mapping patterns from the original space  $X$  to some feature space  $\mathcal{H}$  of high or even infinite dimensionality. As shown in this section, a kernel function  $k(x, x')$  returns the dot product  $\langle \varphi(x), \varphi(x') \rangle$

**Table 2.1:** List of well-known positive definite kernels for vectorial data.

Kernel	Definition	Parameters
Linear Kernel	$k_{lin}(x, x') = \langle x, x' \rangle$	-
Polynomial Kernel	$k_{poly}(x, x') = (\langle x, x' \rangle + c)^p$	$p \in \mathbb{N}, c \geq 0$
Gaussian Kernel	$k_{RBF}(x, x') = \exp\left(-\frac{\ x - x'\ ^2}{2\sigma^2}\right)$	$\sigma > 0$



**Figure 2.6:** Comparing the explicit mapping of patterns  $x$  and  $x'$  in a feature space  $\mathcal{H}$  via  $\varphi$  and subsequent dot product computation with the shortcut kernel-trick. Note that the pattern space  $X$  can be any domain (e.g. the domain of graphs  $G$ , or a vector space  $\mathcal{H}$ ).

between two maps  $\varphi(x)$  and  $\varphi(x')$  in this (implicitly existing) feature space  $\mathcal{H}$ . Thus, kernel functions enable the evaluation of the dot product between two patterns in the feature space  $\mathcal{H}$  without explicitly computing their coordinates in  $\mathcal{H}$  via the mapping  $\varphi(\cdot)$ . This procedure is commonly termed *kernel trick*. Figure 2.6 illustrates the kernel trick. The kernel trick has a huge impact on the design of machine learning algorithms. That is, any algorithm that can be reformulated in terms of dot products only, can be extended implicitly in the feature space  $\mathcal{H}$  by replacing each dot product  $\langle \cdot, \cdot \rangle$  by a kernel evaluation  $k(\cdot, \cdot)$ . Such algorithms together with some kernel function are commonly referred to as kernel machines. Prominent examples for kernel machines are support vector machines, principal component analysis, and k-means clustering.

Another key advantage of kernel methods is their application to non-vectorial data. Contrary to the initial assumption that  $X \in \mathbb{R}^n$ ,  $X$  can also represent any structured domain, such as the space of strings or graphs. In this case, all kernel methods remain applicable, as long as a mapping  $\varphi : X \rightarrow \mathcal{H}$  can be found. A direct consequence of the kernel trick is that the mapping  $\varphi$  does not need to be determined explicitly. Instead, it is sufficient to find a kernel function  $k(x, x') = \langle \varphi(x), \varphi(x') \rangle$  on pairs of objects from  $X$ . Hence, structured data can be compared via kernels without even explicitly building the feature space  $\varphi$ . Due to this finding, the definition of kernel functions for structured data such as graphs has become a hot topic in machine learning and in application domains such as bioinformatics and cheminformatics over recent years [88, 152, 201]. The following section focuses on kernel functions for graphs.

### 2.3.2 Graph Kernels

In recent years, graph kernels have evolved into a fast developing branch of learning on structured data. Kernel functions for graphs connect structural data with kernel methods. Intuitively, a graph kernel is a measure of similarity between pairs of graphs satisfying the conditions of symmetry and positive definiteness as previously discussed. The design of graph kernels is based on a rich set of fundamentals from graph theory. For a brief review on graph theory, the reader is referred to Section 2.1. The challenge is to define similarity measures capable of capturing the structural commonalities between pairs of graphs. The formal definition of a graph kernel is given in the following;

**Definition 2.11 (Graph Kernel):**

Let  $\mathcal{G}$  be the domain of graphs. The function  $k : \mathcal{G} \times \mathcal{G} \rightarrow \mathbb{R}$  is called a graph kernel if there exists a Hilbert space  $\mathcal{H}$  and a mapping  $\varphi : \mathcal{G} \rightarrow \mathcal{H}$  such that:

$$k(G, G') = \langle \varphi(G), \varphi(G') \rangle \quad \forall G, G' \in \mathcal{G} \quad (2.4)$$

Note that in machine learning the term graph kernel occasionally refers to a kernel between nodes of one large graph, which we call a node kernel. Throughout this thesis, graph kernel will denote a kernel that compares graphs to each other.

Most of the existing graph kernels belong to the class of R-convolution kernels as defined by Haussler [101]. Section 2.3.2.1 will give a short review on this family of kernels. In recent years, various graph kernels have been proposed which can be categorized into three classes: graph kernels based on walks [88, 127] and paths [32], graph kernels based on limited-size subgraphs [106, 138, 211, 212], and graph kernels based on subtree patterns [153, 180]. For an extensive review of graph kernels, the reader is referred to the work by Gärtner [87]. This thesis focuses on two types of subgraph kernels, the Weighted Decomposition Kernel (WDK) [166] and the Neighborhood Subgraph Pairwise Distance Kernel (NSPDK) [61], which are described in Sections 2.3.2.2 and 2.3.2.3. These kernels provide the basis for the newly designed graph kernel approach presented in Chapter 5.

**2.3.2.1 R-Convolution Kernels**

Graph kernels are instances of the so-called R-convolution kernels introduced by Haussler [101]. R-convolution kernels provide a general framework to construct kernels on structured objects by comparing all pairs of decompositions thereof [101]. More specifically, the idea behind convolution kernels is to decompose complex objects into smaller parts, for which a simpler similarity measure can be defined and computed more efficiently. Given the similarities between the smaller parts, a convolution operation can be used to define a kernel function between a pair of complex objects. Let  $x \in X$  be such an object (also referred to as *composite structure*), and let  $\vec{x} = (x_1, \dots, x_D)$  denote a *decomposition* of  $x$ , with each  $x_d \in X_d$  ( $d = 1, \dots, D$ ), e.g., the decomposition of graphs into subgraphs. Let  $R$  be the relation with  $R : \vec{X} \times X \rightarrow \{\text{true}, \text{false}\}$  and  $\vec{X} = X_1 \times \dots \times X_D$ , such that  $R(\vec{x}, x)$  is true if and only if  $\vec{x}$  is a tuple of parts for  $x$ , i.e.,  $\vec{x}$  is a valid decomposition of  $x$  and false otherwise. Consider the set of all valid decompositions of an object, which is defined by the inverse  $R^{-1}(x) = \{\vec{x} | R(\vec{x}, x) = \text{true}\}$ . The R-convolution  $\star$  of the kernels  $k_1, k_2, \dots, k_D$  with  $k_d : X_d \times X_d \rightarrow \mathbb{R}$  is defined as:

$$\begin{aligned} k(x, x') &= k_1 \star k_2 \star \dots \star k_D(x, x') \\ &= \sum_{\substack{\vec{x} \in R^{-1}(x) \\ \vec{x}' \in R^{-1}(x')}} \prod_{d=1}^D k_d(x_d, x'_d), \end{aligned} \quad (2.5)$$

where  $k(x, x')$  is a valid kernel, provided that all the individual  $k_i$  are valid kernels and  $R$  is a finite relation [101]. The deliberately vague formulation with regards to the nature of the underlying decomposition leads to a framework in which many different kernels can be defined by simply changing the decomposition. The application of R-convolution kernels to graphs involves the decomposition of graphs into smaller substructures computable in polynomial time. Several instances of graph kernels have been proposed based on this framework, of which some of the most frequently used substructures are random walks, subtrees, cycles, and shortest paths [230].

Decomposition kernels form a rather vast class. As a result, one needs to carefully tune the relation  $R$  to different applications in order to specify a suitable kernel. A widely used family of kernels are *all-substructures kernels*, which count the number of common substructures in two decomposable objects. In this case,  $D = 1$  and  $\mathcal{R} = \langle X, R, \delta \rangle$  where  $R(x_1, x)$  iff  $x_1$  is a substructure of  $x$  and  $\delta$  is the *exact matching kernel*:

$$\delta(x_1, x'_1) = \begin{cases} 1 & \text{if } x_1 = x'_1 \\ 0 & \text{otherwise.} \end{cases} \quad (2.6)$$

The all-substructure kernel becomes:

$$k(x, x') = \sum_{\substack{x_1 \in R^{-1}(x) \\ x'_1 \in R^{-1}(x')}} \delta(x_1, x'_1), \quad (2.7)$$

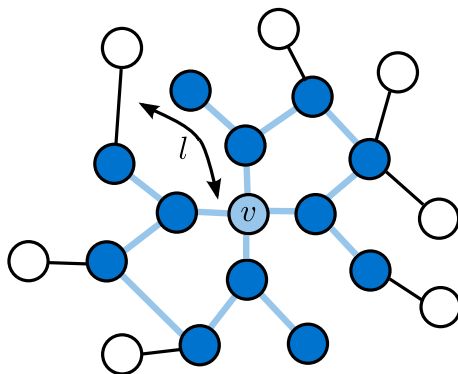
### 2.3.2.2 Weighted Decomposition Kernel

The basic idea of the Weighted Decomposition Kernel (WDK) [166] is to focus on relatively small parts of a structure, called *selectors*, that are matched according to an equality predicate. The importance of the match is then weighted by a factor that depends on the similarity of the *context* in which the matched selectors occur.

More formally, a weighted decomposition kernel is characterized by the decomposition structure  $\mathcal{R} = \langle \vec{X}, R, (\delta, k_1, \dots, k_D) \rangle$ , where  $\vec{X} = (S, Z_1, \dots, Z_D)$ ,  $R = (s, z_1, \dots, z_D, x)$  is true iff  $s \in S$  is a subgraph of  $x$  called the selector and  $\vec{z} = (z_1, \dots, z_D) \in Z_1 \times \dots \times Z_D$  is a tuple of subgraphs of  $x$  called the context of occurrence of  $s$  in  $x$ . This setting results in the following general form of the kernel:

$$K(x, x') = \sum_{(s, \vec{z}) \in R^{-1}(x)} \sum_{(s', \vec{z}') \in R^{-1}(x')} \delta(s, s') \prod_{d=1}^D k_d(z_d, z'_d), \quad (2.8)$$

where  $k_d$  is a kernel on contexts and  $\delta$  is the exact matching kernel applied to selectors. Compared to kernels that simply count the number of substructures, the above function weights different matches between selectors according to contextual information as illustrated in Figure 2.7.



**Figure 2.7:** Example of selector (light blue vertex) and context (dark blue vertices and light blue vertex) for a graph. Adapted from Menchetti [165].

**2.3.2.2.1 A Weighted Decomposition Kernel for Molecules** A molecule is naturally represented by an undirected graph  $x$  where vertices represent atoms and edges represent bonds between atoms. Vertices are annotated with attributes such as atom type, atom charge or membership to specific functional groups (i.e., whether the atom is part of a carbonyl, methyl, alcohol or other group in the molecule), whereas edges are annotated with attributes such as bond type. Contexts are formed as follows: Given a vertex  $v \in V$  and an integer  $l \geq 0$ , called the *context radius*. Further, let  $x(v,l)$  denote the subgraph of  $x$  composed of the vertices within distance  $l$  from vertex  $v$ , and the set of all edges that have at least one end in the vertex set of  $x(v,l)$ . The decomposition relation depending on  $l$  is defined as  $R_l = \{(s,z,x) : x \in X, s = x(v), z = x(v,l), v \in V(x)\}$ , where  $s$  is the selector, which is a single atom, and  $z$  is the context for vertex  $v$ . The matching kernel  $\delta(v,v')$  returns 1 if the two vertices  $v$  and  $v'$  have the same label. Choosing  $D = 1$  the kernel is defined as

$$K(x,x') = \sum_{v \in V(x)} \sum_{v' \in V(x')} \delta(x(v),x'(v')) \cdot k(x(v,l),x'(v',l)). \quad (2.9)$$

### 2.3.2.3 Neighborhood Subgraph Pairwise Distance Kernel

The Neighborhood Subgraph Pairwise Distance Kernel (NSPDK) [61] is an instance of a decomposition kernel, i.e., a composite kernel that operates over all possible parts defined by a given relation. Here, the parts are pairs of special subgraphs, called *neighborhood subgraphs*.

More formally, let  $D(u,v)$  denote the distance between two vertices  $u$  and  $v$ , representing the length of the shortest path between them. Further, let  $N_r(v)$  denote the neighborhood of radius  $r$  of a vertex  $v$ , i.e., the set of vertices at a distance less than or equal to  $r$  from  $v$ . Moreover, let  $N_r^v$  denote the neighborhood subgraph of radius  $r$  of vertex  $v$ , i.e., the subgraph induced by the neighborhood of radius  $r$  of  $v$ . The neighborhood-pair relation  $R_{r,d}(A^v,B^u,G)$ , representing the relation between two rooted graphs  $A^v,B^u$  and a graph  $G$ , is defined to be true iff both  $A^v$  and  $B^u$  are in  $\{N_r^v : v \in V(G)\}$ , where  $A^v$  ( $B^u$ ) is required to be isomorphic to some  $N_r$  and  $D(u,v) = d$ . More precisely, the relation  $R_{r,d}$

selects all pairs of neighborhood graphs of radius  $r$  whose roots are at distance  $d$  in a given graph  $G$ . The authors define  $k_{r,d}$  as the decomposition kernel over  $G \times G$  on the relation  $R_{r,d}$ , i.e.,

$$k_{r,d}(G,G') = \sum_{\substack{A^v, B^u \in R_{r,d}^{-1}(G) \\ A^{v'}, B^{u'} \in R_{r,d}^{-1}(G')}} \delta(A^v, A^{v'}) \delta(B^u, B^{u'}) \quad (2.10)$$

where  $\delta$  is the exact matching kernel, i.e.,  $\delta(x,y)$  is 1 if  $x \simeq y$  (i.e., if the graph  $x$  is isomorphic to  $y$ ) and 0 otherwise. In words:  $k_{r,d}$  counts the number of identical pairs of neighboring graphs of radius  $r$  at distance  $d$  between two graphs.

The NSPDK is finally defined as the sum of all the kernels for all radii and all distances:

$$K(G,G') = \sum_r \sum_d k_{r,d}(G,G'). \quad (2.11)$$

For efficiency reason, the authors consider the zero extension of  $K$  obtained by imposing an upper bound on the radius and the distance:  $K_{r^*,d^*}(G,G') = \sum_{r=0}^{r^*} \sum_{d=0}^{d^*} k_{r,d}(G,G')$ , i.e., NSPDK is limited to the sum of the  $k_{r,d}$  kernels for all increasing values of the radius (distance) parameter up to a maximum given value  $r^*$  ( $d^*$ ).

### 2.3.3 Cluster Kernels

In recent years, the idea of changing the representation given to a classifier by taking into account similarity information induced by a clustering algorithm has attracted some attention. Such techniques are commonly referred to as *cluster kernels*. Several types of cluster kernels, relying on different clustering algorithms, have been proposed by Chapelle *et al.* [47]. The authors present a general framework for constructing cluster kernels which implements the cluster assumption, i.e., the induced distance depends on whether the points are in the same cluster or not. Weston *et al.* [240] investigated the use of cluster kernels for protein classification by developing two simple and scalable methods for modifying a base kernel. The neighborhood kernel uses averaging over a neighborhood of sequences defined by a local sequence similarity measure, and the bagged kernel uses bagged clustering of the full sequence data set to modify the base kernel. In both the semi-supervised and transductive settings, these techniques greatly improve the classification performance when used with mismatch string kernels. In work by Bodo [27], a kernel construction algorithm for supervised and semi-supervised learning was proposed, which constitutes a general framework for semi-supervised kernel construction. The technique clusters the labeled and unlabeled data by an agglomerative clustering technique, and uses the linkage distances induced by the clustering hierarchy to construct the kernel. Bodo and Csato [28] proposed two cluster kernel methods for semi-supervised learning which can be used for different types of data sets: one using hierarchical clustering, and another kernel for reweighting an arbitrary base kernel taking into account the cluster structure

of the data.

### 2.3.4 Support Vector Machines

The Support Vector Machine (SVM) is a state-of-the-art kernel-based technique for classification and regression introduced in 1992 by Boser *et al.* [34]. Based on statistical learning theory and the principle of structural risk minimization [60, 226, 228], SVMs provide better generalization ability than conventional methods such as artificial neural networks which are based on empirical risk minimization [226]. That is, SVMs attempt to minimize the upper bound on the generalization error based on the principle of structural risk minimization rather than minimizing the training error. Due to its good generalization ability, SVMs have been successfully applied in various research areas ranging from image retrieval [219], handwriting recognition [12], text classification [120], molecule classification [152] to gene expression analysis [85].

Consider a training set of  $l$  objects  $\{(x_i, y_i)\}_{i=1}^l \subseteq X \times Y$ , where  $X$  is a pattern space and  $Y$  is a space of class labels. SVMs are kernel machines able to derive a function  $f : X \rightarrow Y$  from the training set which can be used to predict the label of unseen data objects. The basic idea of an SVM is to use a nonlinear mapping to transform the original training data into a higher dimension. Within this new dimension, it tries to separate different classes from the training set by means of hyperplanes. Hyperplanes derived from SVM training are characterized by the property that they are placed in such way that their distance to the closest element of either class is maximal. Such hyperplanes are commonly referred to as *maximum-margin hyperplanes*. Figure 2.8 shows a labeled training set with two classes. The dark blue hyperplane is the maximum-margin hyperplane, i.e., the hyperplane that maximizes the distance to the closest points from both classes (i.e.,  $x_1$  and  $x_2$ ).

In the following, let us consider a binary classification setting with a training set  $\{(x_i, y_i), \dots, (x_l, y_l)\} \subseteq X \times Y$ , where the pattern space is the real vector space ( $X = \mathbb{R}^n$ ) and  $Y = \{-1, 1\}$ . The task is to learn a classifier  $f : X \rightarrow Y$  that predicts the labels of unclassified data objects. Large margin methods try to solve this question by introducing a hyperplane between class  $y = 1$  and class  $y = -1$ . Depending on the location of  $x_i$  with respect to the hyperplane,  $y_i$  is predicted to be 1 or  $-1$ , respectively. Let us assume that there exists such a hyperplane that correctly separates both classes. Then infinitely many of these hyperplanes exist, parameterized by the weight vector  $w \in \mathbb{R}^n$  and the threshold  $b \in \mathbb{R}$ , which can be written as  $\langle w, x \rangle + b = 0$ . Geometrically interpreted,  $w$  is a vector perpendicular to the hyperplane, and  $b$  is a scalar which corresponds to the distance of the hyperplane to the origin of the coordinate system. This distance amounts to  $\frac{|b|}{\|w\|}$ .

These hyperplanes satisfy

$$y_i(\langle w, x_i \rangle + b) > 0 \quad \forall i \in \{1, \dots, l\}, \quad (2.12)$$

corresponding to decision functions

$$f(x) = \text{sign}(\langle w, x \rangle + b), \quad (2.13)$$

where  $f(x)$  is the (predicted) class label of data point  $x$ . Among these hyperplanes a unique optimal hyperplane can be chosen which maximizes the margin (see Figure 2.8), i.e., the minimum distance between the hyperplane and the nearest data points from both classes [227].

Referring to Figure 2.8, implementing a SVM boils down to selecting the variables  $w$  and  $b$  such that the training data can be described by:

$$\langle w, x_i \rangle + b \geq +1 \text{ for } y_i = +1 \quad (2.14)$$

$$\langle w, x_i \rangle + b \leq -1 \text{ for } y_i = -1. \quad (2.15)$$

These equations can be combined into:

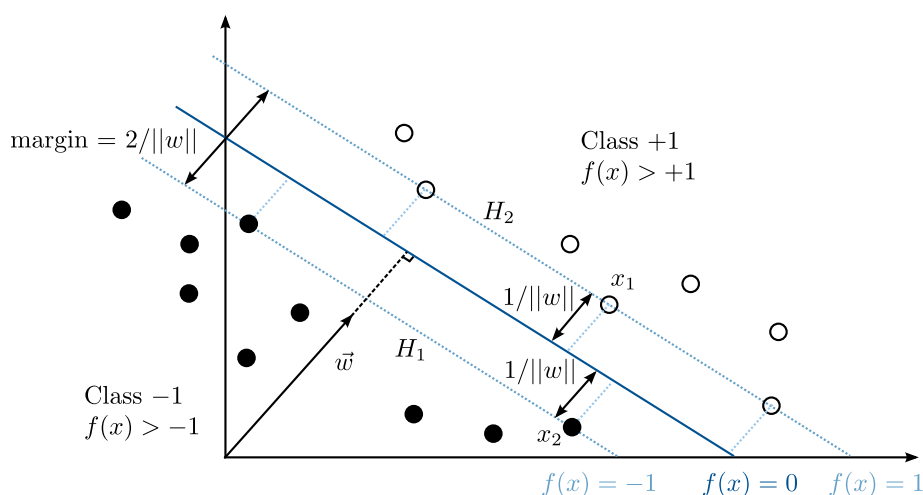
$$y_i(\langle w, x_i \rangle + b) - 1 \geq 0 \quad \forall i \in \{1, \dots, l\}. \quad (2.16)$$

If we now just consider the points that lie closest to the separating hyperplane, i.e., the support vectors, then the two planes  $H_1$  and  $H_2$  that these points lie on can be described by:

$$\langle w, x_i \rangle + b = +1 \text{ for } H_1 \quad (2.17)$$

$$\langle w, x_i \rangle + b = -1 \text{ for } H_2. \quad (2.18)$$

Referring to Figure 2.8, the distance from the separating hyperplane to  $H_1$  is  $\frac{1}{\|w\|}$ . By definition, this is equal to the distance from any point on  $H_2$  to the separating hyperplane.



**Figure 2.8:** Example of a linear classifier separating two classes (filled dots and unfilled dots). The decision surface (in dark blue) is a hyperplane defined by  $\langle w, x_i \rangle + b = 0$ . The margin (dashed line) is defined by the distance of the closest points ( $x_1$  and  $x_2$ ). Data points located on the margin are support vectors.



In order to orientate the hyperplane to be as far from the support vectors as possible, we need to maximize this margin.

### 2.3.4.1 Hard Margin Linear SVMs

For the linearly separable case, finding a maximum separating margin  $d(d = \frac{2}{\|w\|})$  between both classes is a constrained optimization problem (also known as the *primal* formulation) represented by

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2} \|w\|^2 \\ \text{subject to} \quad & y_i(\langle w, x_i \rangle + b) \geq 1, \quad i = 1 \dots l. \end{aligned} \quad (2.19)$$

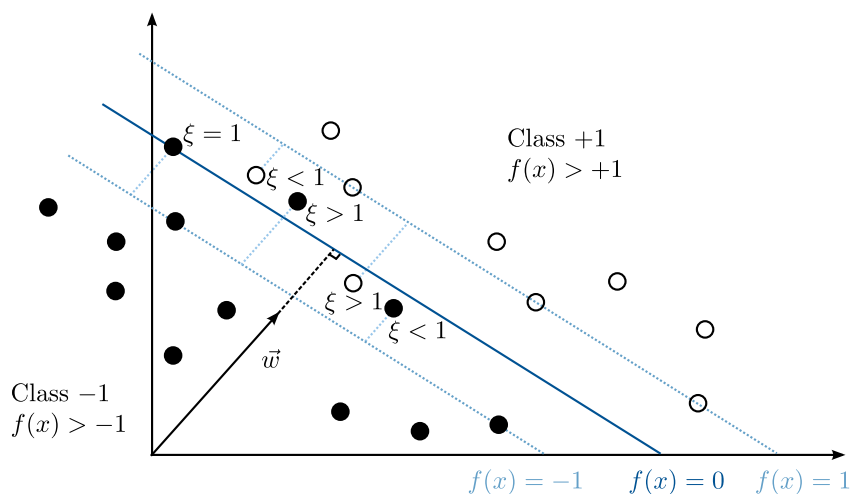
In words, by minimizing  $\frac{1}{2} \|w\|^2$ , the margin between both classes is maximized (see Figure 2.8).

To find an easier solution to this problem it is transformed to its dual [65], by introducing  $\alpha$ , the Lagrange multipliers (dual variables), which are the fundamental unknowns in the dual optimization problem [65]. Hence, the problem in Equation 2.19 becomes:

$$\begin{aligned} \max_{\alpha} \quad & w(\alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l \alpha_i y_i \alpha_j y_j \langle x_i, x_j \rangle \\ \text{subject to} \quad & \alpha_i \geq 0 \quad i = 1 \dots l, \quad \text{and} \quad \sum_{i=1}^l \alpha_i y_i = 0. \end{aligned} \quad (2.20)$$

### 2.3.4.2 Soft Margin Linear SVMs

The formulation described in the previous section is applicable when the data are linearly separable, corresponding to an empirical error of zero. In real world applications, however, data appear under more complex circumstances and might not be separable by a linear



**Figure 2.9:** Illustration of the slack variables  $\xi_i$ , for  $i = 1 \dots n$ . Note that only the values  $\xi_i \neq 0$  are shown, corresponding to points on the wrong side of the margin. All the other points lying either on the margin or on the correct side have  $\xi_i = 0$ .

hyperplane. For this reason, soft-margin SVMs have been developed as an alternative to hard-margin SVMs. While hard-margin SVMs force the condition  $y_i(\langle w, x_i \rangle + b) \geq 1$  to hold, the soft-margin SVMs allow for misclassification of some training points. The goal is to improve the generalization performance of the SVM, i.e., its performance on test samples different from the training set. The soft margin hyperplane can be obtained by relaxing the optimization problem in Equation 2.19, through the introduction of a positive slack variable  $\xi_i$  which is needed in order to allow misclassifications in the set of inequalities [22, 60]. These variables are defined by  $\xi_i = 0$  for points either on the margin or on the correct side of the margin, and  $\xi_i = |y_i - f(x_i)|$  for all other cases. Points with  $\xi_i > 1$  are misclassified because they lie on the wrong side of the decision surface, and those points for which  $0 < \xi_i \leq 1$  lie inside the margin but on the correct side of the decision surface, as illustrated in Figure 2.9.

Including  $\xi_i$ , Equations 2.14 and 2.15 are modified as follows:

$$\langle w, x_i \rangle + b \geq +1 - \xi_i \quad \text{for } y_i = +1 \quad (2.21)$$

$$\langle w, x_i \rangle + b \leq -1 + \xi_i \quad \text{for } y_i = -1, \quad \xi_i \geq 0 \quad \forall i. \quad (2.22)$$

Introducing the regularization parameter  $C$  which controls the trade-off between maximizing the margin and minimizing the training error [65], the optimization problem in Equation 2.19 becomes:

$$\min_{w, b, \xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l \xi_i \quad (2.23)$$

$$\text{subject to } y_i(\langle w, x_i \rangle + b) \geq 1 - \xi_i, \quad \xi_i \geq 0,$$

where  $\sum_{i=1}^l \xi_i$  denotes the upper bound on the training error. The optimization problem in Equation 2.23 is called the *primary problem*. The first part of the objective function tries to maximize the margin between both classes, whereas the second part minimizes the misclassification error.

The dual to Equation 2.23 is:

$$\max_{\alpha} w(\alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l \alpha_i y_i \alpha_j y_j \langle x_i, x_j \rangle \quad (2.24)$$

$$\text{subject to } C \geq \alpha_i \geq 0 \quad i = 1 \dots l, \quad \text{and} \quad \sum_{i=1}^l \alpha_i y_i = 0.$$

Solving for  $\alpha$ , training examples with non-zero  $\alpha$  are called support vectors and the hyperplane is being completely defined by the support vectors only. As shown in Equation 2.24, the  $C$  parameter constitutes the upper bound on  $\alpha_i$ . As a result, three types of support vectors are characterized based on the values of  $\alpha_i$  and  $\xi_i$  as follows (see also Figure 2.9):

- Support vectors with  $\alpha_i < C$ , which lie outside the margin  $d$  and will be correctly classified.
- Support vectors with  $\alpha_i = C$  and  $\xi_i > 1$ , which lie at the wrong side of the hyperplane, and represent errors (will be misclassified points by the hyperplane).
- Support vectors with  $\alpha_i = C$  and  $0 < \xi_i \leq 1$ , which lie inside the margin  $d$  (i.e., closer than  $\frac{1}{\|w\|}$  from the hyperplane).

A disadvantage of the soft margin SVM is that the parameter  $C$  is a rather unintuitive parameter and there is no a priori way to select it. For this reason, an alternative soft margin SVM, the so-called  $\nu$ -SVM, was proposed [200].

### 2.3.4.3 $\nu$ -SVM

The  $\nu$ -SVM was developed to automatically adjust the penalty parameter  $C$  in the original SVM formulation by an alternative parameter,  $\nu \in [0,1]$ , which applies a slightly different penalty and has a more meaningful interpretation. This is because  $\nu$  represents an upper bound on the fraction of training samples which are errors and a lower bound on the fraction of samples which are support vectors [200]. Introducing the parameter  $\nu$ , the soft margin optimization problem is rewritten as:

$$\min_{w,b,\xi} \frac{1}{2} \|w\|^2 - \nu\rho + \frac{1}{l} \sum_{i=1}^l \xi_i \quad (2.25)$$

$$\text{subject to } y_i(\langle w, x_i \rangle + b) \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad \rho \geq 0.$$

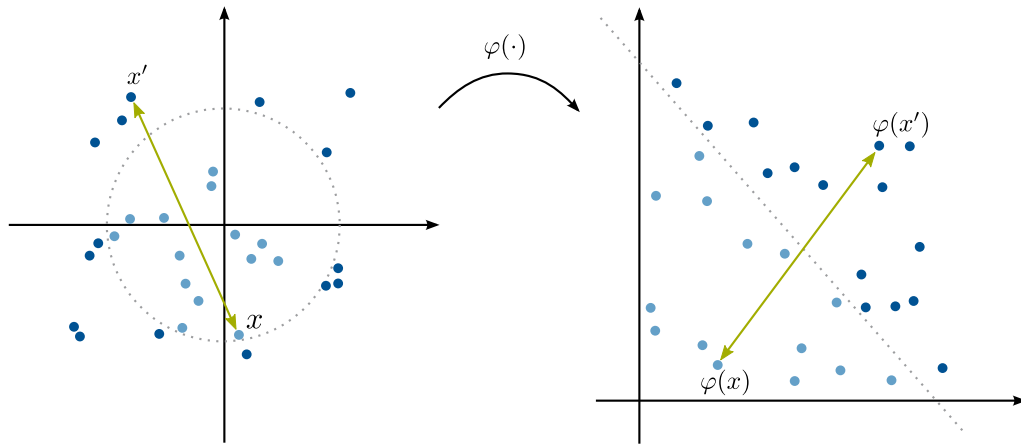
This can be transferred into the corresponding dual:

$$\max_{\alpha} w(\alpha) = -\frac{1}{2} \sum_{i,j=1}^l \alpha_i y_i \alpha_j y_j \langle x_i, x_j \rangle \quad (2.26)$$

$$\text{subject to } 0 \leq \alpha_i \leq \frac{1}{l}, \quad \sum_{i=1}^l \alpha_i \geq \nu, \quad \text{and} \quad \sum_{i=1}^l \alpha_i y_i = 0.$$

### 2.3.4.4 Non-linear SVMs

Still, even soft-margin classifiers cannot solve every classification problem. Consider the following 2-d example in Figure 2.10, where all positive data points lie within a circle and all negative data points lie outside a circle. How to introduce a hyperplane that shows good generalization performance in this case? The trick to overcome these sorts of problems is to map the input space  $x_i \in \mathbb{R}^n$  into a (usually higher dimensional) feature space  $\mathcal{H}$ . The hope is that the non-linearly separable data in the input space become linear in the enlarged space, thus, allowing for linear separation with hyperplanes. The idea is to find a non-linear mapping  $\varphi : \mathbb{R}^n \rightarrow \mathcal{H}$ , such that in  $\mathcal{H}$ , the previous SVM formulation can still be used, simply by replacing  $\langle x_i, x_j \rangle$  with  $\langle \varphi(x_i), \varphi(x_j) \rangle$ . A valid



**Figure 2.10:** Toy example illustrating the kernel trick. Mapping a circle into feature space: data distribution in input space (left) and feature space (right). By transformation from input space  $X$  to feature space  $\mathcal{H}$  by function  $\varphi$ , the light blue and dark blue dots become linearly separable.

kernel function  $k : X \times X \rightarrow \mathbb{R}$  can be defined that enables direct computation of the inner product  $\langle \varphi(x_i), \varphi(x_j) \rangle$  in feature space with the property  $k(x, x') = \langle \varphi(x_i), \varphi(x_j) \rangle$ . The optimization problem now becomes

$$\begin{aligned} \max_{\alpha} w(\alpha) &= \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l \alpha_i y_i \alpha_j y_j k(x_i, x_j) \\ \text{subject to } C &\geq \alpha_i \geq 0 \quad i = 1 \dots l, \quad \text{and} \quad \sum_{i=1}^l \alpha_i y_i = 0. \end{aligned} \quad (2.27)$$

meaning that the classification problem is moved into a higher-dimensional space  $\mathcal{H}$  and solved even without explicitly computing the mapping  $\varphi$  to  $\mathcal{H}$ . This is commonly known as the famous kernel trick (see Section 2.3.1.2).

The decision function of the SVM is given by

$$f(x) = \text{sign} \left( \sum_{s=1}^{sv} \alpha_s y_s k(x_s, x) + b \right), \quad (2.28)$$

where  $sv$  are the model support vectors.

#### 2.3.4.5 SVMs for Regression

SVMs for regression, also called Support Vector Regression (SVR), were mainly developed to introduce the benefit of sparsity into kernel-based regression estimation. For this purpose Vapnik [228] devised the so-called  $\epsilon$ -insensitive loss function. It quantifies the loss incurred by predicting  $f(x)$  instead of  $y$  as

$$L(y, f(x)) = |y - f(x)|_{\epsilon} := \max \{0, |y - f(x)| - \epsilon\} \quad (2.29)$$

The idea is that only training points lying outside a small  $\epsilon$ -tube around the estimated

function contribute to the training error and are penalized in a linear fashion. The goal is to find a regression function  $f(x) = \sum_{i=1}^l \alpha_i k(x, x_i) + b$  with small norm  $\|f\|_{\mathcal{H}}$ , i.e., a function that is as smooth as possible on the one hand and on the other hand leads to a low empirical risk. This corresponds to constructing a linear regression function  $f(x) = \langle w, \varphi(x) \rangle + b$  in feature space  $\mathcal{H}$  with small norm  $\|w\|^2 = \|f\|_{\mathcal{H}}^2$ . Introducing slack variables  $\xi_i, \xi_i^*$  to penalize points that are above or below the  $\epsilon$ -tube, the primal formulation for the regression problem can be formulated as

$$\begin{aligned} \min_{w,b,\xi,\xi^*} \quad & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l (\xi_i + \xi_i^*) \\ \text{subject to} \quad & (\langle w, x_i \rangle + b) - y_i \leq \epsilon + \xi_i \\ & y_i - (\langle w, x_i \rangle + b) \leq \epsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0, \quad i = 1, \dots, l \end{aligned} \quad (2.30)$$

where the constraint in Equation 2.30 provide that the prediction will be close to the regression valued

$$-\epsilon - \xi_i^* \leq (\langle w, x_i \rangle + b) - y_i \leq \epsilon + \xi_i. \quad (2.31)$$

Applying Kuhn-Tucker theory leads to its dual formulation

$$\begin{aligned} \min_{\alpha, \alpha^* \in \mathbb{R}} \quad & \sum_{i=1}^l y_i (\alpha_i^* - \alpha_i) - \epsilon \sum_{i=1}^l (\alpha_i^* + \alpha_i) - \frac{1}{2} \sum_{i,j=1}^l (\alpha_i^* - \alpha_i)' (\alpha_j^* - \alpha_j) k(x_i, x_j) \\ \text{subject to} \quad & 0 \leq \alpha_i, \alpha_i^* \leq C, \quad i = 1 \dots l, \\ & \sum_{i=1}^l (\alpha_i^* - \alpha_i) = 0 \end{aligned} \quad (2.32)$$

The resulting weight vector  $w$  can be written as

$$\sum_{i=1}^l (\alpha_i^* - \alpha_i) \varphi(x). \quad (2.33)$$

Hence the decision function for regression is

$$f(x) = \sum_{i=1}^l (\alpha_i^* - \alpha_i) k(x_i, x) + b. \quad (2.34)$$

## 2.4 Knowledge Extraction from SVMs

SVMs are often considered as state-of-the-art classification techniques in machine learning. Due to their good generalization ability and classification accuracy, they have been widely used in many applications such as bio- and cheminformatics [122, 201]. An important advantage of SVMs is that their classification decision is based on a reduced subset of

training examples, referred to as *support vectors*. However, the main weakness of SVMs is that the generated non-linear models are typically regarded as incomprehensible black box models, i.e., they do not offer an explanation of the classification decisions being made. The opaqueness of SVM models can be remedied through the use of knowledge extraction techniques, which provide some insight into the logics of SVM models. The primary objective of knowledge extraction techniques is to extract the knowledge learned by a black box classifier and represent it in a comprehensible form. Early work on the topic of knowledge extraction mainly focused on the extraction of knowledge from neural networks, particularly in the form of rules [62, 86, 218].

In recent years, the topic of knowledge extraction from SVMs has received considerable attention aiming at opening the black box or making SVMs interpretable. Previous research done in the field of explaining SVM classifications follow different directions. One approach to understand a hypothesis represented by a trained SVM is to try to translate it into a more comprehensible language. Various approaches based on this strategy have been investigated under the header of rule extraction. Rule Extraction methods aim at providing an explanation of the (black box) SVM model by obtaining a set of 'if ... then ... else' rules that approximate the SVM decision boundary. Other methods exist that try to transform the SVM classifier in terms of different basis functions. These methods significantly reduce the number of components needed to describe the classifier. Another direction in the topic of knowledge extraction addresses the task of explaining an SVM by means of visualization. Techniques for visualization of SVMs are based on a dimension reduction of the hypothesis space that is maximally informative about the decision function. Out of the mentioned methods, rule extraction has received the most attention lately. This section provides an overview of various knowledge extraction methods for SVMs with a special focus on rule extraction methods. More specifically, Section 2.4.1 presents related work on the topic of rule extraction from SVMs, while Section 2.4.2 provides an overview of other knowledge extraction methods for SVMs.

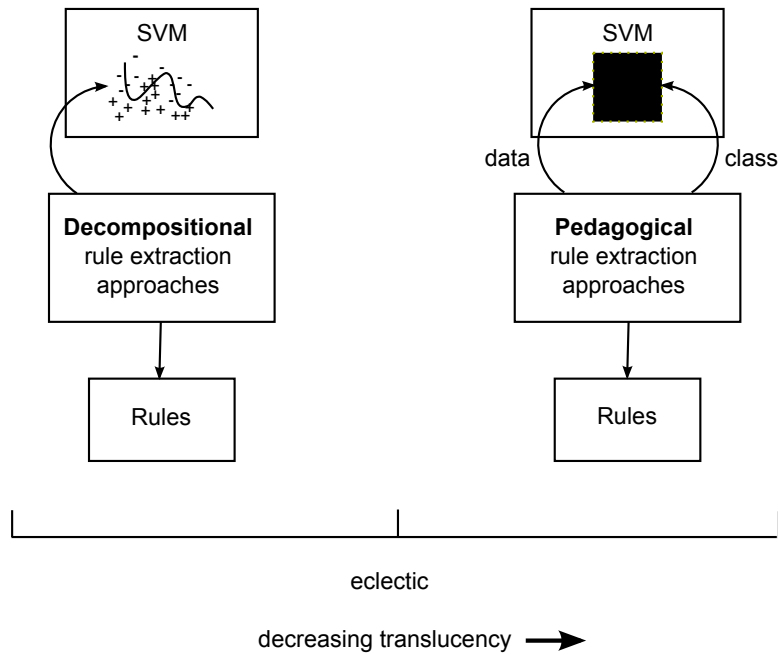
### 2.4.1 Rule Extraction from SVMs

The task of rule extraction from SVMs is to express the knowledge acquired by the SVM during the training process in a comprehensible form which can be easily understood by the end-user. Various methods for rule extraction from SVMs have been proposed in the past which can be classified based on different criteria. One potential way of classifying SVM rule extraction algorithms is in terms of the *translucency* [9, 158]. Translucency refers to the extent to which the internal components of the SVM model are transparent to the rule extraction method. Based on the translucency criteria the rule extraction techniques can be classified into three main categories – decompositional, pedagogical (or learning-based), and eclectic – as suggested by Andrews *et al.* [9] for artificial neural networks. The most translucent approaches are decompositional approaches, while the most opaque ones are pedagogical approaches. Methods that utilize aspects of both approaches are termed

eclectic. Typically, decompositional approaches open the trained SVM model, look into its individual components, i.e., its support vectors, and try to extract rules at the level of these components. In particular, decompositional approaches are closely intertwined with the internal workings of an SVM and its constructed hyperplane and thus typically make use of the support vectors or the decision boundary. Pedagogical approaches, on the other hand, treat the trained model as a black box. Instead of looking at the internal structure, these methods do not make use of the support vectors or the SVM decision boundary, but directly extract rules which relate the inputs and outputs of the SVM. Pedagogical methods typically use the trained SVM model as an oracle to label or classify artificially generated training examples that are later used by a symbolic learning algorithm, such as decision trees, to create a comprehensible classification model. These techniques are motivated by the assumption that the trained model is better able to represent the data than the original data set. Since the model is viewed as a black box, most pedagogical approaches lend themselves very easily to rule extraction from other machine learning algorithms. Hence, it is possible to derive rule extraction techniques from the neural networks domain to the support vector machines domain. The third category are eclectic (i.e., hybrid) approaches, which incorporate elements of both decompositional and pedagogical approaches and have its origin in artificial neural networks. The difference between decompositional, pedagogical and eclectic rule extraction techniques is schematically illustrated in Figure 2.11. Different methods have been proposed to extract rules from SVMs as summarized in the following sections. These methods are classified in terms of the translucency criterion. First, a short description of the proposed decompositional SVM rule extraction techniques is provided, followed by a description of the most commonly used pedagogical and eclectic SVM rule extraction techniques.

#### 2.4.1.1 Decompositional SVM Rule Extraction Techniques

Núñez *et al.* [175] proposed one of the few decompositional approaches for rule extraction from SVMs that was designed specifically for SVMs. The *SVM + prototype* approach uses the output decision function from an SVM and K-means clustering to determine prototype vectors for each class. The approach then uses the prototypes together with the support vectors to determine the boundaries of regions, i.e., ellipsoids and hyperrectangles, defined in the input space by means of geometric methods. These regions are then mapped to two kind of rules: if-then interval rules corresponding to hyperrectangles and equation rules corresponding to ellipsoids. The *SVM + prototype* approach is schematically shown in Figure 2.12. A benefit of the approach is that it produces a small number of rules which are of high accuracy. However, the method has two main limitations. First, the number and the quality of the extracted rules depend upon the initial values of the prototype vectors defining the centers of the clusters, and in turn are affected by the initial parameters of the clustering algorithm. Second, the technique suffers from bad scalability: A large number of training examples and/or input features may result in a large number of clusters



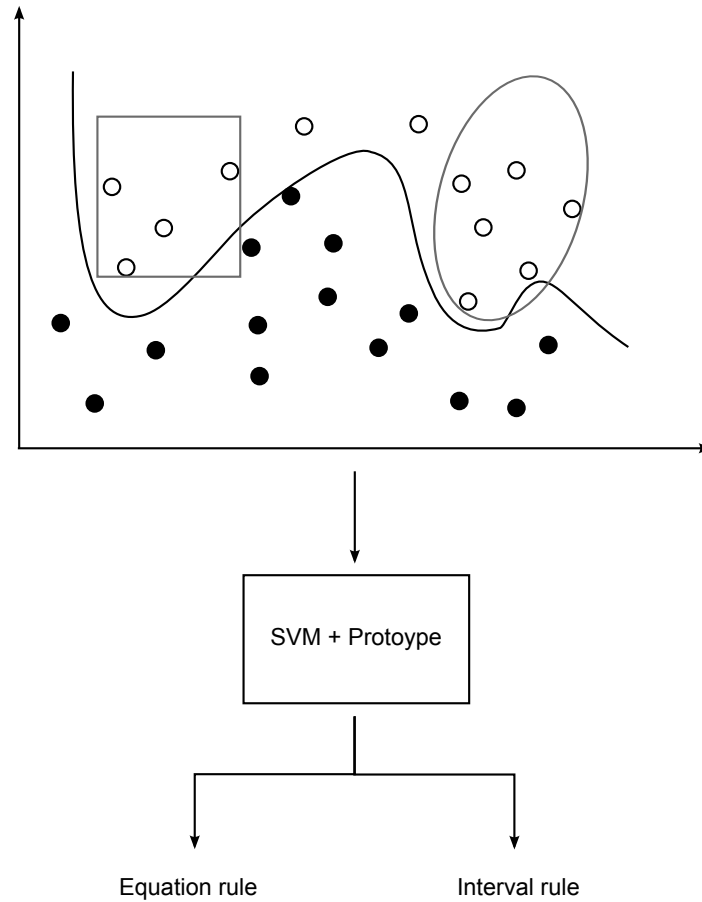
**Figure 2.11:** The transluency criterion for categorizing techniques for extracting knowledge from trained SVMs.

and consequently in a large number of rules, since all the features are present as rule antecedents. As a consequence, the explanation capability may suffer.

In a related study, Zhang *et al.* [256] introduced a hyperrectangle rule extraction algorithm to extract rules from trained SVMs. The approach utilizes the support vector clustering algorithm [20] to find prototype vectors for each class and then uses those vectors together with the support vectors to generate hyperrectangles. A nested generalized exemplar algorithm is utilized to first construct small hyperrectangles around the prototypes, which are then grown incrementally until one of the stopping criteria, which are based on a user-defined minimum confidence threshold or a minimum support threshold, is met. The purpose of the stopping criteria is to control the size of the hyperrectangles and hence the quality of the rules generated. If-then rules are then generated by projecting these hyperrectangles onto coordinate axes. The experimental results demonstrate the high accuracy of the hyperrectangle rule extraction algorithm. However, similar to the approach of Núñez *et al.* [175], one limitation of this approach is the low comprehensibility of the rules as all the input features appear as rule antecedents. Another limitation is the difficulty of selecting the parameters of the support vector clustering algorithm which are critical in defining the number of clusters and hence the number of rules [256].

Pursuing the same basic idea of generating hyperrectangle rules, Fu *et al.* [83] suggested another method for rule extraction from non-linear SVMs trained with a radial basis function (RBF) kernel function. Similar to the previous approaches, the method, called *RuleExSVM*, utilizes the SVM decision boundary in addition to the support vec-





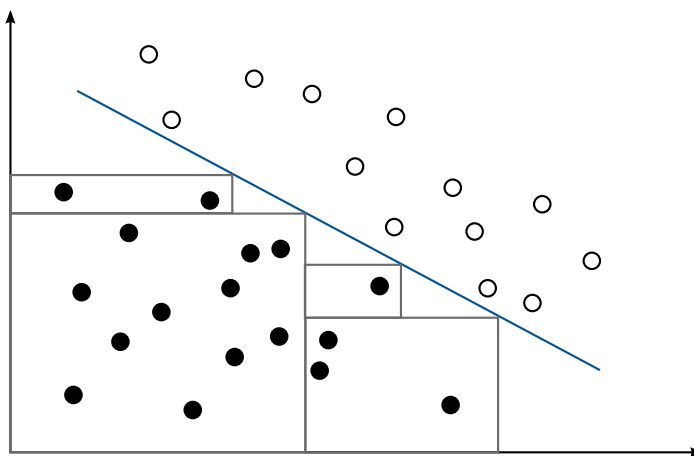
**Figure 2.12:** Example illustrating the approach by Núñez *et al.* [175].

tors. RuleExSVM proceeds in three phases: initial, tuning, and pruning. In the initial phase, hyperrectangles are found whose upper and lower corners are defined by finding the intersections of lines extended from each of the support vectors with the SVM decision boundary. In the tuning phase, outliers from the other class are excluded by chopping each hyperrectangle. Finally, in the pruning phase, overlapping hyperrectangles (representing redundant rules) are removed. The method suffers from several limitations. First, it is only valid for rule extraction from SVMs with nonlinear RBF kernel functions. Even though the authors argue that this method could be easily extended to other types of kernels, they do not provide a framework for this. Second, rather than providing a method to control the overlap between hyperrectangles at the rule generation phase, the approach removes them later at the pruning phase. Third, the algorithm constructs the hyperrectangles based on the number of support vectors.

In another study, Fung *et al.* [84] proposed an algorithm for converting linear SVMs and any other arbitrary hyperplane-based linear classifiers into a set of rules. The rule extraction technique is similar to the SVM + prototype approach [175], but does not include computationally expensive clustering. Instead, the algorithm transforms the problem to a simpler, equivalent variant and constructs hypercubes by solving multiple constrained optimization problems. Each hypercube, which is a subset of one of the bounded regions,

is then transformed to a rule. In order to obtain disjoint rules, each hypercube is required to have one vertex that lies on the separating hyperplane (as illustrated by Figure 2.13). Employing a depth-first search, the rule extraction algorithm iterates over the training data in each half-space to find rules for the examples which have not been covered by any previous rule. The authors presented two variants of their algorithm based on different criteria for selecting “optimal rules”. The method is considered to be decompositional, as it is only applicable when the underlying model provides a linear decision boundary. One drawback of the approach is that it produces a relatively large number of rules, thus the comprehensibility of the rules is low. The generalization performance and the quality of rules are not tested by the authors. However, since the rules extracted from the generated hypercubes cover the training data, they may not provide an explanation for new unseen data.

Chen *et al.* [51] proposed a rule extraction algorithm for gene expression data to improve the comprehensibility of SVMs. The approach, named *MK-SVMII*, constitutes one component of a multiple kernel SVM (MK-SVM) scheme, comprising feature selection, prediction modeling and rule extraction. In the feature selection module, a new single feature kernel is defined by transforming the computationally expensive feature selection problem into the problem of finding sparse feature coefficients representing the weight of a single feature kernel. Features with zero coefficients have no impact on the output of SVM and can be discarded. Using the MK-SVMII Kernel, and making some substitutions into the ordinary SVM quadratic programming formulation, a new mixture coefficient is introduced. If an input vector has at least one non-zero mixture coefficient, the corresponding input data vector becomes a support vector. In this case, the Lagrange multiplier in the ordinary SVM formulation is replaced by the mixture coefficient in MK-SVMII. The rule extraction method proposed for MK-SVMII is similar to the one addressed by Fung *et al.* [84] which solves multiple optimization problems to find rules that describe non-empty hypercubes in each class half-space. However, in MK-SVMII, support vectors are used as



**Figure 2.13:** Example illustrating the approach by Fung *et al.* [84]. The non-overlapping rules covering the half-space are represented as rectangles.

vertices of hypercubes, where a series of hypercubes approximates the subspace of each class. The rules extracted by this method show good generalization capacity and good comprehensibility. At the same time, the approach yields compact gene subsets which were found to be useful in defining possible pathways of genetic networks [51].

Zhang *et al.* [255] suggested a different approach for rule extraction from SVMs trained with a special Disjunctive Normal Form (DNF) Boolean kernel. The approach, called *DRC-BK*, is based on the idea that an SVM trained with a Boolean kernel finds the optimal separating hyperplane by learning Boolean functions in a higher dimensional feature space. The authors have made an initial proposition that if each dimension of the input space can be regarded a Boolean literal, then each dimension in the feature space can be regarded as a conjunction of several of these Boolean literals. In the rule generation process, only conjunctions with a significant contribution are considered. The authors reported that DRC-BK generates rules of high accuracy. However, a drawback of the approach is that it generates a large number of rules, resulting in lower comprehension. Another limitation of the method is that the preprocessing step leads to an increase of the number of the features that may appear as a rule antecedent, which makes the comprehensibility even worse.

Barakat and Bradley [18] proposed an algorithm for rule extraction from SVMs, termed *SQRex-SVM*. The approach extracts rules directly from a subset of the support vectors of a trained SVM using a modified sequential covering algorithm [170]. Rules are generated based on an ordered search of the most discriminative features, as measured by inter-class separation. The procedure is limited to binary classification problems. The experimental results demonstrate that the rules produced by SQRex-SVM exhibit both good generalization performance and comprehensibility.

Another method belonging to the class of decomposition SVM rule extraction methods is the method proposed by Chaves *et al.* [66] for extracting fuzzy rules from trained SVMs. The main idea of the approach is to project each feature in each of the support vectors along its coordinate axes, forming a number of fuzzy sets with equal domain size for each coordinate [130]. Next, the fuzzy membership degree of each fuzzy set is computed and each of the support vectors is assigned to the fuzzy set with the highest membership degree. Finally, from each support vector a fuzzy rule is generated. The method has several limitations. First, it may not be suited for binary and categorical attributes, since in these cases the evaluation of membership degree value is non-trivial. Second, the rules extracted by this approach are reported to be of poor quality (the best results had 53,2% in accuracy). However, the authors argue that the objective of the proposed method is to extract interpretable knowledge from a trained SVM, not to improve the SVM performance. Third, the comprehensibility of the extracted rule set is low due to the large number of the extracted rules and due to all the attributes appearing as antecedents.

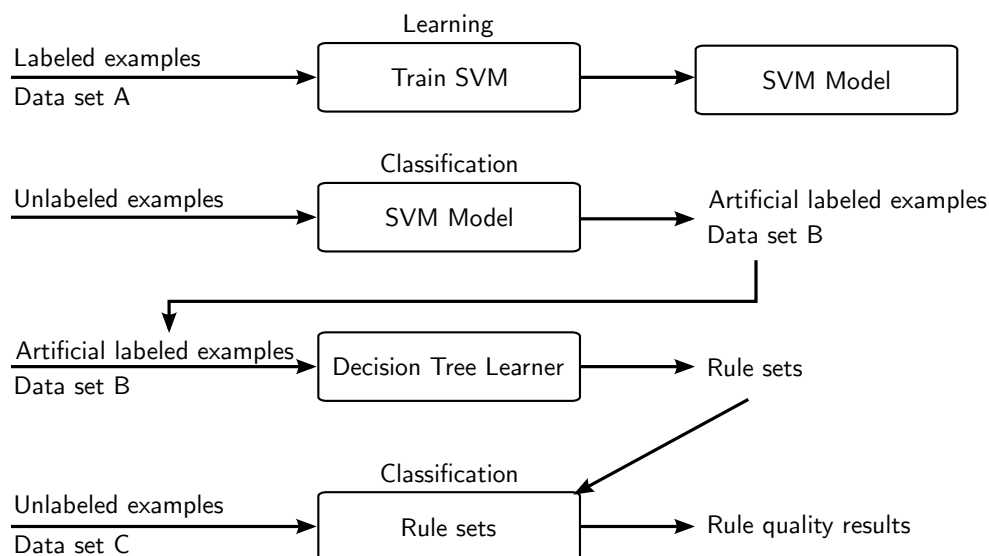
The method proposed by Martens *et al.* [157] is similar to the pedagogical methods which will be described in Section 2.4.1.2, but has an additional step which generates additional training examples close to the support vectors. The generated examples are then

used with the training data, all provided with a class label by the trained SVM model, to train different decision tree algorithms that learn what the SVMs have learned. Since it incorporates elements of both the pedagogical (such as using the black box also as an oracle) and decompositional approaches, it could be categorized as an eclectic approach. However, the authors argue that a decompositional approach with elements of a pedagogical approach still remains decompositional. According to them, their approach uses the SVM as an oracle, but at the same time explicitly uses concepts linked to SVMs, i.e., the support vectors, and is therefore considered a decompositional approach.

#### 2.4.1.2 Pedagogical SVM Rule Extraction Techniques

The pedagogical method suggested by Barakat and Diederich [16] primarily uses a decision tree learner to effectively learn what the SVM has learned. As all pedagogical approaches, the method treats the SVM as a black box. The basic idea of the approach is to create artificially labeled examples, whose target classes are replaced with the SVM predicted classes, which, in turn, represent the knowledge learned by the SVM. The approach then uses this data set to train a machine learning technique with explanation capability (decision tree learners) [177]. As a result, rules are extracted that represent both the concepts learned by the SVM and its generalization behavior. The steps involved in the pedagogical SVM rule extraction approach by Barakat and Diederich [16] are shown in Figure 2.14.

Huysmans *et al.* [111] proposed an algorithm called *Iter* that uses a sequential covering approach, entailing the learning of one rule for one class at the time. The method works by an iterative expansion of hypercubes, whereby each hypercube represents a propositional rule. More precisely, the algorithm starts with the creation of a user-defined number of random starting cubes which are infinitesimal and correspond to points in the input space. These initial cubes are gradually expanded, with randomly distributed extra generated



**Figure 2.14:** Pedagogical rule extraction approach by Barakat and Diederich [16]. Adapted from [18].

data instances provided with a class label by the trained SVM model, until they cover the entire input space or until they can no longer be expanded. The algorithm was designed to build predictive regression rules from a trained SVM regression model. With minor adaptations it is suitable for classification problems as well.

Martens *et al.* [158] introduced two rule extraction techniques for SVM, *Trepan* [63, 64] and *G-REX* (Genetic Rule EXtraction) [121]. The approaches are taken from the artificial neural networks domain, but can be easily applied for rule extraction from SVM. *Trepan* was first introduced by Craven and Shavlik [63]. Using a best-first expansion strategy, the approach grows a tree by recursive partitioning. Given a trained SVM, *Trepan* first relabels the training examples according to the classifications made by the model in order to mimic the behavior of the SVM black box model. The relabeled training data are then used to initiate the decision tree growing process. *Trepan* can also enrich the training data by generating artificial data automatically, using the trained SVM to label the new instances. The SVM model is thus used as an oracle to answer class membership queries about artificially generated data points. In case the number of training data points in a node is less than a user defined parameter, *Trepan* additionally generates data point which are labeled by the SVM model. This process is often referred to as *active learning*. Hence, in this way *Trepan* can overcome the limitation of decision-tree-induction algorithms which typically suffer from having fewer and fewer training observations available for deciding upon the splits or leaf node class labels at lower levels of the tree.

The G-REX algorithm [121] for knowledge extraction from artificial neural networks was also modified to handle SVMs. The method uses genetic programming [136] to evolve an optimal set of rules. The information extracted using this method can be represented using different types of rules such as Boolean rules, decision trees, M-of-N type rules, or fuzzy rules. The experimental results by Martens *et al.* [158] show that the SVM rule extraction techniques lose only a small percentage in performance compared to SVMs and therefore rank at the top of comprehensible classification techniques.

Ren and Garcez [187] proposed a rule extraction algorithm which uses the points on the SVM classification boundary and synthetic training examples to construct a set of optimized hypercube rules without considering the inner structure and the support vectors. The approach maximizes the area covered by the extracted rules and, thus, approximates the SVM. In more detail, given input vectors  $x_i$ , the approach queries an SVM to obtain the classification for those input vectors. After querying, clustering is employed on those input vectors  $x_i$  with the same  $y_i$ , in order to group them into a set of clusters. Subsequently, the approach looks for the points  $P$  that lie on the SVM classification boundaries by means of a binary search algorithm. For both the points in  $P$  and the synthetic training instances, an initial optimal rule set can be extracted by solving an optimization problem attempting to find the largest consistent hypercubes in the input space. Finally, the approach applies several post-processing measures to the initial rule set to construct a smaller number of non-overlapping rules with high coverage and generalization rate. An advantage of the algorithm is that it is neither restricted to a specific SVM classifier, nor does it depend on

the availability of specific training sets for rule extraction.

### 2.4.1.3 Eclectic SVM Rule Extraction Techniques

Following the translucency dimension, Barakat and Diederich [17] introduced an eclectic rule extraction approach that utilizes the knowledge acquired by an SVM and represented in its support vectors and uses a decision tree learner to learn what the SVM has learned. The algorithm steps are similar to the authors' previously proposed pedagogical rule extraction approach [16], but instead of using all the training data the model support vectors are used to generate the artificial labeled examples, which are then used to train a decision tree for extracting the rules. The experimental results demonstrate that the approach extracts comprehensible rules with a high degree of accuracy and fidelity. However, a limitation of the approach is its sensitivity to the noise in the training data, as the rules are extracted from all types of the SVM model support vectors. The approach [17] fits well into the eclectic rule-extraction algorithm category, as it elects the patterns that have influence in defining the separating hyperplane and it has also a pedagogical component.

Another eclectic rule extraction approach was proposed by Barakat and Bradley [15]. The approach employs the eclectic rule extraction approach described in the work by Barakat and Diederich [17] and uses the area under the receiver operating characteristics (AUROC) to assess the quality of rules extracted from an SVM.

### 2.4.2 Other Knowledge Extraction Methods from SVMs

Besides rule extraction techniques, other techniques for extracting knowledge from SVMs exist. In work by Bakir *et al.* [13], a general learning-based framework for finding *pre-images* was proposed. The pre-image problem consists of finding a reverse mapping from feature space back to input space. More formally, given a positive definite kernel function  $k$ , which computes the dot product of pairs of members  $x, x'$  of an input space  $X$ . The kernel induces some reproducing kernel Hilbert space  $\mathcal{H}_k$ , called the feature space, and a mapping  $\varphi : X \rightarrow \mathcal{H}_k$ , such that  $k(x, x') = \langle \varphi(x), \varphi(x') \rangle$ . Given the feature space representation  $\psi = \varphi(x^*)$  ( $\psi \in \mathcal{H}_k$ ) of a desired output  $x^*$ , the pre-image problem consists of finding  $x^* \in X$ . In their work, the authors seek to learn a function  $\Gamma : \mathcal{H}_k \rightarrow X$ , to compute the pre-image, such that, approximately,  $\Gamma(\varphi(x)) = x$ . However, this is often not possible, since  $\mathcal{H}_k$  is usually a far larger space than  $X$ . The trick employed by Bakir *et al.* is to use a finite-dimensional basis in the feature space  $\mathcal{H}_k$  and to work on the basis coordinates instead of the possibly infinite original space. It is based on the observation that a finite set of points  $x_i$  always spans only a finite dimensional subspace of  $\mathcal{H}_k$ . The problem of estimating a pre-image function  $\Gamma' : \mathbb{R}^n \rightarrow X$  can then be reduced to a standard regression problem. The basis itself can be found by means of kernel Principal Component Analysis with the kernel induced by  $\varphi$ . The pre-image problem, illustrated in Figure 2.15, has a wide range of applications in kernel methods, such as for reduced set methods [44] and for Kernel Dependency Estimation [239], which aims at finding a

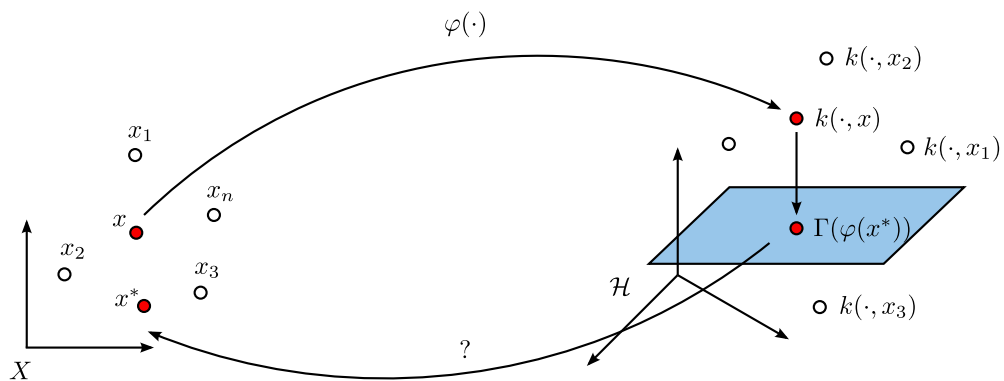
mapping between paired sets of objects.

In subsequent work, Bakir *et al.* considered the pre-image problem in the domain of graphs [14]. More precisely, the authors take the input space  $\mathcal{G}$  to be the set of node-labeled graphs, and assume the kernel  $k$  to be the marginalized graph kernel introduced by Kashima *et al.* [127]. For this setting, an algorithm to approximate the pre-image of a graph  $g^* \in \mathcal{G}$  of a point  $\psi$  in feature space  $\mathcal{H}_k$  is proposed.

Weston *et al.* [239] considered the task of learning dependencies between a general class of objects. Their approach, referred to as Kernel Dependency Estimation, uses kernel Principal Component Analysis to implicitly model correlations among both inputs and outputs. More specifically, Kernel Dependency Estimation decouples output correlations by first applying kernel Principal Component Analysis over the outputs and then learning the mappings from the input space to the dimension reduced space by ridge regression. In order to recover the output in the original representation a pre-image calculation is required.

Three other families of methods for understanding classifiers such as SVMs are sensitivity analysis, inverse classification and borderline classification. Sensitivity analysis determines how much a change in the input of a model will affect the output of the model. This analysis may, for instance, be useful to determine which input features are most important to obtain accurate output values [247]. Inverse classification methods take a completely different approach. Instead of trying to explain a model, inverse classification describes how one point can be moved into another classification. More precisely, the inverse classification problem consists of determining the required changes to attribute values to reclassify a member of one class as a member of a different desired class [156].

Barbella *et al.* [19] introduced two techniques for providing insight into local classifications obtained by an SVM on continuous data. Both techniques have in common that they explain the model on the local level, that is, for an individual test point. The first technique involves finding the support vectors that contribute the strongest to the classification of a particular test point. The second technique is a variation of the inverse classification technique, called *Borderline Classification*. The goal of inverse classification



**Figure 2.15:** Illustration of the pre-image problem in kernel-based machines. Adapted from Honeine and Richard [105].

is to determine which features of the test point would need to be modified to cause a switch in classification. Borderline Classification differs from inverse classification in the sense that instead of switching the point to another class, it determines the locally minimal change required to place the test point on the separating surface between the two classes. Both techniques add explainability to the results of an SVM classifier.

Subianto and Siebes [215] proposed two concepts that explain arbitrary classifiers defined on discrete data both at the local level, i.e., for an individual data point and at the global level, i.e., for the entire model. The first approach provides insight into why a classifier classifies a data point as it does and, thus, ensures local understandability. More precisely, local explanations are defined as “a minimal set of attributes, such that there exists a change of values for the attributes in that set that would change the assigned class” [215]. This approach is similar to the idea of inverse classification as well as to borderline classification. The key difference is that the approach by Subianto and Siebes only works on discrete-valued data sets and involves finding a nearest point in the opposite class. On the other hand, borderline classification is defined for continuous-valued features and finds a point on the separating surface, not on the opposite side. Further, the authors describe a technique for explaining arbitrary classifiers at a global level by introducing attribute weights as a global measure of the importance of an attribute for a given classifier. The higher the weight of an attribute, the more often it is decisive in the classification of a data point. The approach is not restricted to SVMs, but can be used for any classifier. However, it is limited by its restriction to discrete data.



# CHAPTER 3

---

## Graph Clustering

---

Graph clustering has become an important and active topic in data mining, to detect groups of similar instances while at the same time highlighting differences between dissimilar ones. The goal of graph clustering is to partition instances in a graph database into different clusters based on various criteria such as vertex connectivity, neighborhood similarity or the size of the maximum common subgraph. This can serve to structure the graph space and to improve the understanding of the data. Traditional graph clustering approaches ignore the structure in the graph data and transform the graphs into a feature vector-based representation [164, 250]. These techniques have the advantage of being highly efficient, but at the same time imply a loss of information with respect to the graph topology. On the other hand, more sophisticated graph clustering approaches are directly based on the structure of the graphs. These techniques have the desirable property that the calculated similarity measure is intuitive and can be visualized easily. However, they suffer from efficiency and scalability problems with respect to large graph data sets.

This chapter addresses the problem of clustering large graph databases according to scaffolds (i.e., large structural overlaps) that are shared between cluster members and presents three novel methods for performing that task with a special focus on scalability. In contrast to many related approaches, the methods do not rely on computationally expensive maximum common subgraph (MCS) operations or variants thereof, but on frequent subgraph mining, i.e., graph clustering without generating features or decomposing graphs into parts.

The chapter starts by introducing an online algorithm for the task of graph clustering that produces overlapping (non-disjoint) and non-exhaustive clusterings. In the proposed clustering approach, clusters encompass all graphs that share a sufficiently large common subgraph. The size of the common subgraph of a graph in a cluster has to take at least a user-specified fraction of its overall size. Section 3.2 presents an extension of this approach, a highly parallelized graph clustering method, called PSCG, that takes advantage of high-performance parallel hardware and further improves the algorithm in several ways. Finally, in Section 3.3, a scalable graph clustering approach is proposed that can cluster even larger graph databases. The approach, named SCAP (Structural Clustering by Abstract Pre-

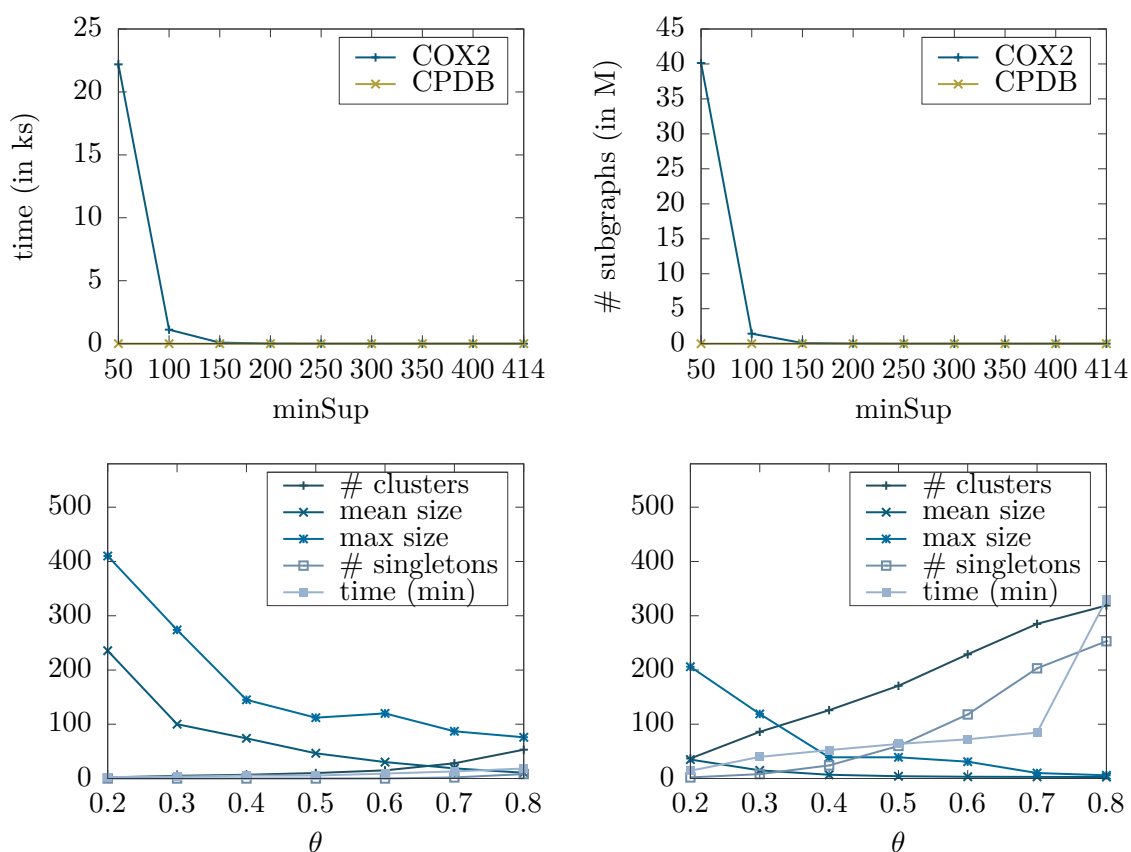
clustering), employs two clustering stages, an initial partitioning step to partition the original data set into several smaller data sets and a further step of refinement to split these coarse-grained results into more accurate clusters.

### 3.1 Structural Graph Clustering

Mining graph data has attracted a lot of attention in the past years [112, 224, 245]. One family of methods is concerned with mining subgraph patterns in graph databases [112, 245]. The criteria for interestingness are often based on the support of a pattern in the graph database, e.g., requiring a minimum and/or maximum frequency, closedness, freeness or class-correlation. However, in all of these cases, the structural diversity of graph databases, i.e., the existence of groups of similar or dissimilar graphs, is not explicitly taken into account or revealed by the algorithm. Vice versa, the structural composition and existence of groups of similar graphs have a serious impact on the output and runtime performance of pattern mining algorithms. To gain insights into the structural characteristics of graph data sets, a graph clustering algorithm was developed that discovers groups of structurally similar and dissimilar graphs. The algorithm can be practically useful for a variety of purposes, such as for benchmarking other graph mining algorithms, for descriptor calculation (e.g., for QSAR studies), for computing local models for classification or regression (e.g., one per cluster) and for calculation of the so-called applicability domain of models.

To illustrate the impact of structural diversity on graph mining results, two data sets of molecular graphs of the same size and with approximately the same number of atoms per molecule are considered. The first data set, COX2 [216], contains 414 compounds, which possess a relatively high structural homogeneity. The second data set is a subset of the Carcinogenic Potency Database (CPDB) [92] that matches the COX2 data both in the number of structures and the number of atoms per structure. The results of a typical graph mining representative, gSpan [245], and the results of the graph clustering algorithm presented in this section are shown in Figure 3.1. In the upper part of the figure, the huge difference in the runtime and the number of discovered patterns can be seen. For structurally homogeneous data (COX2), the number of patterns and runtime explodes, whereas for structurally heterogeneous data (CPDB) the algorithm behaves as expected. The reason for this difference in performance becomes evident in the graph clustering results in the lower part of Figure 3.1. As can be seen, there is a small number of large clusters in COX2 and a large number of small clusters in CPDB (for each value of a parameter that is varied on the x-axis). This indicates a high degree of structural homogeneity in COX2 and a low degree in CPDB, and also hints at the usefulness of graph clustering to make the characteristics of a graph database explicit.

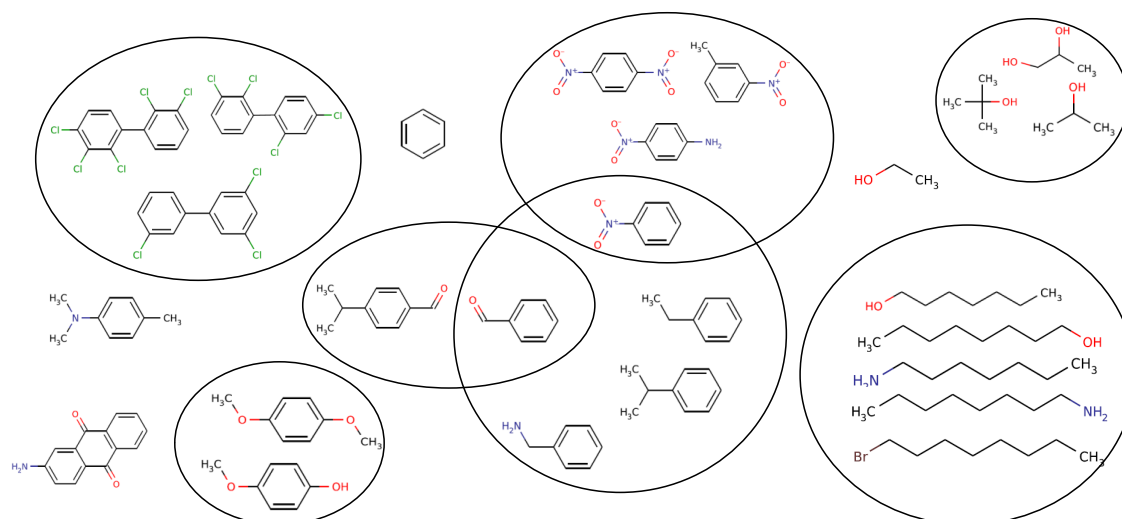
The graph clustering algorithm presented in this section operates directly on the graphs, i.e., it does not require the computation of features or the decomposition into subgraphs. It works online (processing one graph after the other) and creates a non-disjoint and non-



**Figure 3.1:** (Above) Results of gSpan: Runtime behavior (left) and number of subgraphs (right) on COX2 and CPDB. (Below) Results of structural clustering on COX2 (left) and CPDB (right).

exhaustive clustering: graphs are allowed to belong to several clusters or no cluster at all. One important component of the algorithm is a variant of gSpan [245] to determine cluster membership. Thus, the proposed graph clustering approach is based on a practically fast graph mining algorithm and not on typically time-consuming maximum common subgraph (MCS) operations [213]. In contrast to another graph clustering approach based on graph pattern mining [225], the (often quite numerous) frequent subgraphs are just by-products, and not part of the output of the algorithm: the actual output consists just of the clustered graphs sharing a common scaffold. Figure 3.2 provides a sample output of the proposed structural clustering approach on a small subset of molecular graphs.

The remainder of the section is organized as follows. In Section 3.1.1 and 3.1.2 the methodology of the structure-based clustering algorithm is introduced. Section 3.1.3 presents a description of the data sets and experiments as well as an interpretation of the results. Section 3.1.4 gives a conclusion and an outlook to future work.

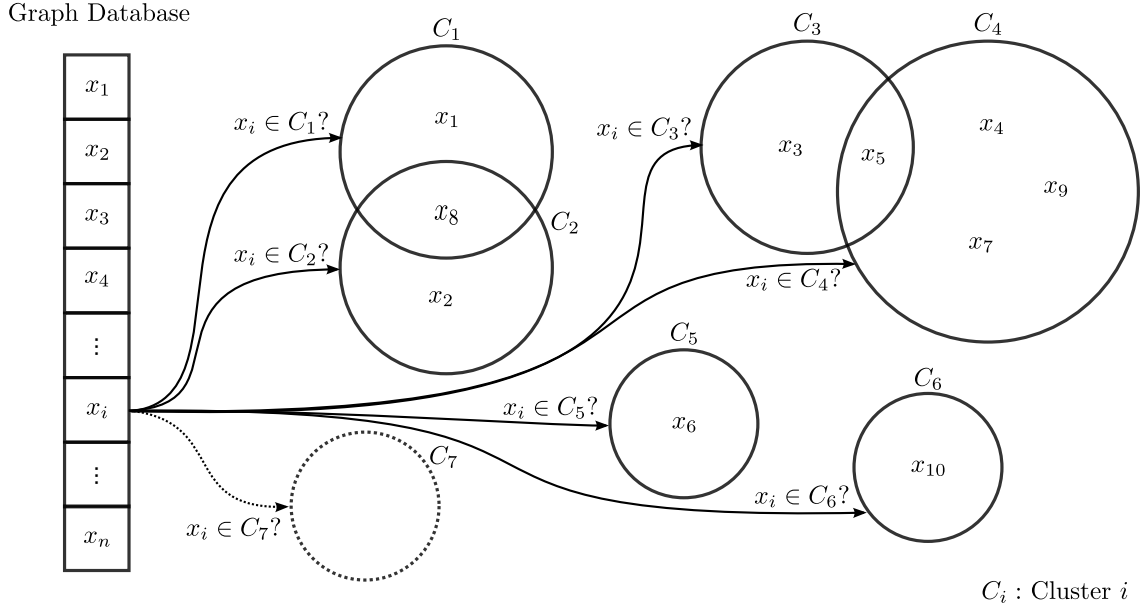


**Figure 3.2:** Sample output from structural clustering. Atoms correspond to labeled vertices, bonds to edges. Vertices without atom labels represent carbon (C) atoms. Only heavy atoms are considered, i.e., hydrogen atoms (H) are ignored. The figure distinguishes two bond types: Single bonds and double bonds.

### 3.1.1 Problem Definition

Structural clustering is the problem of finding groups of graphs sharing some structural similarity. Instances with similar graph structures are expected to be in the same cluster provided that the common subgraphs match to a satisfactory extent. Only connected subgraphs are considered as common subgraphs. The similarity between graphs is defined with respect to some user-defined size threshold. The threshold is set such that the common subgraphs shared among a query graph and all cluster instances make up a specific proportion of the size of each graph. A graph is assigned to a cluster provided that there exists at least one such common subgraph whose size is equal or bigger than the threshold. In this way, a graph instance can simultaneously belong to multiple clusters (overlapping clustering) if the size of at least one common subgraph with these clusters is equal or bigger than the threshold. If a graph instance does not share a common subgraph with any cluster that meets the threshold, this instance is not included in any cluster (non-exhaustive clustering). A graphical overview is shown in Figure 3.3. For one graph after the other, it is decided whether it belongs to an existing cluster or whether a new cluster is created.

Formally, the problem of structural clustering is framed as follows. Given a set of graph instances  $X = \{x_1, \dots, x_n\}$ , the task is to assign the instances to a set of clusters which may overlap with each other. In clustering these graph instances, one objective is considered: to maximize the average number of instances contained in a cluster, such that at any time for each cluster  $C$  there exists at least one common subgraph that makes up a specific proportion,  $\theta$ , of the size of each cluster member. Considering the state of a



**Figure 3.3:** Schematic overview of the cluster membership assignment for instance  $x_i$ . Graph instances are represented by  $x_1, \dots, x_n$ , clusters by  $C_1, \dots, C_k$ .

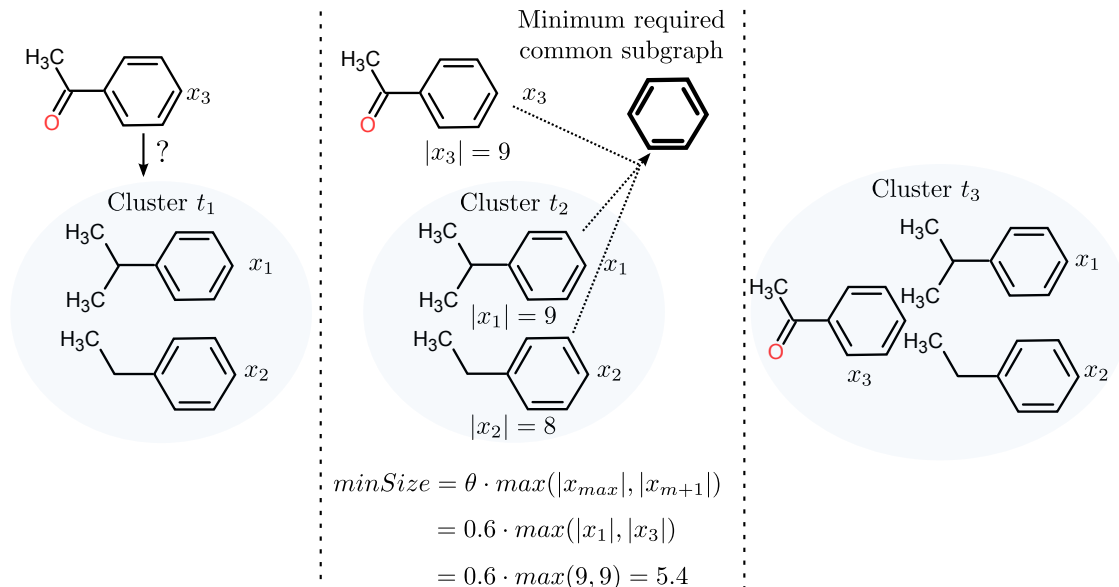
cluster  $C = \{x_1, \dots, x_m\}^2$  at any point in time, the criterion can formally be defined as:

$$\exists s \in cs(\{x_1, \dots, x_m\}) \forall x_i \in C : |s| \geq \theta |x_i| \quad (3.1)$$

where  $s$  is a subgraph and  $\theta \in [0,1]$  is a user-defined similarity coefficient. According to this goal, a minimum threshold for the size of the common subgraphs shared by the query graph  $x_{m+1}$  and the instances in cluster  $C$  can be defined as

$$minSize = \theta \max(|x_{max}|, |x_{m+1}|), \quad (3.2)$$

where  $\theta \in [0,1]$  and  $x_{max}$  is the largest graph instance in the cluster. Figure 3.4 shows an example of the cluster assignment step of the structural clustering approach. To obtain meaningful and interpretable results, the minimum size of a graph considered for cluster membership is further constrained by a *minGraphSize* threshold. Only graphs whose size is greater than *minGraphSize* are considered for clustering. Thus, the identification of the general cluster scaffold will not be impeded by the presence of a few graph structures whose scaffold is much smaller than the one the majority of the cluster members share. This will be especially useful in real-world applications that often contain small fragments (see the minimum size column in Table 3.1).



**Figure 3.4:** Example illustrating the cluster assignment step of the proposed structural clustering approach for  $\theta = 0.6$ . The figure shows the clustering state at different time steps. To be assigned to the cluster, query instance  $x_3$  needs to share at least one common subgraph with the cluster members  $x_1$  and  $x_2$  that meets the  $\text{minSize}$  threshold defined in Equation 3.2. As  $x_3$  shares such a subgraph with  $x_1$  and  $x_2$  that meets the  $\text{minSize}$  threshold of 5.4,  $x_3$  is assigned to the cluster.

### 3.1.2 Method

The proposed structural graph clustering algorithm works as follows. Let  $\text{minGraphSize}$  be the minimum threshold for the graph size and  $\text{minSize}$  be the minimum threshold for the size of the common subgraphs specified by the user and defined in Equation 3.2. In the first step, an initial cluster is created containing the first graph instance that is larger than  $\text{minGraphSize}$ . In the following steps, each graph is compared against all existing clusters. In case the query instance meets the  $\text{minGraphSize}$  threshold and shares at least one common subgraph with one or more clusters that meets the cluster criterion in Equation 3.2, the instance is added to the respective cluster. Unlike many traditional clustering algorithms, a graph is allowed to belong to no cluster, since it is possible that a graph is not similar to any cluster. Thus, in this case, a new singleton cluster is created containing the query instance. The proposed clustering algorithm has two main advantages over many clustering algorithms. First, the algorithm works in an online mode, since it does not keep all the examples in memory at the same time, but processes them one by one in a single pass. Second, in contrast to many clustering algorithms which assume that the number of clusters is known beforehand, the proposed algorithm does not require the specification of the number of clusters a priori. The pseudocode for the structural clustering algorithm is shown in Algorithm 1.

For computing common subgraphs, a modified version of the graph mining algorithm

<sup>2</sup> In slight abuse of notation, the same indices are used as above.

**Algorithm 1** Structural Clustering

---

**Input:** *graphs* - queue of  $n$  graphs to be clustered  
 $\theta$  - similarity threshold ( $\theta \in [0,1]$ )  
*minGraphSize* - minimum graph size

**Output:** *clusters* - resulting clusters

```

1: procedure SC(graphs,  $\theta$ , minGraphSize)
2:   clusters  $\leftarrow \emptyset$ 
3:    $C = \text{newCluster}(\text{dequeue}(\textit{graphs}))$ 
4:   clusters.add( $C$ )
5:   for all  $g \in \textit{graphs}$  do ▷ loop over all graphs
6:     assigned  $\leftarrow \text{false}$ 
7:     if ( $|g| \geq \textit{minGraphSize}$ ) then
8:       for all  $C \in \textit{clusters}$  do ▷ compare graph against all existing clusters
9:          $\textit{minSize} \leftarrow \theta \cdot \max(|g|, |\max(C)|)$ 
10:        if (3) || (4) then ▷ check for cluster exclusion criteria in Eq. 3.3 and 3.4
11:          continue
12:        else
13:           $\textit{minSup} \leftarrow |C| + 1$ 
14:           $\textit{ret} \leftarrow \text{gSpan}''(g \cup C, \textit{minSup}, \textit{thr})$  ▷ checks for common subgraphs
           between  $g$  and graphs in  $C$  that meet the size threshold  $\textit{thr}$ ; returns 1 if at least one
           such subgraph exists, else 0
15:          if  $\textit{ret} = 1$  then
16:             $C.\textit{add}(g)$  ▷ add  $g$  to  $C$  if there exists at least one such subgraph
17:            assigned  $\leftarrow \text{true}$ 
18:          end if
19:        end if
20:      end for
21:      if assigned = false then ▷ create new cluster if  $g$  was not clustered
22:         $C \leftarrow \text{newCluster}(g)$ 
23:        clusters.add( $C$ )
24:      end if
25:    end if
26:  end for
27:  return clusters
28: end procedure

```

---

*gSpan* [245] that mines frequent subgraphs in a database of graphs satisfying a given minimum frequency constraint is used. In this thesis, a minimum support threshold of  $\textit{minSup} = 100\%$  is required in a set of graphs, i.e., all common subgraphs have to be embedded in all cluster members. For the experiments with molecular graph data, *gSpan'*, an optimization of the *gSpan* algorithm for mining molecular databases is used [115]. Since it is not necessary to determine all common subgraphs of a set of graphs, but it is rather important to know if there exists at least one common subgraph that meets the minimum size threshold defined in Equation 3.2, it is possible to terminate search once a solution is found. Due to the structural asymmetry of the search tree (all descendants of a subgraph are generated before its right siblings are extended), it is thus possible to

modify gSpan’ such that the procedure exits immediately when a common subgraph is found that satisfies the minimum size threshold defined in Equation 3.2. In this way, a substantial improvement in runtime performance can be achieved. In the pseudocode, this modification of gSpan’ is called gSpan”. To ensure that cyclic graph structures are not subdivided any further, a special label for edges in cyclic graphs is introduced. Moreover, the following two cluster exclusion criteria are employed to avoid unnecessary calls to the gSpan” algorithm:

$$|x_{m+1}| > |x_{max}| \wedge minSize > |x_{min}|, \quad (3.3)$$

$$|x_{m+1}| < |x_{min}| \wedge minSize > |x_{m+1}|, \quad (3.4)$$

where  $x_{min}$  is the smallest graph in a cluster and  $x_{m+1}$  and  $x_{max}$  are defined as above. Due to these exclusion criteria, graph instances which cannot fulfill the minimum subgraph size threshold are eliminated from further consideration. The first criterion (Equation 3.3) excludes too large query instances that would break up an existing cluster while the second one (Equation 3.4) excludes too small query instances. In case at least one of the two exclusion criteria is met, the computation of the common subgraphs is omitted and the algorithm continues with the next cluster comparison.

In summary, three factors contribute to the practically favorable performance of the approach: First, the use of a gSpan variant to compute a sufficiently large common subgraph, which is known to be effective on graphs of low density. Second, the possibility to terminate search as soon as such a subgraph is found. Third, the cluster exclusion criteria to avoid unnecessary runs of gSpan”.

### 3.1.3 Experiments

To evaluate the the effectiveness and efficiency of the new structure-based clustering approach introduced in Section 3.1.2, several experiments were conducted on eight publicly available data sets of molecular graphs (Table 3.1). In this section, the experimental set-up and the results are described.

**Table 3.1:** Overview of the data sets used for assessing the structural clustering method.

Data set	$n$	min size	mean size	max size	Reference(s)
CPD MOUSE	444	2	13	64	[93]
CPD RAT	580	2	14	90	[93]
CYP	700	1	24	86	[248]
SACA	107	5	27	79	[35]
EPAFHM	580	2	10	55	[193]
FDAMDD	1216	3	23	90	[162]
RepDose	590	2	10	88	[25]
NCI anti-HIV	36255	3	25	139	[238]



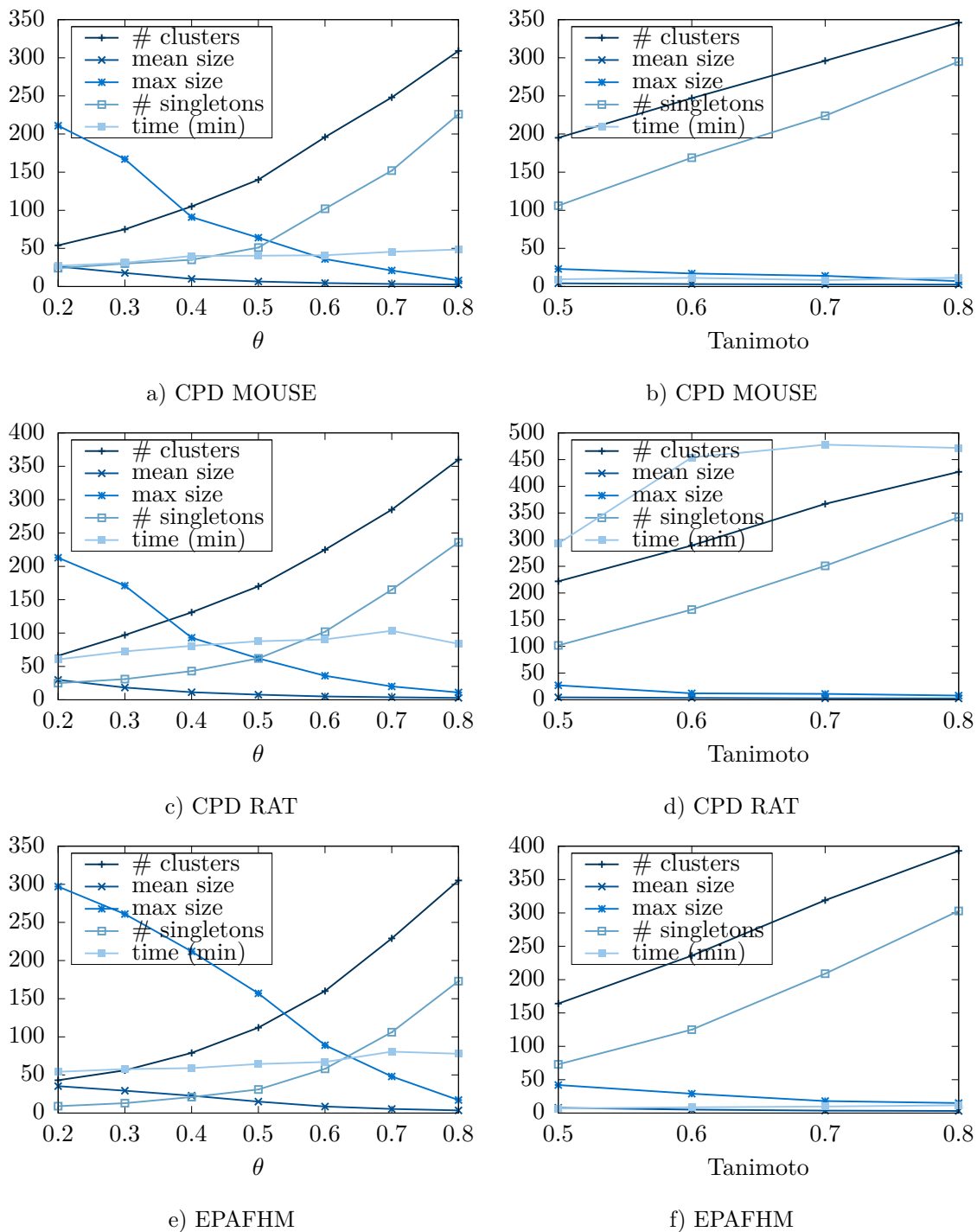
### 3.1.3.1 Baseline Comparison with Fingerprint Clustering

The structure-based clustering algorithm was compared with a clustering algorithm based on fingerprint similarity. The goal of this experiment is to determine if the structural clustering method is able to increase cluster homogeneity as compared to fingerprint clustering. Fingerprint-based similarities can be calculated extremely fast and have been found to perform reasonably well in practice. For the fingerprint calculation of the molecular graph data, the chemical fingerprints in Chemaxon’s JChem Java package are used [118]. The Tanimoto similarity coefficient is used as similarity measure between fingerprints, since these fingerprints are equivalent to Daylight fingerprints<sup>3</sup> which were shown to work well in combination with the Tanimoto coefficient [160, 213].

The fingerprint-based clustering (FP clustering) works as follows. Iteratively, each molecular graph is compared against all yet existing clusters. In case the query graph meets a predefined minimum graph size threshold, *minGraphSize*, and the Tanimoto similarity between the query graph and each graph in the cluster, respectively, exceeds a predefined threshold, the query graph is added to the respective cluster; otherwise a new singleton cluster is created containing the query graph. For each cluster of the FP clustering, the MCS is determined in order to assess intra-cluster homogeneity. Note that the computation of the MCS can be omitted by the structural clustering procedure, since the user-defined similarity threshold provides a measure for the relative size of the common subgraph with respect to the size of the cluster members. In the FP clustering approach, the MCS is obtained as follows. In case a graph  $x_i$  is added to a cluster  $C_j$ , the MCS between  $x_i$  and the current MCS of cluster  $C_j$  is calculated. This MCS is iteratively reduced in size as it is compared to the new cluster members that may not share the entire subgraph. For the MCS calculation, the maximum common edge subgraph algorithm was used which is implemented in Chemaxon’s JChem java package [118].

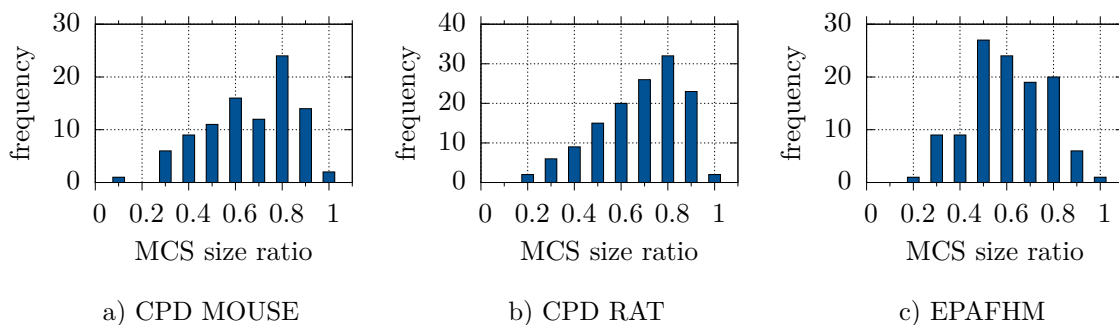
The structural clustering approach was compared with the baseline FP clustering approach on the data sets in Table 3.1. In the following, the results on three representative data sets, i.e., on CPD MOUSE, CPD RAT and EPAFHM, are presented. In all experiments, structural clustering was performed for  $\theta \in [0.2, 0.8]$  using a step size of 0.1. For FP clustering, the Tanimoto similarity threshold was varied from 0.4 to 0.8 with a step size of 0.1. Due to the different input parameters of both clustering approaches, it is not obvious how to compare the clustering results. However, the clustering statistics in Figure 3.5 suggest a correlation between the results from structural clustering for a similarity threshold of  $\theta \in [0, 1]$  and the results from FP clustering for a Tanimoto similarity threshold of  $t = \theta - 0.1$  ( $t \in [0, 1]$ ), due to similar clustering results in terms of the number of clusters, the number of singletons and the mean and maximum size of the clusters. Thus, the clustering results from both algorithms are compared with respect to this heuristic. Figure 3.6 shows the histogram of the relative size of the MCS with respect to the size of

<sup>3</sup> <http://www.daylight.com/dayhtml/doc/theory/theory.finger.html>



**Figure 3.5:** Results of structural clustering (a), (c), (e) vs. fingerprint clustering (b), (d), (f) on CPD MOUSE, CPD RAT and EPAFHM.

the largest cluster instance for all non-singleton FP clusters for a Tanimoto threshold of 0.6. The results indicate that the relative sizes of the MCS with respect to the sizes of the largest cluster instances are, in many cases, below the corresponding structural similarity coefficient  $\theta$ , which serves as a lower bound on the relative size of the MCS. In contrast, each cluster obtained by structural clustering contains at least one common subgraph



**Figure 3.6:** Histogram of the share of the MCS of the largest cluster instance for fingerprint clustering on (a) CPD MOUSE, (b) CPD RAT and (c) EPAFHM using a Tanimoto coefficient value of 0.6.

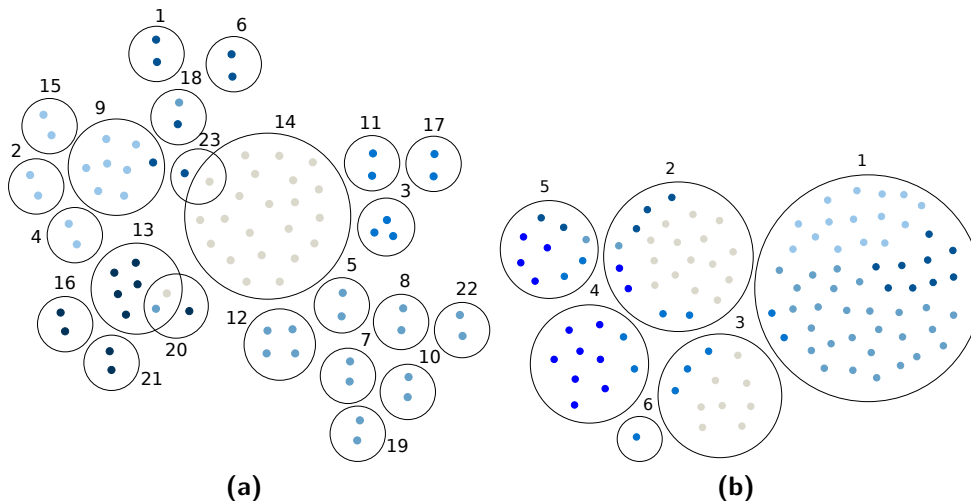
whose share of each cluster member is equal or larger than  $\theta$ . The results suggest that, in comparison to FP clustering, the proposed structural clustering approach provides a superior clustering with reduced intra-cluster heterogeneity in the overall clustering.

### 3.1.3.2 Qualitative Analysis of Structure-Based Clustering

Cluster analysis was performed on the standard anti-cancer agents (SACA) data set consisting of 107 chemical compounds whose class labels corresponding to their mechanisms of action have been clearly classified [237, 135]. The purpose of the experiment is to test if the clusters obtained by structural clustering are in good agreement with the known SACA class labels. As an external measure for clustering validation the Rand index was used to quantify the agreement of the clustering results with the SACA classes. Larger values of the Rand index indicate a better agreement between the clustering results and the SACA classes, with 1.0 indicating perfect concordance. Table 3.2 shows the Rand index values for different similarity coefficient values. Structural clustering clearly shows the peak point of the Rand index at  $\theta = 0.6$ . In the following, the clustering results for  $\theta = 0.6$  are presented partitioning the 107 agents into 52 clusters. 23 of these clusters have at least two members, while the final 29 clusters consist of a single graph. Figure 3.7a gives a representation of the structural clusters with at least two instances in a hypothetical (non-Euclidean) two-dimensional (descriptor) space, where large circles represent clusters and dots, rectangles and stars denote cluster members according to the SACA classes. The results indicate that the clusters tend to be associated with certain SACA classes. Across different values of  $\theta$  it was observed that with a higher similarity coefficient a finer but cleaner grouping of the structures at the cost of generating a larger number of smaller clusters is achieved. The graphs in each class are more cleanly discriminated from other graphs in the data set. Moreover, the clustering produces less overlapping clusters with internally higher structural similarity. In summary, structural clustering is capable of effectively grouping the 107 agents. Graphs instances from the same cluster not only share common subgraphs but are also strongly associated with specific SACA classes of mechanisms of action.

**Table 3.2:** Number of clusters and Rand index values for structural clustering on SACA.

$\theta$	0.02	0.05	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
# Clusters	6	7	11	22	32	39	48	52	60	66	82
Rand Index	0.408	0.436	0.515	0.765	0.827	0.854	0.863	0.869	0.866	0.848	0.833

**Figure 3.7:** Results of (a) structural clustering for  $\theta = 0.6$  and (b) DP Clustering for  $\alpha = 0.1$  and  $m = 1000$  on the SACA data set. The different symbols for the cluster instances represent the six SACA classes.

### 3.1.3.3 Comparison with DP Clustering

The structural clustering method was compared with a graph-based clustering approach based on variational Dirichlet process (DP) mixture models and frequent subgraph mining by Tsuda and Kurihara [225]. The DP clustering approach addresses the problem of learning a DP mixture model in the high dimensional feature space of graph data. The goal of this experiment is to investigate if the approach is also able to rediscover the known structure classes in the SACA database.

In the experiment, the number of features  $m$  was varied from 50 to 5000 and the parameter  $\alpha$  was chosen from the set  $\{0.01, 0.1, 1, 10\}$ . Table 3.3 shows the experimental results for  $\alpha = 0.1$ . The results for  $\alpha = \{0.01, 1, 10\}$  are similar. For  $m \leq 500$  the number of clusters are observed to increase along with the number of features; for  $m > 500$  the number of clusters decreases significantly. Compared to structural clustering, DP clustering produces less clusters. In order to make the results of the DP clustering comparable with the results of the proposed structural clustering approach, the user-specified parameters were varied. Nonetheless, it was impossible to parameterize the DP clustering method to obtain more than seven clusters. Figure 3.7b presents the clustering results for  $m = 1000$ , since Tsuda reported a good behavior of the algorithm for this value. Moreover, additional features can reveal detailed structure of the data. However, this advantage presents a disadvantage at the same time, since graph clusters with thousands of features are difficult to interpret. The DP clustering results indicate that the method is not able to discriminate the known

structure classes in the SACA data set very well. In contrast to the results of structural clustering presented in Section 3.1.3.2, the DP clusters are, in many cases, associated with different structure classes, indicated by lower values of the Rand index (Table 3.3).

**Table 3.3:** Number of clusters and size of the DFS code tree for DP clustering on the SACA data set with  $\alpha = 0.1$ .

# Features	50	100	500	1000	5000
# Clusters	6	7	7	6	2
Rand Index	0.639	0.572	0.761	0.747	0.364

#### 3.1.3.4 Cluster Stability Analysis

The proposed structure-based clustering approach is order-dependent. That is, different clusters are obtained for different orders in which the data is processed. To study the impact of the order of instances in a data set, the stability of the clusters generated by structural clustering was assessed under different permutations of the data. To this end, multiple clustering runs were applied on different permutations of the SACA data set [237, 135] and the agreement of the obtained clusterings was measured.

To assess the stability of a cluster of the initial clustering with respect to a new clustering, a similarity measure between clusters is needed. In this work, the Jaccard coefficient [114] is used as a cluster-wise measure of cluster stability, which is defined as the size of the intersection divided by the size of the union of the sample sets used. Formally,

$$\gamma(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}, \quad (3.5)$$

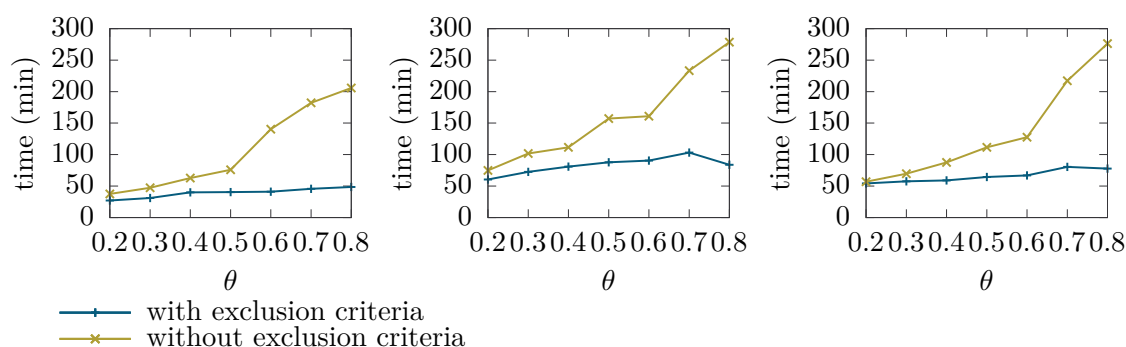
where  $C_1$  and  $C_2$  are two clusters and  $|C|$  denotes the number of instances in cluster  $C$ . This index ranges from 0 (no common instances in cluster  $C_1$  and  $C_2$ ) to 1 (the instances in cluster  $C_1$  and  $C_2$  are exactly the same).

In more detail, given a clustering on the data set generated by the structural clustering method, cluster stability is assessed as follows. Different permutations of the data set are generated and the Jaccard similarities between the original clusters and the most similar clusters in the permuted data are computed. The mean over these similarities is used as an index of the stability of a cluster.

The experimental results suggest that the proposed structural clustering approach generates clusters that are stable with respect to input data permutations. That is, approximately 85% of the clusters of size  $\geq 2$  of a given clustering yield a Jaccard similarity value of 1 with respect to the most similar cluster in a reference clustering. Taking also singletons into account, the similarity of the clusters rises to 94%.

## 3.1.3.5 Performance with/without Cluster Exclusion Criteria

In another experiment, the impact of the cluster exclusion criteria defined in Equation 3.3 and Equation 3.4 on the performance of the structure-based clustering algorithm was investigated. To this end, structural clustering was performed on the data sets in Table 3.1 both with and without the exclusion criteria. Figure 3.8 shows the results of the experiment on three representative data sets, i.e., on CPD MOUSE, CPD RAT and EPAFHM. The results indicate that a significant performance improvement can be achieved with the application of the cluster exclusion criteria.



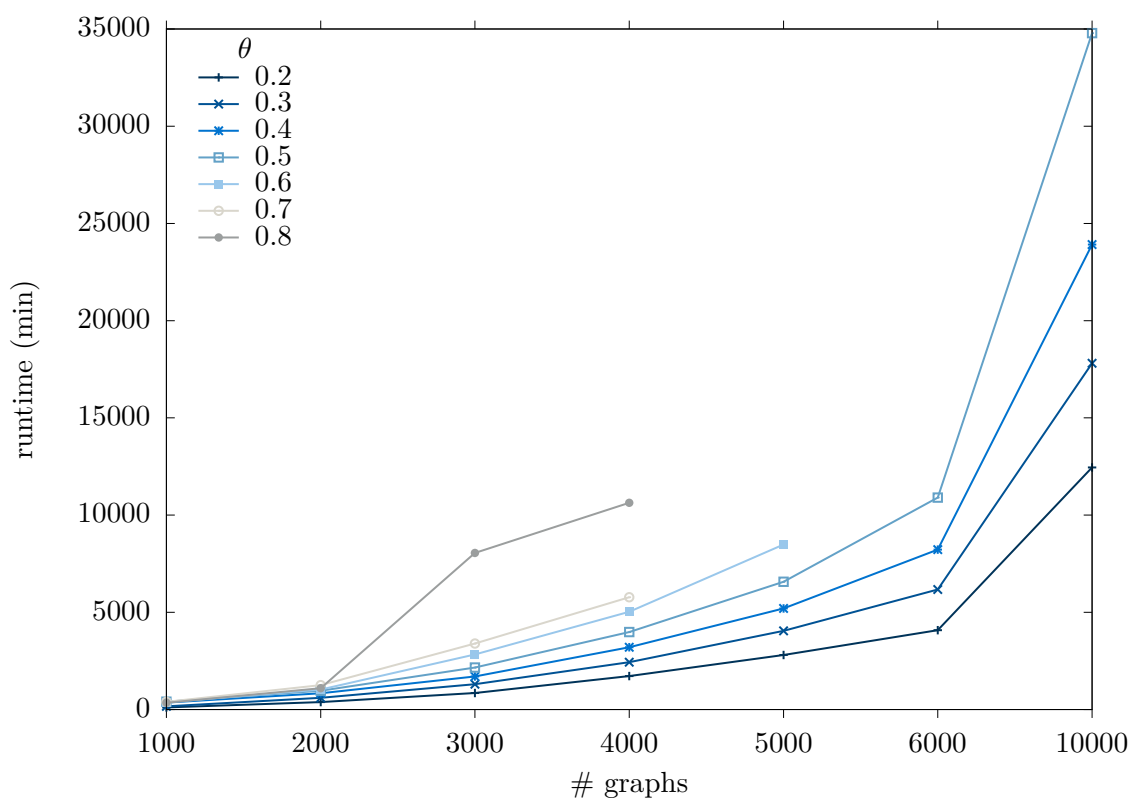
**Figure 3.8:** Runtime performance of the structure-based clustering approach with and without clustering exclusion criteria on CPD MOUSE (left), CPD RAT (middle) and EPAFHM (right).

## 3.1.3.6 Scalability Experiments

To study the scalability of the proposed clustering approach, experiments were performed on ten data sets from the NCI anti-HIV database that consist of  $x$  graphs ( $x \in [1000, 10000]$ ) using a similarity coefficient  $\theta \in [0.2, 0.8]$ . As shown in Figure 3.9, the structure-based clustering algorithm scales favorably as the size of the data set increases. However, for  $0.6 \leq \theta \leq 0.8$ , the algorithm did not respond within a certain timeout period for data sets larger than 4000 and 5000 graphs, respectively. The overall results suggest that, depending on reasonable parameter settings, the proposed clustering approach can handle data sets of at least 10,000 graphs.

## 3.1.4 Conclusion

In this section, a new online algorithm for clustering graphs in a data set in terms of structural similarity was proposed. Structural graph clustering can offer interesting new insights into the composition of graph data sets. Moreover, it can be practically useful to benchmark other graph mining algorithms, to derive new substructural descriptors, to compute local models for classifying graphs, and to calculate the applicability domain of models. Several experiments were designed to evaluate the effectiveness and efficiency of the proposed approach on various real world data sets of molecular graphs. First of



**Figure 3.9:** Runtime performance of the structure-based clustering approach on ten data sets from the NCI anti-HIV database consisting of  $x$  graphs ( $x \in [1000, 10000]$ ).

all, a qualitative analysis was conducted to show that the approach is able to rediscover known structure classes in data sets. Moreover, a baseline comparison with a fingerprint-based clustering was presented. The results demonstrate that the structural clustering approach yields larger and more representative cluster scaffolds compared to FP-based clustering, thus reducing the heterogeneity in the clusters obtained by fingerprint clustering. Further, to show the importance of the defined cluster exclusion criteria, the performance of the structural clustering approach was evaluated with and without these criteria. Finally, to show how well the algorithm scales with respect to the data set size, extensive experiments were performed on a data set comprising 10,000 compounds from the NCI aids anti-viral screen data. In summary, the results suggest that the presented overlapping, non-exhaustive structural clustering approach generates interpretable clusterings in acceptable time. Further work, from an application point of view, includes the following: First, it would be interesting to investigate the effects of preprocessing steps, e.g., downweighting longer chains (acyclic substructures) or reduced graph representations (transforming cycles, in chemical terms: rings, into special nodes). Second, the algorithm could be extended easily to take into account the physico-chemical properties of whole molecules. Technically, this would mean that only graphs within a certain distance with respect to such global graph properties are added to a cluster.

## 3.2 Parallel Structural Graph Clustering

This section addresses the problem of clustering large graph databases according to scaffolds, i.e., large structural overlaps that are shared among all cluster members. More precisely, the cluster members are required to share at least one common subgraph that covers a specific fraction of all graphs in the cluster. An important challenge in this endeavor is the scalability to large graph data sets (of the order of  $10^5$  to  $10^6$  graphs). Graph databases such as the ones representing chemical compounds routinely encompass several hundred thousand graphs; thus, clustering methods that are able to explore and structure the vast graph space are highly desirable. Clustering large databases has emerged as a challenging research area with a large variety of applications, such as in the field of virtual screening, where the task is to analyze large databases of chemical compounds to identify possible drug candidates [233]. By applying clustering techniques it is, for example, possible to prestructure the chemical space, e.g., for local modeling to capture the multi-mechanistic nature of many endpoints, the rediscovery of analog series or visualization. The majority of structural (i.e., scaffold-based) graph-based clustering algorithms, involving e.g., the computation of the MCS, is hardly suitable for such data sets. Graph data sets used in related papers typically contain only several hundred graphs [3, 107, 182], and hardly any effort has been spent on characterizing the performance of the clustering algorithms.

In the previous section, a scaffold-based structural graph clustering algorithm was presented that has been shown to handle graph data sets of at least 10,000 graphs. As this algorithm is still limited in performance, this section presents a parallel, scalable version of the algorithm. The proposed approach, called PSCG (Parallel Structural Clustering of Graph), is based on the idea of task partitioning in conjunction with refined cluster membership tests. More precisely, a set abstraction of graphs and a size-based clustering criterion was used to reduce the number of expensive subgraph search computations, which are not affordable exhaustively on large databases. Moreover, to avoid cluster comparisons with all cluster members, which grow computationally more expensive with increasing cluster size, a cluster representative is defined for each cluster once a unique cluster scaffold is found.

The remainder of the section is organized as follows: In Section 3.2.1, the new approach PSCG is described in detail. Section 3.2.2 presents a description of the data sets and experiments as well as an interpretation of the results. Finally, a conclusion is given in Section 3.2.3.

### 3.2.1 Method

This section presents enhancements and optimizations of the structural clustering algorithm proposed in the previous section that enable PSCG to handle large data sets. The main idea of PSCG is to partition the clustering task into independent tasks which are



distributed among a set of processes, i.e., each process is responsible for one cluster. The motivation behind partitioning the set of clusters instead of the graph data set is that each process can compare all relevant graph instances, i.e., all graph instances with an index greater than the index of the graph that initiated the singleton cluster, against the assigned cluster without the need to wait for the intermediate results of the other processes. To achieve this, a master process is needed which is responsible for managing the cluster results of all processes. The implementation of PSCG adopts the master-worker paradigm for parallelization. The master-worker programming model consists of two kinds of entities: a single master and multiple workers. The master is responsible for decomposing a clustering problem into a subset of clustering tasks and distributing these tasks among a farm of workers (by putting the tasks in a shared queue), as well as for gathering the partial results in order to produce the final computation result. A queue, shared between the master and the workers, is used to represent the shared space where the pending clusters reside. Each worker is responsible for only one cluster at any point in time, independently computing one iteration: It pulls a clustering task (input) from the queue, processes the task by comparing all relevant graphs in the graph database against the cluster, and sends the result, i.e., the processed cluster, back to the master (output).

One of the advantages of using this pattern is that the algorithm is based on a dynamic load balancing of the cluster queue, i.e., the algorithm automatically balances the load. This is possible due to the adoption of a receiver-initiated dynamic load balancing approach based on polling: the work set is shared, and the workers continue to pull work from the set until there is no more work to be done. A static load balancing policy is not adequate for the algorithm as the work load is not known in advance and cannot be estimated easily.

The following sections describe the parallel structural clustering algorithm PSCG in more detail.

### 3.2.1.1 Cluster Comparisons

Let  $minGraphSize$  be the minimum threshold for the graph size and  $minSize$  be the minimum threshold for the size of the common subgraphs specified by the user and defined in Equation 3.2. The algorithm starts with an empty set of clusters. In the first step, the master initiates the computation by creating an initial cluster containing the first graph that is larger than  $minGraphSize$  (Algorithm 2, line 7-11). The master process is responsible for putting the initial cluster in the cluster queue (line 12) which stores cluster instances that are exchanged with the workers. Subsequently, the master increases the number of necessary cluster comparisons for all subsequent graphs (explained in more detail later in this section) (line 14-15). In the following steps, idle workers continue to pull one cluster at a time from the queue (Algorithm 3, line 4) and perform clustering (line 6) by comparing all graph instances in the graph database that lie within a specified index range (which will be explained in more detail in Section 3.2.1.2) against the assigned cluster (Algorithm 4, line 6). In case a query instance meets the  $minGraphSize$  threshold

**Algorithm 2** PSCG: Master

---

**Input:** *graphs* - array of  $n$  graphs  
            $p$ : number of processors

**Output:** *results* - final clusters

```

1: results  $\leftarrow \emptyset$  ▷ resulting cluster queue shared by all threads
2: stable_sort(graphs) ▷ see Section 3.2.1.2
3: for  $i \in 1, \dots, p$  do
4:   |  $w \leftarrow \text{newWorker}()$ 
5:   |  $w.\text{start}()$ 
6: end for
7:  $g \leftarrow \text{dequeue}(\text{graphs})$ 
8: while  $|g| < \text{minGraphSize}$  do
9:   |  $g \leftarrow \text{dequeue}(\text{graphs})$ 
10: end while
11:  $C \leftarrow \text{newCluster}(g)$ 
12: queue.add( $C$ )
13: activeWorkers = true ▷ set to true, if active workers exists
14: for  $g \in \text{graphs}$  do
15:   |  $g.\text{nrClusterComparisons} \leftarrow 1$ 
16: end for
17: while (activeWorkers = true | queue.isEmpty = false) do
18:   | do nothing
19: end while
20: for  $i \in 1, \dots, p$  do
21:   |  $w.\text{terminate}()$ 
22: end for
23: return results

```

---

**Algorithm 3** PSCG: Worker

---

```

1: procedure START
2:   | terminationSignalByMaster = false
3:   | while terminationSignalByMaster = false do
4:     |  $C \leftarrow \text{dequeue}(\text{queue})$ 
5:     | if  $C \neq \text{null}$  then
6:       |  $\text{PSCG}(C, \text{startIdx}, \theta, \text{minGraphSize})$ 
7:     | end if
8:   | end while
9:   | terminate()
10: end procedure

11: procedure TERMINATE
12:   | terminationSignalByMaster = true
13: end procedure

```

---

and shares at least one common subgraph with the cluster that meets the cluster criterion in Equation 3.2 (line 21), the instance is added to the respective cluster (line 22). In case a graph does not belong to any cluster, a new cluster is created. In contrast to the sequential clustering setting, however, in the parallel setting the information whether a

**Algorithm 4** PSCG: Structural Clustering

---

**Input:**  $C$  - cluster  
 $startIdx$  - index of graph that initiated the singleton cluster  
 $\theta$  - similarity threshold ( $\theta \in [0,1]$ )  
 $minGraphSize$  - minimum graph size

```

1: procedure PSCG( $C, startIdx, \theta, minGraphSize$ )
2:    $ret$  - gSpan" return value; returns 0 if no common subgraph exists that meets the
3:     size threshold  $thr$ , 1 if there exists only one such subgraph, 2 if there exists
4:     more than one such subgraph
5:    $endIdx \leftarrow idx(graph \in graphs : |graph| \leq \theta \cdot min(C))$   $\triangleright$  see Section 3.2.1.2
6:   for  $j \in startIdx, \dots, endIdx$  do
7:     if  $graphs[j] \geq minGraphSize$  then
8:        $assigned \leftarrow false$ 
9:       if  $s(\mathbf{f}_{graphs[j]}, \mathbf{f}_C) < \theta \max(|graphs[j]|, |min(C)|)$  then  $\triangleright$  see Section 3.2.1.3
10:         $Mismatch(j, j + 1)$ 
11:         $continue$ 
12:       else
13:          $minSize \leftarrow \theta \cdot \max(|graphs[j]|, |max(C)|)$ 
14:         if  $uniqueScaffold = false$  then  $\triangleright$  see Section 3.2.1.4
15:            $minSup \leftarrow |C| + 1$ 
16:            $ret \leftarrow gSpan'''(graphs[j] \cup C.graphs, minSup, thr)$ 
17:         else
18:            $minSup \leftarrow 2$ 
19:            $ret \leftarrow gSpan''(graphs[j] \cup C.scaffold, minSup, thr)$ 
20:         end if
21:         if  $ret \geq 1$  then
22:            $C.add(graphs[j])$ 
23:            $assigned \leftarrow true$ 
24:           if  $ret = 1$  then
25:              $uniqueScaffold \leftarrow true$ 
26:           end if
27:         end if
28:       end if
29:       if  $assigned = false$  then
30:          $Mismatch(j, j + 1)$ 
31:       end if
32:     end if
33:   end for
34:   if ( $endIdx + 1 < |graphs|$ ) then
35:      $Mismatch(endIdx + 1, |graphs|)$ 
36:   end if
37:    $results.add(C)$   $\triangleright results: resulting cluster queue shared by all threads$ 
38: end procedure

```

---

graph belongs to a cluster is distributed over the set of workers. Since a new cluster can only be created if it is not assigned to any existing cluster, the master needs to maintain the cluster membership information for all graph instances. In particular, for each graph two cluster membership parameters need to be maintained: the number of

**Algorithm 5** PSCG: Maintenance of cluster membership information

---

**Input:** *startIdx* - start index  
*endIdx* - end index

```

1: procedure MISMATCH(startIdx,endIdx)
2:   for idx  $\in$  startIdx, ..., endIdx - 1 do
3:     graphs[idx].nrMismatches++
4:     if graphs[idx].nrCluComp = graphs[idx].nrMismatches then
5:       C  $\leftarrow$  new Cluster(graphs[idx])
6:       queue.add(C)
7:       for i  $\in$  idx + 1, ..., |graphs| - 1 do
8:         | graphs[i].nrClusterComparisons++
9:       end for
10:    end if
11:  end for
12: end procedure

```

---

necessary cluster comparisons as well as the numbers of clusters the graph does not fit into (denoted as the number of cluster mismatches). If a graph does not belong to a cluster the worker forwards the non-membership information to the master (Algorithm 4, line 30). Note, that due to the overlapping nature of the clustering algorithm, a graph can be directly assigned to a cluster in case it meets the cluster criterion without informing the master. Each time a worker reports a cluster mismatch for a graph, the master first increases the mismatch parameter for the graph (Algorithm 5, line 2-3) and then checks the two cluster membership parameters. If the number of necessary cluster comparisons is equal to the number of cluster mismatches (line 4), suggesting that the corresponding graph does not belong to any cluster, a new cluster is created (line 5). The master puts the cluster in the task queue (line 6) and increases the cluster comparison parameter for all subsequent graphs in the graph data set (line 7-8). Once a worker is done with an iteration, the resulting cluster is added to the result queue managed by the master (Algorithm 4, line 37). The flowcharts in Figures 3.10 and 3.11 illustrate the master-worker paradigm of PSCG. A graphical illustration of the clustering process on a sample data set of molecular graphs is shown in Figure 3.12, where large circles represent clusters and the single structures outside denote singleton clusters. The table contains the cluster membership parameters maintained by the master.

As in the sequential clustering algorithm, *gSpan''* is used for computing common subgraphs. Given that pairwise subgraph similarity computation is very expensive, it would be highly desirable to reduce the number of subgraph computations. Therefore, the following cluster exclusion criteria are introduced to avoid unnecessary calls to the *gSpan''* algorithm in the first place: a refined cluster membership test based on node feature vectors of graphs, and a clustering exclusion criterion based on the size of graph instances which requires the graph data set to be sorted according to size. These criteria are employed to perform a search space pruning on the actual clustering. The aim of search space pruning is to reduce the number of graph candidates in the database that need to

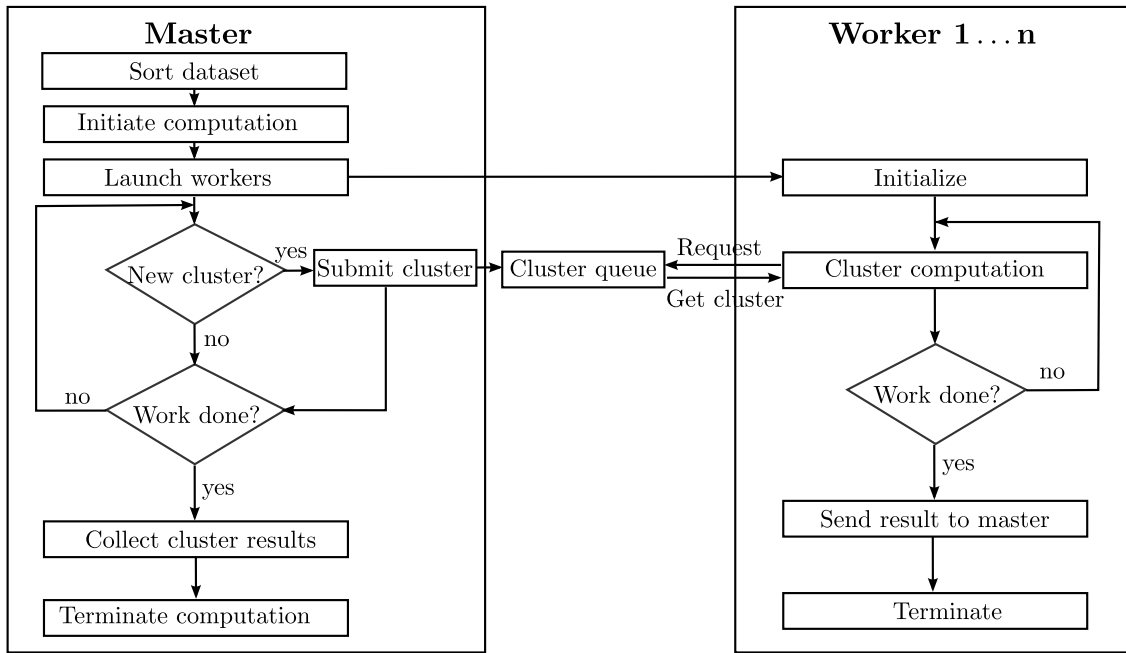


Figure 3.10: Flowchart of the master-worker paradigm employed by PSCG.

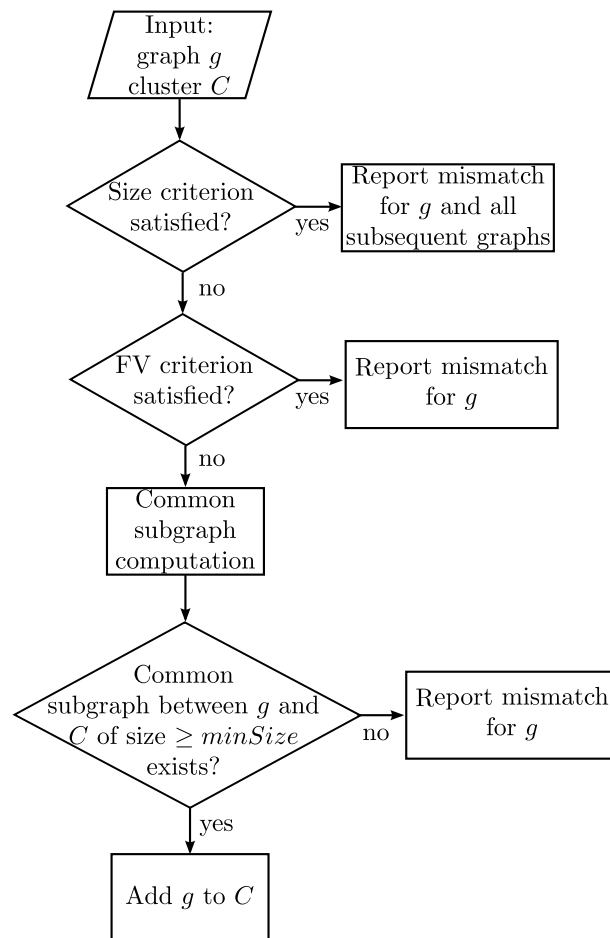


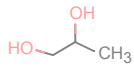
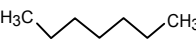
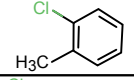
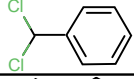
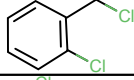
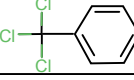
Figure 3.11: Flowchart of the worker computation.

undergo an expensive, full fledged graph matching process. Further, to reduce gSpan running times for larger clusters, a cluster representative is defined for each cluster composed of the common cluster scaffold once this scaffold is unique and thus also minimal. In the following three subsections, the employed cluster exclusion criteria and the intuition behind the definition of a cluster representative are described in more detail. The impact of these algorithmic improvements will be investigated in Section 3.2.2.

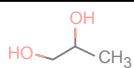

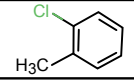
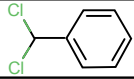
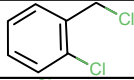
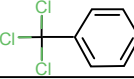
### 3.2.1.2 Size-based Exclusion Criterion

The cluster criterion defined in Equation 3.2 constrains the set of graphs being considered for clustering. More precisely, only graphs in a certain size range are considered for comparison with a specific cluster, i.e., graphs whose sizes lie in the range  $[[\theta x_{max}], \lfloor \frac{1}{\theta} x_{min} \rfloor]$ , where  $x_{min}$  is the smallest and  $x_{max}$  is the largest graph instance in the cluster. The lower bound of the size range ensures that only graph instances that are equal to or larger than the minimum required size for at least one common subgraph,  $minSize$ , are considered for cluster membership. This is necessary since at any point in time at least one common subgraph should make up a proportion  $\theta$  of the size of each cluster member. The upper bound excludes query instances that are larger than  $minSize$  and thus would break up an existing cluster. Incorporating this information in the clustering process would give us the possibility to avoid comparing a cluster to the complete database.

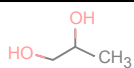

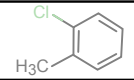
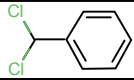
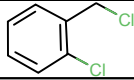
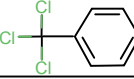
To effectively employ the size-based criterion, the data set is sorted in increasing order of graph size. Thus, the subsequent graphs do not need to be compared against a cluster, once a query graph exceeds the upper bound of the size range. To preserve the incremental character (i.e., each graph in the graph database is only processed once by comparing it against all existing clusters) of the structural clustering algorithm [204], the graph index corresponding to the lower bound needs to be greater than the index following the index of the graph instance that initiated the assigned singleton cluster. However, due to the ordering of the data set by size, the graph index corresponding to the lower bound is always equal to or smaller than the index of the graph that initiated the singleton cluster. Thus, the graph indices that are considered for comparison against a cluster lie in the range  $[idx(x_{min}) + 1, idx(x : |x| \leq \lfloor \frac{1}{\theta} x_{min} \rfloor)]$ , where  $x_{min}$  is the smallest graph in the cluster. Due to the ordering of the data set, this graph corresponds to the graph that initiated the clustering. Figure 3.13 illustrates the use of the size-based exclusion criterion during the clustering process on a data set of eight molecular graphs for  $\theta = 0.5$ . The left figure shows the clustering at time  $t_1$ . Only graphs of size  $\leq \frac{1}{\theta} \cdot x_{min} = \frac{4}{0.5} = 8$  need to be considered for comparison against cluster 1, since for graphs of size  $> 8$  it is impossible to find a common subgraph with  $x_1$  that covers at least 50% of both graphs. Hence, graph  $x_2$  is the only graph that needs to be compared against cluster 1. The right figure shows the clustering at time  $t_2$ . Here, only graphs of size  $\leq \frac{1}{\theta} \cdot x_{min} = \frac{7}{0.5} = 14$  need to be considered for clustering against cluster 2. The only graphs that meet this criterion are graphs  $x_3$  to  $x_7$ .

		# cluster comparisons	# mis-matches
$x_1$		0	0
$x_2$		1	0
$x_3$		1	0
$x_4$		1	0
$x_5$		1	0
$x_6$		1	0

(a)

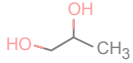

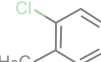
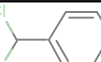
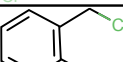
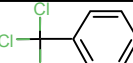
		# cluster comparisons	# mis-matches
$x_1$		0	0
$x_2$		1	1
$x_3$		2	0
$x_4$		2	0
$x_5$		2	0
$x_6$		2	0

(b)

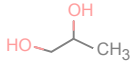

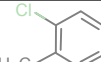
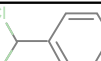
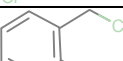
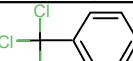
		# cluster comparisons	# mis-matches
$x_1$		0	0
$x_2$		1	1
$x_3$		2	2
$x_4$		3	0
$x_5$		3	0
$x_6$		3	0

(c)

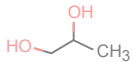

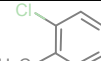
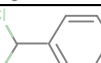
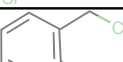
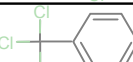
Figure 3.12: Example sequence of steps of PSCG.

		# cluster comparisons	# mis-matches
$x_1$		0	0
$x_2$		1	1
$x_3$		2	2
$x_4$		3	3
$x_5$		4	0
$x_6$		4	0

(d)

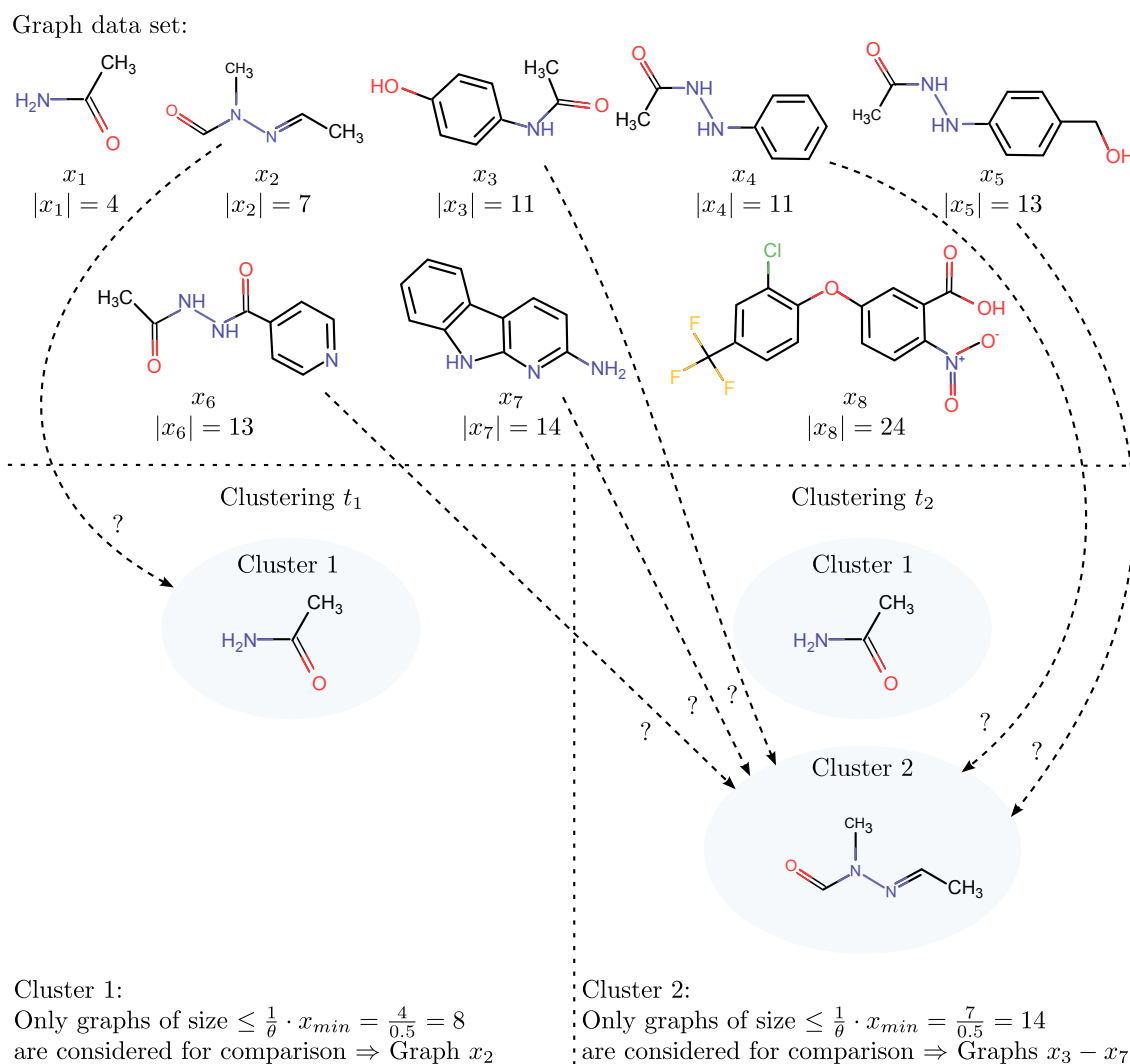
		# cluster comparisons	# mis-matches
$x_1$		0	0
$x_2$		1	1
$x_3$		2	2
$x_4$		3	3
$x_5$		4	2
$x_6$		4	0

(e)

		# cluster comparisons	# mis-matches
$x_1$		0	0
$x_2$		1	1
$x_3$		2	2
$x_4$		3	3
$x_5$		4	2
$x_6$		4	3

(f)





**Figure 3.13:** Example illustrating the use of the size-based cluster exclusion criterion on a data set of chemical compounds containing eight graphs ( $\theta = 0.5$ ). Left: Clustering at time  $t_1$ . Only graphs of size  $\leq \frac{1}{\theta} \cdot x_{min} = \frac{4}{0.5} = 8$ , i.e., graph  $x_2$ , need to be considered for comparison against cluster 1. Right: Clustering at time  $t_2$ . Only graphs of size  $\leq \frac{1}{\theta} \cdot x_{min} = \frac{7}{0.5} = 14$ , i.e., graphs  $x_3$  to  $x_7$ , need to be considered for clustering against cluster 2.

### 3.2.1.3 Exclusion Criterion based on Node Feature Vectors

The second clustering exclusion criterion is based on a set abstraction of graphs, i.e., a numerical feature vector representing the number of node types in a graph. The underlying idea is that for two graphs the overlapping node set represents an upper bound for the size of the maximum common subgraph. Thus, given a query instance, the computation of common subgraphs shared with the members of a cluster can be omitted if the size of the overlapping node set of the query graph and the cluster representative is smaller than *minSize*.

Formally, during the preprocessing phase of structural clustering, each graph  $x_i$  is rep-

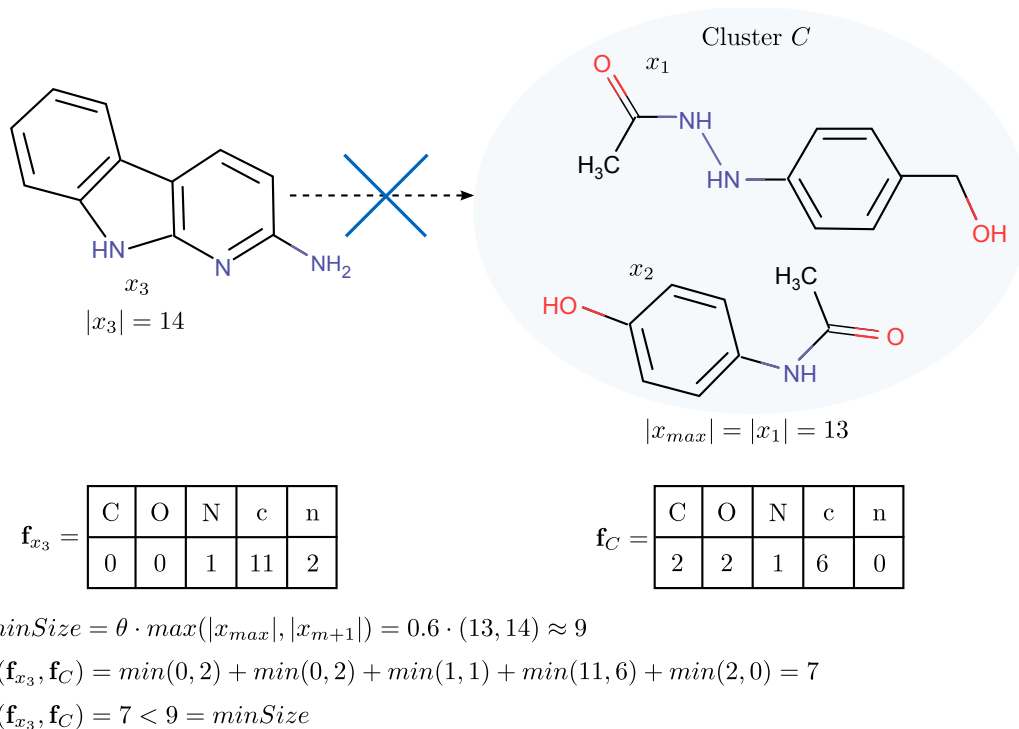
represented by a numerical feature vector  $\mathbf{f}_{x_i} = (f_{x_i}^1, \dots, f_{x_i}^n)$  corresponding to a set of vertex types  $l_1, \dots, l_n$ . Each entry in the feature vector records the number of a specific vertex type occurring in the respective graph. Let  $f_{x_i}^k$  denote the numerical feature associated with the vertex type  $v_k$ . Each cluster  $C_j$  is represented by a vector  $\mathbf{f}_{C_j} = (f_{C_j}^1, \dots, f_{C_j}^n)$  defined in terms of the overlap of the feature vectors of the instances in that cluster, i.e., the common vertex type set shared by all cluster instances. The similarity  $s$  between  $\mathbf{f}_{x_i}$  and  $\mathbf{f}_{C_j}$  is computed by summing up the minimum of each pair of feature vector components

$$s(\mathbf{f}_{x_i}, \mathbf{f}_{C_j}) = \sum_k (\min(\{f_{x_i}^k \in \mathbf{f}_{x_i}\} \cup \{f_{C_j}^k \in \mathbf{f}_{C_j}\})) \quad (3.6)$$

representing an upper bound on the size of the maximum common subgraph (Algorithm 4, line 9). If the similarity  $s(\mathbf{f}_{x_i}, \mathbf{f}_{C_j})$  is lower than the minimum threshold for the size of the common subgraphs,  $minSize$  (Equation 3.2), i.e.,  $s(\mathbf{f}_{x_i}, \mathbf{f}_{C_j}) < minSize$ , the computation of the common subgraphs is omitted and the cluster mismatch is reported to the master (line 10). PSCG then continues with the next cluster comparison (line 11). In this way, graphs with a limited degree of resemblance to the target cluster are eliminated, and the overall speed of the algorithm is increased. Figure 3.14 shows a sample application of the feature vector criterion for  $\theta = 0.6$ . In this example, the query graph  $x_3$  is compared against cluster 1 containing two graphs,  $x_1$  and  $x_2$ . As the similarity between the node feature vector of the query graph and the the node feature vector of the cluster is smaller than the minimum required size of the common subgraph  $minSize$ , the query graph is not considered for the cluster membership test. Hence, the computation of the common subgraphs can be omitted.

#### 3.2.1.4 Definition of a Cluster Representative

As mentioned in Section 3.1.2, the structural clustering method limits subgraph mining to the search of one common subgraph that satisfies the minimum size threshold,  $minSize$  to avoid the computation of all frequent common subgraphs. This limitation forces us to compare each query graph against all cluster members which may have a remarkable impact on the runtime of gSpan, in particular for larger clusters. To reduce running times, a cluster representative is defined for each cluster once all cluster members share a unique cluster scaffold, i.e., the minimum required common subgraph is the only common subgraph all cluster members have in common. Since in the structural clustering algorithm introduced in Section 3.1 subgraph mining is terminated once a common subgraph is found that satisfies  $minSize$ , the existence of further common subgraphs is unknown. Therefore, it is necessary to go one level deeper in the subgraph mining process and check if there exists at least another common subgraph with size equal to or greater than  $minSize$ . In the pseudocode, this modification of gSpan is called gSpan''' (Algorithm 4, line 16). As soon as all graphs in a cluster share no more than one common subgraph, this unique subgraph is used as the cluster representative (see example in Figure 3.15). In the following, all subsequent query graphs are compared against the cluster representative

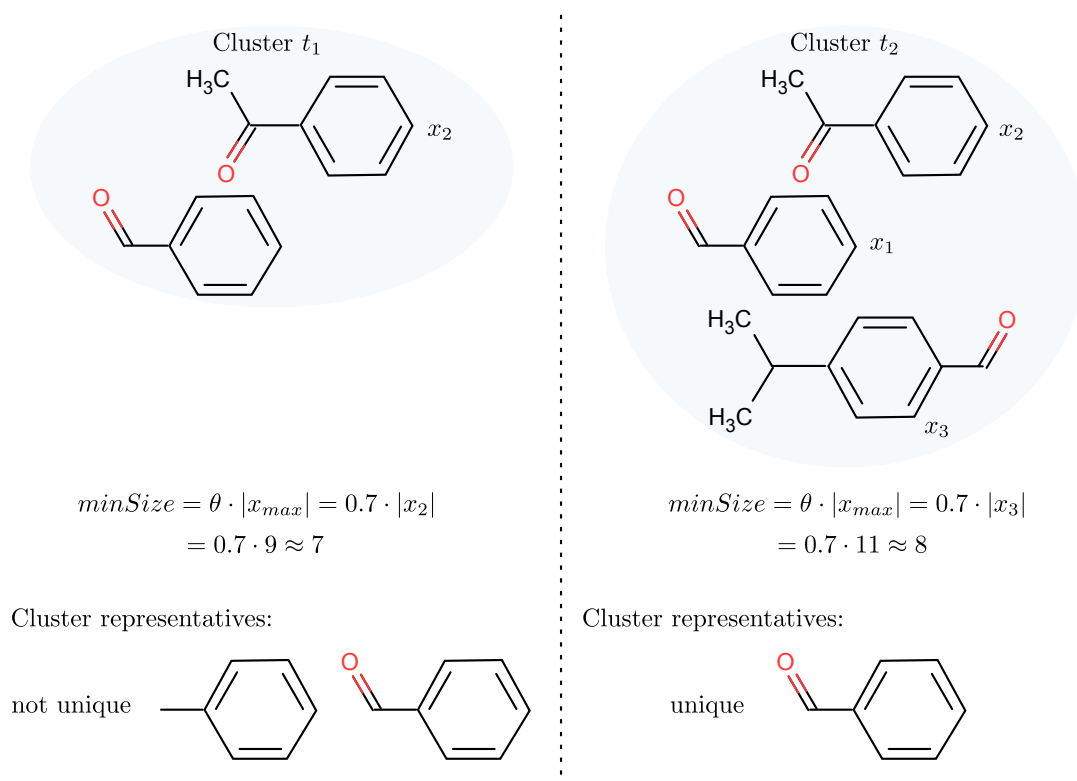


**Figure 3.14:** Example use of the feature vector-based cluster exclusion criterion ( $\theta = 0.6$ ). Since the similarity between the feature vectors of query graph  $x_3$  and cluster 1 is smaller than the minimum required size of the common subgraph, the common subgraph computation step can be omitted.

instead of comparing it against all graphs in the cluster (line 18-19). Further, subgraph mining is terminated as soon as a common subgraph of size  $\text{minSize}$  is found that is covered by the query graph and the cluster representative, i.e.,  $\text{gSpan}''$  is used. Note, that the reason for not defining a cluster representative before the existence of a unique cluster scaffold is due to the following two reasons. First, there may exist at least another common subgraph of size  $\text{minSize}$ . By using the first common subgraph found as cluster representative, it may be the case that the query graph and the cluster representative share a common subgraph of size  $\text{minSize}$  that is not the first common subgraph. In this case, by mistake the query graph would not be assigned to the cluster. Second, there may exist larger subgraphs. By ignoring the existence of these subgraphs and using the first common subgraph found as cluster representative, it may be the case that the  $\text{minSize}$  threshold is smaller than the size of the common subgraph shared by the query graph and the cluster representative. Thus, the query graph would not be assigned to the cluster even if there exist larger common subgraphs that fulfill the size threshold.

### 3.2.2 Experimental Results

To evaluate the efficiency of the parallel structural clustering algorithm PSCG, introduced in Section 3.2.1, several experiments were conducted on several publicly available data sets



**Figure 3.15:** Example illustrating the definition of a cluster representative. At time  $t_1$ , the cluster members  $x_1$  and  $x_2$  share two common subgraphs that meet the *minSize* threshold. Hence, there does not exist a unique cluster scaffold. At time  $t_2$ , there exists a unique cluster scaffold which is used as cluster representative for further cluster assignments.

of molecular graphs. In this section, the data sets, the experimental set-up and the results are described.

### 3.2.2.1 Test Environment and Data Sets

The clusterings on the data sets containing up to 200,000 graphs were carried out on a SUN x4600 system with 32 AMD Opteron CPU cores (8 CPU sockets with 4 CPU cores) using the multi-threaded version of the algorithm. The processor in each node runs at 2.5 GHz with 2 GB of main memory. The clusterings on the data set containing 300,000 structures were carried out using the MPI parallelized version of the algorithm. Here, the compute cluster consists of 2016 AMD Opteron (Magny-Cours) CPU cores (42 Dell R815 nodes with 48 CPU cores and 128-256 GB main memory) and Qlogic infiniband interconnects. The algorithm was implemented in C++ using the boost libraries ([www.boost.org](http://www.boost.org)) for multi-threading support. For the experiments, the chemical domain was employed as application area by using real data sets of molecular graphs. The first data set contains the first 10,000 structures of the NCI anti-HIV database ([http://dtp.nci.nih.gov/docs/aids/aids\\_data.html](http://dtp.nci.nih.gov/docs/aids/aids_data.html)) which contains 36,255 compounds. The second data set, ChemDB, contains nearly 5 M commercially available small molecules [48, 49]. From this data set, data sets

sized from 100,000 to 300,000 graphs were created using random sampling.

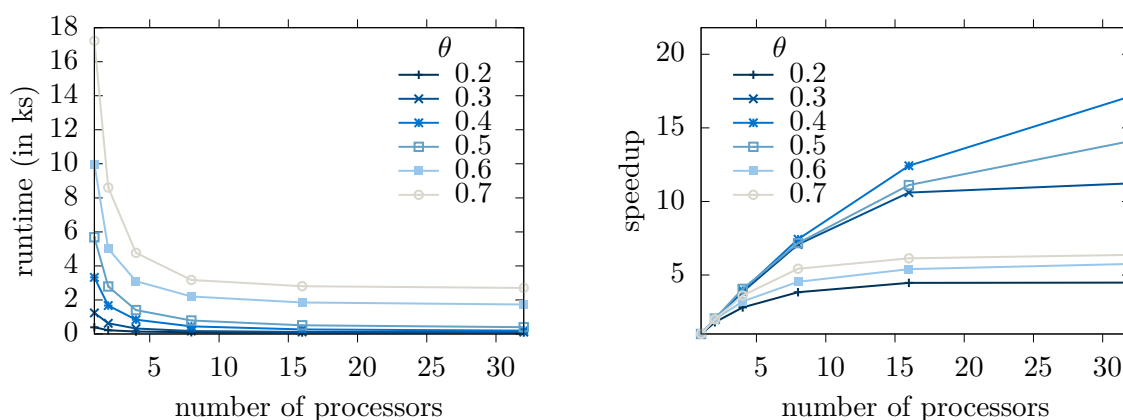
### 3.2.2.2 Performance Evaluation

The runtime performance of PSCG was investigated for different numbers of processors (1, 2, 4, 8, 16 and 32) and different values of  $\theta$  using the first 10,000 graph structures from the NCI anti-HIV database. An index called the speedup factor was used to determine the advantage afforded by the parallel implementation. Speedup ( $S$ ) is defined as a ratio of the time taken in running the sequential algorithm ( $T_s$ ) to the time taken in running the parallel algorithm ( $T_p$ ) with  $P$  processors, i.e.,  $S = \frac{T_s}{T_p}$ .

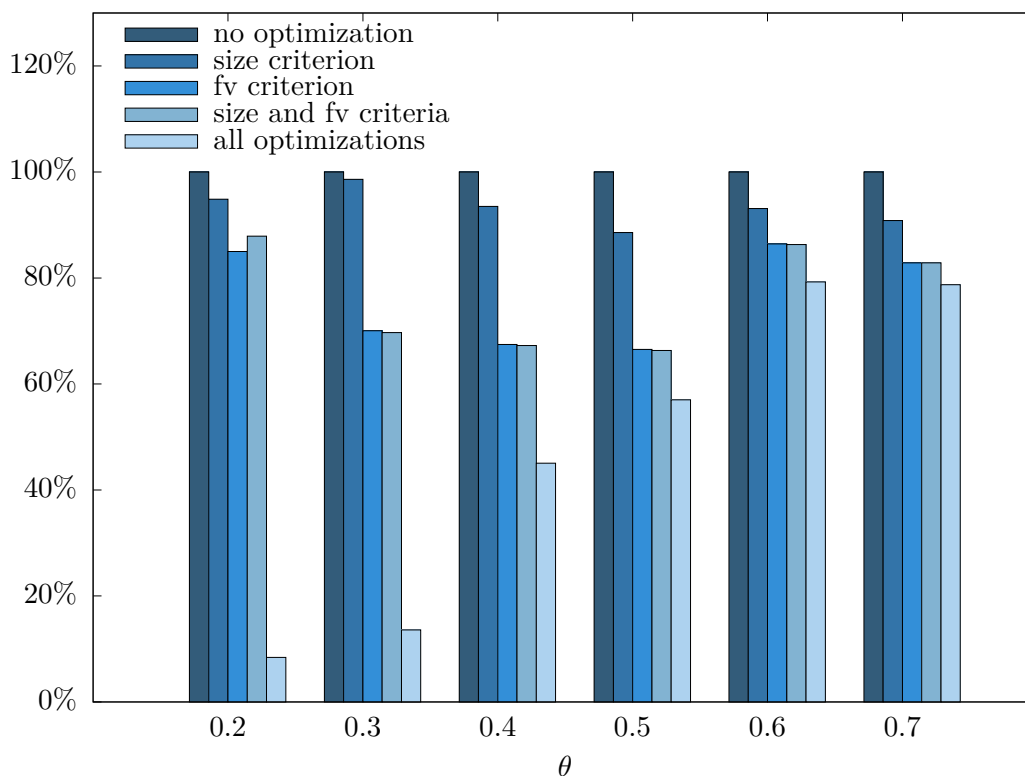
Figure 3.16 shows the execution time and the speedup for different values of  $\theta$ . The results indicate that PSCG scales well with the number of processors and has a good speedup which is close to linear for certain parameter settings, i.e., for smaller values of  $\theta$ . For larger similarity coefficients, there is a higher number of computationally more demanding cluster comparisons, especially at the end of the clustering when the graphs become larger and the runtime degenerates.

### 3.2.2.3 Effects of Algorithm Improvements

Further, the impact of the algorithm improvements presented in Section 3.2.1.2, 3.2.1.3 and 3.2.1.4 on the performance of PSCG was investigated. For this, clustering was performed on the NCI anti-HIV data set with 32 processors using (i) no optimizations, (ii) only the size-based exclusion criterion, (iii) only the feature vector-based exclusion criterion, (iv) both the size- and feature vector-based criteria and (v) all optimizations including the definition of a cluster scaffold once it is unique. Figure 3.17 shows the runtime reduction and Figure 3.18 an overview of the relative frequency of both exclusion criteria as well as the frequency of gSpan calls. The results indicate that significant performance improvements, especially for  $\theta \leq 0.5$ , can be achieved with the application of the cluster exclusion criteria and the definition of a cluster scaffold.



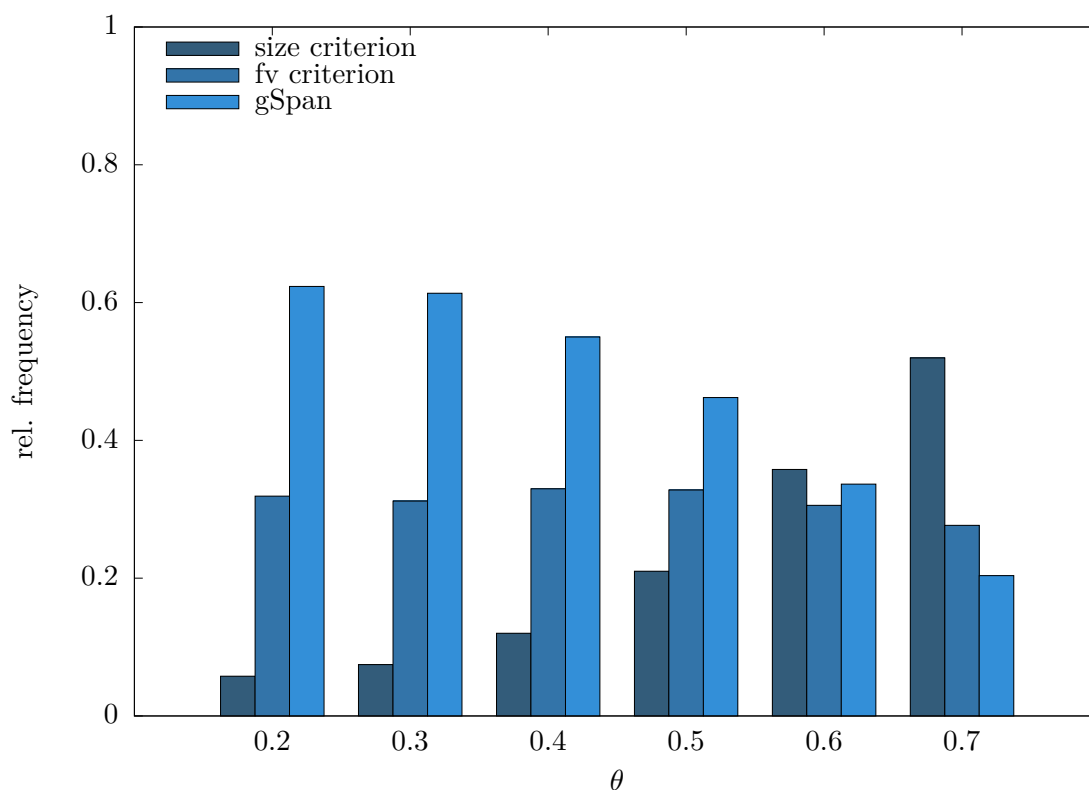
**Figure 3.16:** Execution time (left) and speedup (right) of PSCG on the first 10,000 graphs of the NCI anti-HIV data set.



**Figure 3.17:** Runtime reduction due to algorithm improvements.

#### 3.2.2.4 Comparison to Sequential Structural Clustering

The runtime performance of PSCG was compared with the sequential structural clustering algorithm presented in Section 3.1 on the first 10,000 structures of the NCI anti-HIV data set. For accurate comparison, the same experimental setup was used for both methods. This section only shows the experimental results for  $\theta \in [0.2, 0.5]$ , since for  $\theta \geq 0.5$ , the sequential algorithm did not terminate within a certain timeout period. Table 3.4 shows the runtime performance of both clustering versions. The runtime advantage of PSCG over the sequential clustering version is clear, showing improved computation efficiency by factors of 300 fold to 1900 fold for PSCG. The reasons for this can be explained by the following improvements in PSCG. First, the clustering task is partitioned into independent tasks which are distributed among a set of workers. Each worker compares the graph structures in the data set against the assigned cluster without the need to wait for the intermediate results of the other processes. Second, two clustering exclusion criteria were introduced which reduce the number of cluster membership tests. Third, a cluster representative is defined once the scaffold of a cluster is unique, to avoid cluster comparisons with all cluster members. Fourth, the invocation overhead of the individual gSpan runs was reduced. This optimization is especially efficient for gSpan runs with low overall runtimes.



**Figure 3.18:** Relative frequency of size-based and feature vector-based exclusion criterion and number of gSpan calls.

**Table 3.4:** Runtime (in sec) of the sequential clustering version vs. PSCG on the first 10,000 graphs of the NCI anti-HIV data set for different values of  $\theta$ .

$\theta$	0.2	0.3	0.4	0.5
$t_{seq}$	747,000	1,068,420	1,434,780	2,087,280
$t_{par}$	396	1,244	3,394	6,235

### 3.2.2.5 Experiments on Large Graph Data Sets

PSCG was tested on three data sets sampled from the ChemDB data set containing 100,000, 200,000 and 300,000 graphs respectively. For the experiments, 32 CPUs were

**Table 3.5:** Runtime (in sec) for the sampled data sets.

$ D $	$\theta = 0.4$	$\theta = 0.6$
100,000	31,103 ●	67,563 ●
200,000	122,204 ●	349,568 ●
300,000	610,577 ○	1,163,761 ★

●: 32 processors ○: 96 processors ★: first half: 96 processors, second half: 48 processors

**Table 3.6:** Number of clusters for the sampled data sets.

$ D $	$\theta = 0.4$	$\theta = 0.6$
100,000	4,112	16,295
200,000	6,096	25,685
300,000	9,811	38,775

used for the data sets with 100,000 and 200,000 graphs. For the data set containing 300,000 graphs 96 CPUs were used for  $\theta = 0.4$ . For  $\theta = 0.6$ , 96 (48) CPUs were used to cluster the first (second) half of the data set. The rationale for the change in the CPU number is that the parallel efficiency of the algorithm can change over the runtime of the algorithm (i.e., towards the end a large number of workers may be idle constantly). The MPI version contains a checkpoint/restart facility which allowed us to adjust the number of used CPU cores to account for this by manually balancing the workload on the cluster. Tables 3.5 and 3.6 show the runtime performance as well as the number of created clusters on the sampled data sets for  $\theta = 0.4$  and  $\theta = 0.6$  using all three previously described algorithmic improvements.

### 3.2.3 Conclusion

This section presented PSCG, a parallel and improved version of the structural graph clustering algorithm introduced in Section 3.1. PSCG uses a task partitioning approach and makes use of two clustering exclusion criteria to reduce cluster membership tests. Further, to reduce gSpan running times for larger clusters, a cluster representative is defined for each cluster composed of the common cluster scaffold once this scaffold is unique. To study the effectiveness of the proposed algorithm for clustering large data sets, extensive experiments were conducted. The experimental results suggest that the algorithm scales well with the increasing size of the data and, for certain parameter settings, speeds up nearly linearly with the increasing number of processors. For real world data sets, this algorithm is able to handle a much greater number of graph instances compared to previously proposed structure-based clustering algorithms. Given these performance improvements, the algorithm should already be applicable to the large structure databases from virtual screening.



### 3.3 Structural Clustering by Abstract Pre-clustering

The previous section introduced an approach for clustering large databases of graphs. The parallelized scaffold-based structural graph clustering approach, PSCG, has been shown to handle graph data sets of at least 300,000 graphs was presented. Still, common real-world compound libraries contain millions of graph structures, and clustering algorithms to structure these libraries are needed. In this section, a new scaffold-based algorithm for clustering such very large molecular graph databases is proposed.

The approach, named SCAP (Structural Clustering by Abstract Pre-clustering), employs two stages. It first partitions the original data set into several smaller data sets using a greedy clustering approach inspired by dynamic seed-based clustering (DySC) [257] for RNA reads and a similarity measure based on an abstraction from the actual structural similarity measure. The pre-clustering approach is referred to as APreClus (Abstract Pre-Clustering). APreClus is an online and instance incremental clustering algorithm delaying the final cluster assignment of an instance until one of the so-called pending clusters the instance belongs to has reached significant size and is converted to a fixed cluster. Once a cluster is fixed, APreClus recalculates the cluster centers which are used as representative for further cluster assignments. This has the advantage that instances initially wrongly assigned to a cluster can be either still assigned to other pending clusters, or are recycled as new input instances. In other words, errors of assigning instances too early to a cluster can be corrected, even if an initial assignment to a cluster has been made. The pre-clustering methodology employs a cluster membership test based on a set-abstraction of graphs. The motivation behind the pre-clustering step is that by using this pre-clustering approach and the abstraction-based similarity measure, dissimilar partitions of the original data set are generated, without the loss of information. The similarity measure ensures that only graphs which have the potential of being structurally similar are assigned to the same partition. Overall, this leads to a reduction in the number of expensive subgraph search computations performed in the second clustering stage, which are now not required as the partitions are dissimilar to each other. The resulting partitions are further clustered into a finer level of granularity using a variant of the highly parallelized scaffold-based clustering approach, PSCG, that produces overlapping (non-disjoint) and non-exhaustive clusters. The second-stage clustering avoids cluster comparisons with all cluster members, which grow computationally more expensive with increasing cluster size, by defining a cluster representative for each cluster.

The remainder of the section is organized as follows: Section 3.3.1 introduces the scaffold-based structural graph clustering approach SCAP. In Section 3.3.2, the experimental results are presented and discussed, before Section 3.3.3 concludes.

### 3.3.1 Method

SCAP clusters graphs according to scaffolds that are common between cluster members. The pseudocode of SCAP is shown in Algorithm 6. The approach employs two clustering stages. In the first clustering stage, the data set is pre-partitioned into a set of smaller data sets employing an approach based on a dynamic seeding strategy [257] (see Algorithm 6, line 3). Section 3.3.1.1 describes the pre-clustering methodology in more detail. As mentioned earlier, the motivation behind the pre-clustering step is to generate dissimilar partitions of the original data set, without loss of information. Choosing a similarity measure based on a set-abstraction of graphs, it is guaranteed that only graphs which have the potential of being structurally similar are assigned to the same partition. Hence, the number of computationally expensive subgraph computations performed in the second clustering stage, which are not required any more as the partitions are dissimilar to each other, can be drastically reduced. In the second clustering stage the resulting data partitions are clustered into a finer level of granularity using a scaffold-based graph clustering approach (Algorithm 6, line 5) inspired by PSCG that can handle even larger data sets of graphs. The modification is referred to as PSCG' which will be described in Section 3.3.1.2.

---

**Algorithm 6** SCAP

---

**Input:** *graphs* - queue of  $n$  graphs to be clustered

$\theta$  - similarity threshold ( $\theta \in [0,1]$ )

*max\_pending* - cluster size threshold

**Output:** *clusters* - clusters

```
1: procedure SCAP(graphs, $\theta$ ,max_pending)
2:   clusters  $\leftarrow \emptyset$ 
3:   partitions  $\leftarrow$  APreClus(graphs, $\theta$ ,max_pending)
4:   for  $C \in$  partitions do
5:     | clusters.add(PSCG'(C, $\theta$ ))
6:   end for
7:   return clusters
8: end procedure
```

---

#### 3.3.1.1 APreClus

The pre-clustering approach APreClus is a variation of a recently proposed online and instance incremental clustering algorithm, called Dynamic Seed-based Clustering (DySC) [257]. The approach uses a dynamic seeding strategy to achieve higher accuracy while preserving scalability and efficiency. In the following, the most important concepts of the algorithm are presented. The main purpose is to reduce the amount of inaccurately formed clusters in the early stages of a greedy clustering. Additionally, Figure 3.21 illustrates the clustering workflow of APreClus.

**Algorithm 7** APreClus

---

**Input:**  $graphs$  - queue of  $n$  graphs to be clustered  
 $\theta$  - similarity threshold ( $\theta \in [0,1]$ )  
 $max\_pending$  - cluster size threshold

**Output:**  $\mathcal{C}_{Fix}$  - resulting set of clusters

```

1: procedure APRECLUS( $graphs, \theta, max\_pending$ )
2:    $\mathcal{C}_{Fix}, \mathcal{C}_{Pending} \leftarrow \emptyset$ 
3:    $\mathcal{C}_{Pending}.add(newCluster(dequeue(graphs)))$ 
4:   while  $graphs \neq \emptyset$  do
5:      $g \leftarrow dequeue(graphs)$ 
6:      $assigned \leftarrow false$ 
7:      $m \leftarrow \{(C, sim(g, seed_C)) \mid C \in \mathcal{C}_{Fix} \wedge sim(g, seed_C) \geq \theta\}$   $\triangleright$  check if  $g$  belongs to
a fixed cluster
8:     if  $m \neq \emptyset$  then
9:        $assigned = true$   $\triangleright$  assign  $g$  to  $C$  with highest similarity  $sim(g, seed_C)$ 
10:    end if
11:    if  $assigned \neq true$  then  $\triangleright g$  not assigned to fixed cluster? Find pending ones
12:      for all  $C \in \mathcal{C}_{Pending}$  do
13:        if  $sim(g, seed_C) \geq \theta$  then
14:           $C.add(g)$ 
15:           $assigned \leftarrow true$ 
16:        end if
17:      end for
18:    end if
19:    if  $assigned \neq true$  then
20:       $\mathcal{C}_{Pending}.add(newCluster(g))$   $\triangleright$  create new pending Cluster
21:       $assigned \leftarrow true$ 
22:    end if
23:    for all  $C \in \mathcal{C}_{Pending}$  do  $\triangleright$  check if one pending cluster is over size threshold
24:      if  $|C| > max\_pending$  then
25:         $\mathcal{C}_{Pending}.remove(C)$ 
26:         $\mathcal{C}_{Fix}.add(C)$ 
27:         $seed_{C_f} \leftarrow determineFixedSeed(C)$ 
28:         $reuseGs = reassign(C, \mathcal{C}_{Pending}, graphs, \theta)$   $\triangleright$  reassign cluster members
29:         $removeFromPendingClusters(C, \mathcal{C}_{Pending})$ 
30:        for all  $r \in reuseGs$  do
31:           $graphs.enqueue(r)$ 
32:        end for
33:      end if
34:    end for
35:  end while
36:  for all  $C \in \mathcal{C}_{Pending}$  do
37:     $\mathcal{C}_{Pending}.remove(C)$ 
38:     $\mathcal{C}_{Fix}.add(C)$ 
39:     $removeFromPendingClusters(C, \mathcal{C}_{Pending})$ 
40:  end for
41:  return  $\mathcal{C}_{Fix}$ 
42: end procedure

```

---

**Algorithm 8** APreClus: Reassign step**Input:**  $C$  - cluster whose members needs to be reassigned $\mathcal{C}_{Pending}$  - pending clusters $\theta$  - similarity threshold ( $\theta \in [0,1]$ )**Output:**  $reuseGs$  - recycled graphs

```

1: procedure REASSIGN( $C, \mathcal{C}_{Pending}, \theta$ )
2:    $recycleInstance \leftarrow true$ 
3:   for all  $g \in C$  do
4:     if  $sim(g, seed_C) < \theta \cdot max(|g|, |seed_C|)$  then
5:        $C.remove(g)$ 
6:       for all  $C' \in \mathcal{C}_{Pending}$  do
7:         if ( $g \in C'$ ) then
8:            $recycleInstance \leftarrow false$ 
9:           break
10:        end if
11:       end for
12:       if  $recycleInstance$  then
13:          $reuseGs.add(g)$ 
14:       end if
15:     end if
16:   end for
17:   return  $reuseGs$ 
18: end procedure

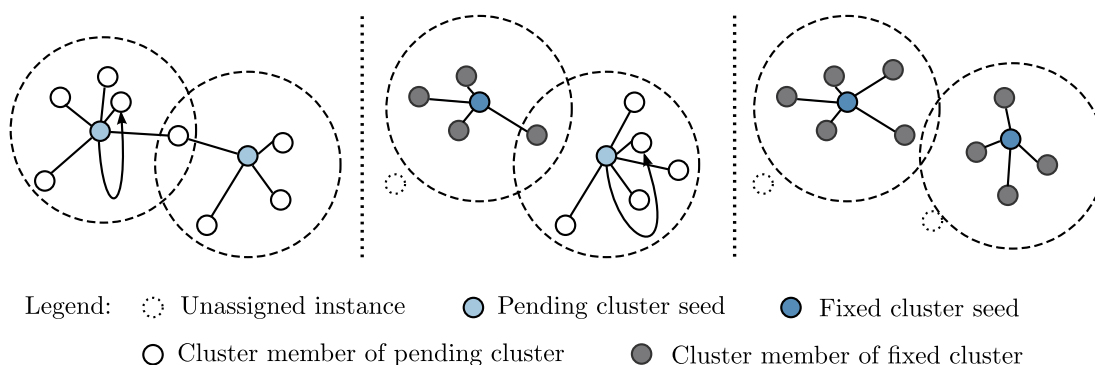
```

3.3.1.1.1 Initialization At the beginning of the clustering process, no clusters exist, and therefore the first input instance is determined to be the pending seed of the first pending cluster (Algorithm 7, line 3).

3.3.1.1.2 Pending cluster A pending cluster consists of a set of assigned instances and is represented by a cluster representative, called “pending seed”. Pending clusters are used at the early stages of cluster construction. Instance membership to a pending cluster is neither exclusive nor final. Once a pending cluster reaches a threshold size ( $max\_pending$ ), it is converted into a fixed cluster.

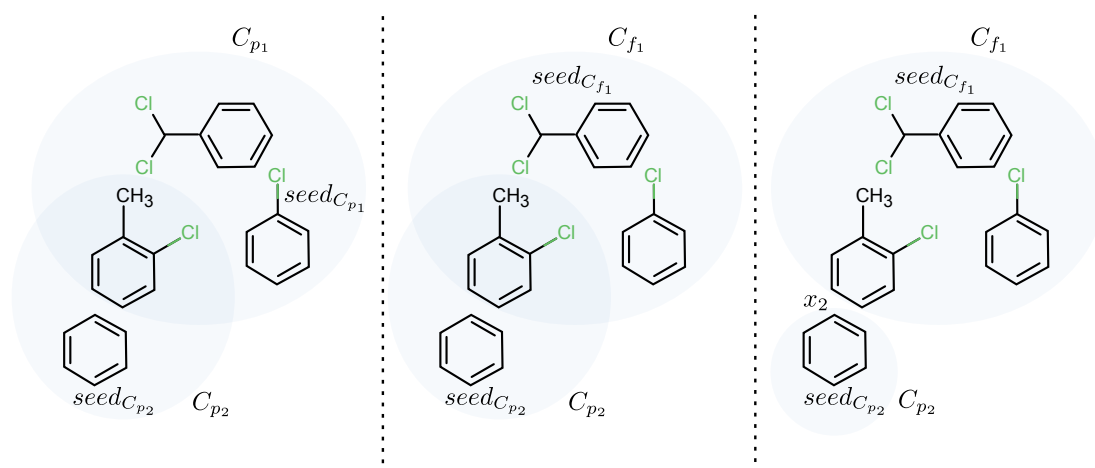
3.3.1.1.3 Fixed cluster A fixed cluster consists of a set of assigned instances and is represented by a cluster representative, called “fixed seed”. Instance membership to a fixed cluster is exclusive and final. A fixed cluster can still grow in size.

3.3.1.1.4 Pending cluster to fixed cluster conversion First, a fixed seed is determined as the assigned instance that maximizes a feature vector-based inner cluster link score, which is defined as the sum of common vertex labels shared between the selected instance and all other instance assigned to the respective cluster. Second, all instances in the pending cluster are reassigned to the new fixed seed in order to decide whether they are members of the new fixed cluster (Algorithm 7, line 28 and Algorithm 8). More precisely, the feature



**Figure 3.19:** Example of the pending cluster conversion process using two neighboring clusters reaching the threshold size  $max\_pending = 5$  (left). For both clusters, the fixed seeds and the fixed cluster members are determined (center and right). After the conversion process, the former pending seed of the cluster can be shifted to another member of the original pending cluster (see arrows).

vector-based similarity between the new fixed seed and each cluster member is calculated. In case the similarity is below a given threshold, the instance is removed from the cluster. If an instance is assigned to the new fixed cluster, its membership to any pending clusters is removed. Otherwise the instance is recycled as input instance if it is not assigned to any pending cluster (Algorithm 7, lines 30-31). Delaying the creation of fixed clusters through the usage of pending clusters has the advantage that instances initially inappropriately assigned to a cluster may find their representative seed. In case they were not assigned to other pending clusters they are recycled as new input instances. In other words, errors of assigning instances too early to a cluster can be corrected, even if an initial assignment to a cluster has been made. Figures 3.19 and 3.20 illustrate the cluster conversion process.



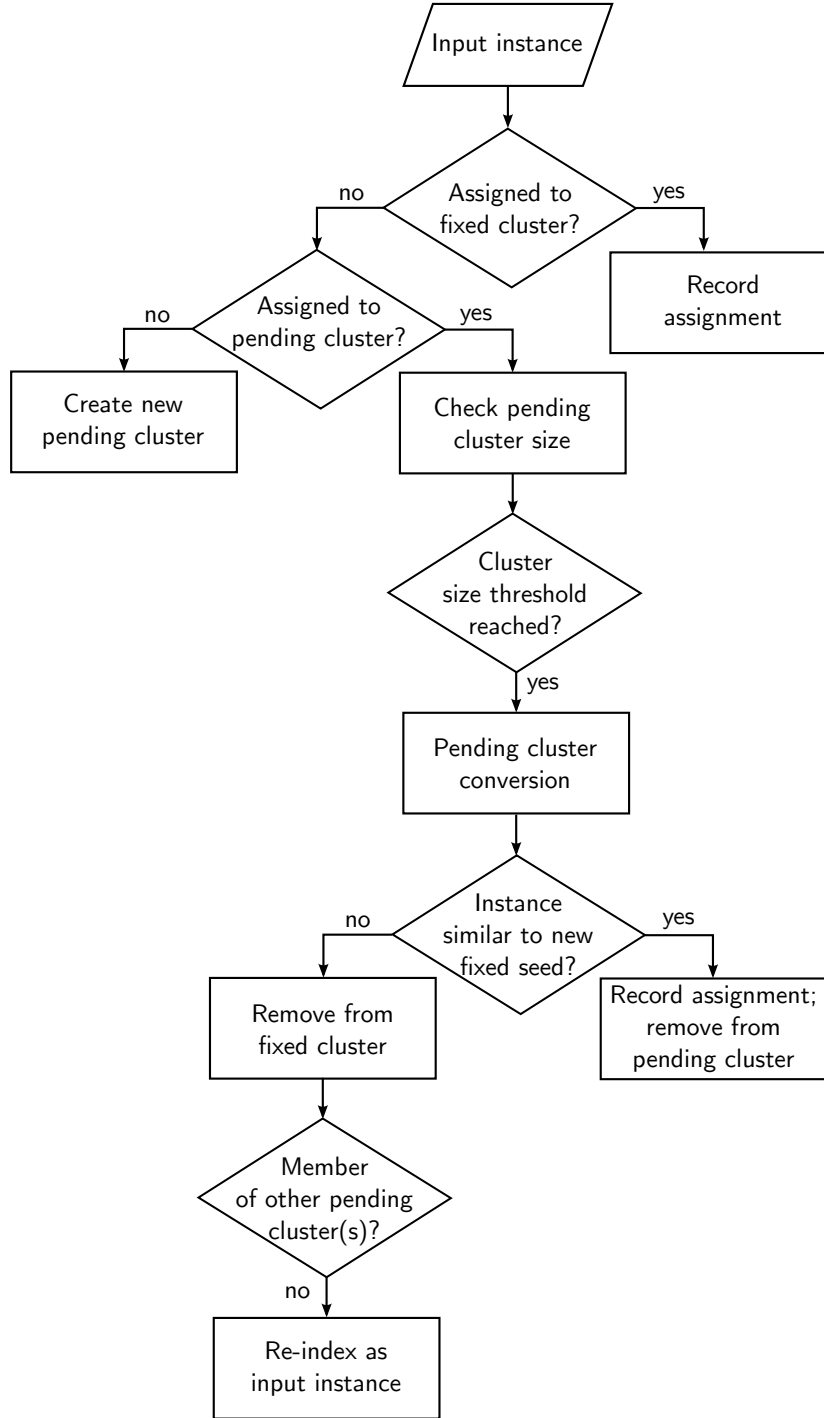
**Figure 3.20:** Example of the pending cluster conversion process using two neighboring clusters. The pending cluster  $C_{p_1}$  reaches the threshold size  $max\_pending = 3$  (left). Hence, for this cluster, the fixed seed and the fixed cluster members are determined (center). If a graph that is assigned to the new fixed cluster is also member of a pending cluster, it has to be removed from the pending cluster (right).

**3.3.1.1.5 Instance assignment** Consider a set of fixed clusters (each represented by a fixed seed) and a set of pending clusters (each represented by a pending seed). APreClus tries to assign the instances to the fixed clusters first (Algorithm 7, lines 7-10). For a given input instance the similarity to all fixed seeds is calculated in order to find the best representative seed. A set of tuples, with each tuple consisting of a cluster and the similarity to the respective cluster is returned for which the similarity is above a given threshold (line 7). APreClus then assigns the instance to the cluster with the highest similarity (line 9). Subsequently, each input instance that has not been assigned to any fixed cluster (line 8) is compared with all pending seeds (line 11) using the similarity measure based on the vertex feature vectors (line 13). An instance is assigned to all pending clusters for which the dissimilarity is below a given threshold (line 14). Each remaining instance that has not been assigned to any pending cluster is considered as a new pending seed (lines 19-20). If a pending cluster reaches the size of *max\_pending* cluster members (line 24), the conversion procedure described in Section 3.3.1.1.4 is triggered (lines 25-29).

**3.3.1.1.6 Termination** At the end of the clustering procedure, there might still be some instances assigned to pending clusters or as pending seeds. These clusters are sequentially converted to be fixed clusters and the member instances will be assigned to exactly one cluster (lines 36-39). When a pending seed is converted, the corresponding member instances are forced to be assigned to this instance and their memberships to other pending seeds are removed (line 39). The pending seeds are transferred to fixed seeds sequentially until all the instances are assigned.

**3.3.1.1.7 Abstraction-Based Similarity Calculation** The pre-clustering approach APreClus uses a similarity measure based on an abstraction from the actual structural similarity measure used by PSCG<sup>1</sup>. More precisely, the assignment procedure of APreClus calculates pairwise similarities based on a set-abstraction of graphs, i.e., an integer feature vector representing the number of vertex types in a graph. The underlying idea is that if two graphs are not similar with respect to their vertex label sets, they cannot be regarded similar with respect to their graph structure. In other words, given two graphs  $x_1$  and  $x_2$  and a similarity threshold  $\theta$ . If  $x_1$  and  $x_2$  do not share a common vertex set that covers a specific fraction  $\theta$  of both  $x_1$  and  $x_2$ , it is impossible that they share a common subgraph  $x$  that covers at least a fraction  $\theta$  of both  $x_1$  and  $x_2$ , i.e.,  $\nexists x : |x| \geq \theta|x_1| \wedge |x| \geq \theta|x_2|$ . For two graphs the common vertex set represents an upper bound for the size of the maximum common subgraph.

Formally, each graph instance  $x_i$  is represented by a numerical feature vector  $\mathbf{f}_{x_i} = (f_{x_i}^1, \dots, f_{x_i}^n)$  corresponding to a set of vertex types  $l_1, \dots, l_n$ . Each entry in  $\mathbf{f}_{x_i}$  records the number of a specific vertex type occurring in graph  $x_i$ . Let  $f_{x_i}^k$  denote the numerical feature associated with the vertex type  $v_k$ . Further, each cluster  $C_j$  is represented by the vector of its pending or fixed seed depending on whether the cluster is pending or fixed. Let  $\mathbf{f}_{seed_{C_j}} = (f_{seed_{C_j}}^1, \dots, f_{seed_{C_j}}^n)$  denote the feature vector of the cluster representative.

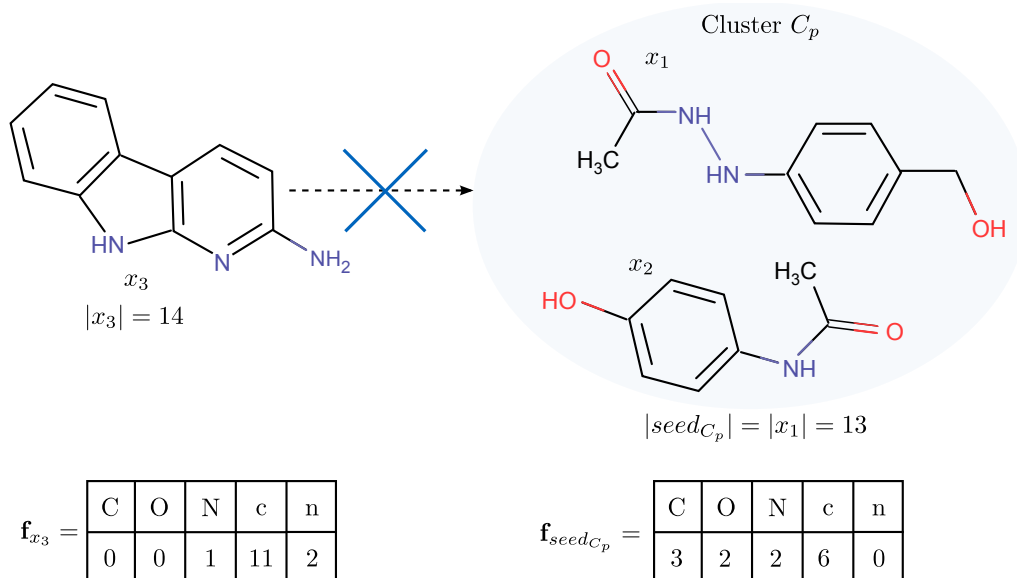


**Figure 3.21:** APreClus workflow

The similarity  $s$  between  $\mathbf{f}_{x_i}$  and  $\mathbf{f}_{seed_{C_j}}$  is computed by summing up the minimum of each pair of feature vector components:

$$s(\mathbf{f}_{x_i}, \mathbf{f}_{C_j}) = \sum_k (\min(\{\mathbf{f}_{x_i}^k \in \mathbf{f}_{x_i}\} \cup \{\mathbf{f}_{seed_{C_j}}^k \in \mathbf{f}_{seed_{C_j}}\})). \quad (3.7)$$

As mentioned earlier, the similarity between the feature vectors of two instances rep-



$$minSize = \theta \cdot \max(|seed_{C_p}|, |x_{m+1}|) = 0.6 \cdot (13, 14) \approx 9$$

$$s(\mathbf{f}_{x_3}, \mathbf{f}_C) = \min(0, 3) + \min(0, 2) + \min(1, 2) + \min(11, 6) + \min(2, 0) = 7$$

$$s(\mathbf{f}_{x_3}, \mathbf{f}_C) = 7 < 9 = minSize$$

**Figure 3.22:** Example illustrating the use of the abstraction-based similarity measure employed by APreClu ( $\theta = 0.6$ ). Since the similarity between the feature vectors of query graph  $x_3$  and the cluster representative  $seed_{C_p}$  is smaller than the minimum required size of the common subgraph,  $x_3$  won't be assigned to cluster  $C_p$ .

resents an upper bound on the size of the maximum common subgraph. Thus, if the normalized similarity  $\frac{s(\mathbf{f}_{x_i}, \mathbf{f}_{seed_{C_j}})}{\max(|x_i|, |seed_{C_j}|)}$  between an input instance and the cluster representative is lower than the similarity threshold,  $\theta$ , i.e.,  $\frac{s(\mathbf{f}_{x_i}, \mathbf{f}_{seed_{C_j}})}{\max(|x_i|, |seed_{C_j}|)} < \theta$ , both instances do not share a common subgraph that covers a specific fraction  $\theta$  of both instances. Figure 3.22 illustrates the use of the abstraction-based similarity measure for  $\theta = 0.6$ . In the pseudocode, the notation  $sim(x, C_{seed})$  is used to represent the feature vector based similarity  $s(\mathbf{f}_{x_i}, \mathbf{f}_{seed_{C_j}})$  between a graph  $x_i$  and a cluster representative  $seed_{C_j}$ .

### 3.3.1.2 PSCG'

This section presents the scaffold-based structural graph clustering algorithm PSCG'. The approach is a variant of the previously proposed scaffold-based parallel structural graph clustering algorithm PSCG. The pseudocode for PSCG' is shown in Algorithm 9. The main motivation behind the approach is to avoid cluster comparisons with all cluster members, which grow computationally more expensive with increasing cluster size, by defining a cluster representative for each cluster. The algorithm works as follows. Let  $\theta$  be the user-defined similarity coefficient and let  $minThresh$  be the minimum threshold for the size of the common subgraphs defined in Equation 3.2. In the first step, an initial cluster

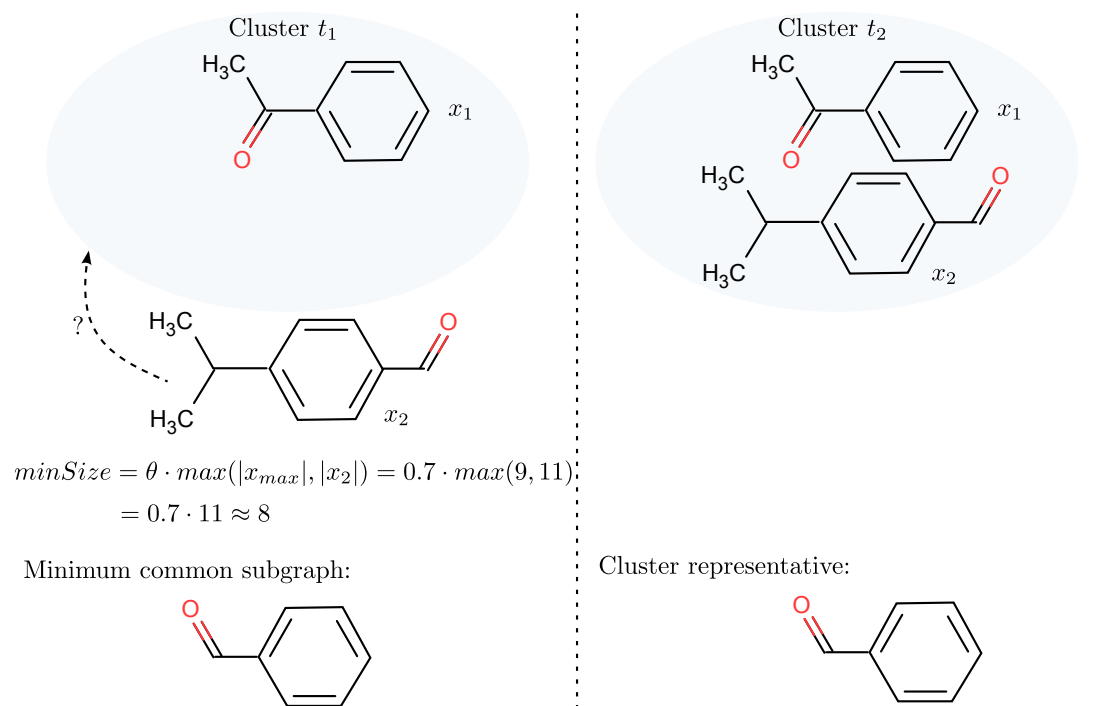


is created containing the first graph instance (see Algorithm 9, line 4). In the following steps, each graph instance is compared against all existing clusters. In case the query instance shares at least one common subgraph with one or more clusters that meets the cluster criterion in Equation 3.2 (line 12), the instance is added to the respective clusters (line 23). The first computed common subgraph is then taken as cluster representative (line 26) (see example in Figure 3.23). In the following, all subsequent graph instances are compared only against the cluster representative (line 17-21) instead of comparing them against all graphs in the cluster. Unlike many traditional clustering algorithms, an instance is allowed to belong to no cluster, since it is possible that an instance is not similar to any cluster. Thus, in this case, a new singleton cluster is created containing the query instance (line 32). The advantage of defining a cluster scaffold at an early stage is that the graph structures can be compared in parallel against a cluster, since the cluster scaffold is fixed and the assignment of an instance to an cluster has no influence on the subsequent cluster comparisons. Thus, substantial performance improvements may be achieved. By additionally sorting the data set in decreasing order of graph size (defined as the number of vertices) (line 2), the number of potential candidates is decreased for the cluster representative.

To reduce the number of expensive subgraph computations, the following two cluster exclusion criteria were employed to avoid unnecessary calls to the gSpan algorithm in the first place: a cluster membership test based on node feature vectors of graphs as used by APreClus (lines 14 and 18), and an exclusion criterion based on the size of graph instances resulting from the cluster criterion defined in Equation 3.2 (line 14). The criterion constrains the set of graphs being considered for clustering. More precisely, only graphs in a certain size range are considered for comparison with a specific cluster, i.e., graphs whose sizes lie in the range  $[\lceil \theta x_{max} \rceil, \lfloor \frac{1}{\theta} x_{min} \rfloor]$ , where  $x_{min}$  is the smallest and  $x_{max}$  is the largest graph instance in the cluster. Hence, by sorting the data set in decreasing order of graph size, graphs larger than  $\theta x_{max}$  do not need to be considered for cluster membership.

For computing common subgraphs, a modified version of the graph mining algorithm gSpan [245] that mines frequent subgraphs in a database of graphs satisfying a given minimum frequency constraint was used. A minimum support threshold of  $minSup = 100\%$  in a set of graphs is required, i.e., all common subgraphs have to be embedded in all cluster members. For the experiments with molecular graph data, gSpan', an optimization of the gSpan algorithm for mining molecular databases [115], is used. Since it is not necessary to compute all common subgraphs in a set of graphs, but it is only important to know if there exists at least one common subgraph that meets the minimum size threshold defined in Equation 3.2, it is possible to terminate the subgraph mining process once a solution is found. In this way, a substantial improvement in runtime performance can be achieved. This modification of gSpan' is called gSpan'' as used in the pseudocode.

PSCG' is parallelized to take advantage of high-performance parallel hardware. The approach adopts a *master-worker paradigm* consisting of two kinds of entities: a single master and multiple workers. The master is responsible for decomposing a clustering



**Figure 3.23:** Example illustrating the definition of a cluster representative (for  $\theta = 0.7$ ). Left: Cluster at time  $t_1$ . To be assigned to the cluster, the query graph  $x_2$  needs to share at least one common subgraph with the cluster member  $x_1$  that meets the  $\minSize$  threshold defined in Equation 3.2. As  $x_2$  shares such a subgraph with  $x_1$  that meets the  $\minSize$  threshold of 8,  $x_2$  is assigned to the cluster. Right: Cluster at time  $t_2$ . The minimum common subgraph is taken as cluster representative. In the following, all subsequent graphs are compared only against the cluster representative.

problem into a subset of clustering tasks and distributing them among a farm of workers (by putting the tasks in a shared queue), as well as for gathering the partial results in order to produce the final computation result. A queue, shared between the master and workers, is used to represent the shared space where the pending clusters reside. Each worker is responsible for only one cluster at any point in time, independently computing one iteration: It pulls a clustering task (input) from the queue, processes the task by comparing all relevant graphs in the graph database against the cluster, and sends the result, i.e., the processed cluster, back to the master (output). Once a cluster representative is defined, the cluster computations against the respective cluster can be done in parallel. All idle workers, i.e., all workers that are waiting for the creation of a new cluster will be responsible for this task, hence providing better utilization and efficiency than PSCG.

### 3.3.2 Experimental Results

To evaluate the effectiveness and efficiency of SCAP, several experiments were conducted on a number of publicly available data sets of molecular graphs. This section describes the experimental setup, the baseline methods, the data sets and the results.

**Algorithm 9** PSCG'

**Input:** *graphs* - queue of  $n$  graphs to be clustered  
 $\theta$  - similarity threshold ( $\theta \in [0,1]$ )

**Output:** Clusters *clusters*

```

1: procedure PSCG'(graphs,  $\theta$ )
2:   sort(graphs)
3:   clusters  $\leftarrow \emptyset$ 
4:    $C = \text{newCluster}(\text{dequeue}(\text{graphs}))$ 
5:    $C.\text{useScaffold} = \text{false}$ 
6:   clusters.add( $C$ )
7:   for all  $g \in \text{graphs}$  do
8:     assigned = false
9:     for all  $C \in \text{clusters}$  do
10:      scaffold = null
11:      minThresh = 0
12:       $\text{minSize} \leftarrow \theta \cdot \max(|g|, |\max(C)|)$   $\triangleright$  calculate the minimum threshold thr
13:      if  $\neg C.\text{useScaffold}$  then
14:        if  $|g| \geq \theta \cdot \max(C) \ \& \ \text{sim}(g, C) \geq \text{thr}$  then
15:           $\text{scaffold} \leftarrow \text{gSpan}''(g \cup C, \text{thr})$   $\triangleright$  returns common scaffold if existent,
16:        end if
17:      else
18:        if  $\text{sim}(g, C.\text{scaffold}) \geq \text{thr}$  then
19:           $\text{scaffold} \leftarrow \text{gSpan}''(g \cup C.\text{scaffold}, \text{thr})$ 
20:        end if
21:      end if
22:      if scaffold  $\neq$  null then
23:         $C.\text{add}(g)$ 
24:        assigned  $\leftarrow$  true
25:        if  $\neg C.\text{useScaffold}$  then
26:           $C.\text{scaffold} \leftarrow \text{scaffold}$ 
27:           $C.\text{useScaffold} \leftarrow \text{true}$ 
28:        end if
29:      end if
30:    end for
31:    if assigned = false then
32:       $C \leftarrow \text{newCluster}(g)$ 
33:      clusters.add( $C$ )
34:       $C.\text{useScaffold} = \text{false}$ 
35:    end if
36:  end for
37:  return clusters  $\triangleright$  return clusters
38: end procedure

```

## 3.3.2.1 Baseline Methods

The scaffold-based clustering approach SCAP was compared against the following methods.

**3.3.2.1.1 PSCG** Similar to SCAP, PSCG [202] is a scaffold-based structural graph clustering algorithm that produces overlapping (non-disjoint) and non-exhaustive clusters. Contrary to SCAP, PSCG does not employ a pre-processing clustering step. Further, each graph in PSCG is compared against all cluster members unless the common cluster scaffold is unique. On the other hand, SCAP defines a cluster scaffold at an early stage in the clustering. Hence, PSCG requires a much larger number of expensive subgraph search computations compared to SCAP.

**3.3.2.1.2 DP Clustering** SCAP was compared with a graph-based clustering based on variational Dirichlet process (DP) mixture models and frequent subgraph mining by Tsuda and Kurihara [225]. The clustering approach addresses the problem of learning a DP mixture model in the high dimensional feature space of graph data.

**3.3.2.1.3 Fingerprint-Based Clustering** SCAP was compared against two fingerprint (FP)-based clusterings. These approaches cluster data sets of graphs using an FP-based similarity measure. For the first approach, the chemical fingerprints provided by Chemaxon's JChem Java package were used [118]. These fingerprints are equivalent to the Daylight fingerprints<sup>4</sup>. The Tanimoto coefficient is used as similarity measure between fingerprints which was shown to work well in combination with fingerprints [213]. The first FP clustering method is based on an incremental clustering approach. Iteratively, each graph is compared against all existing clusters. In case the Tanimoto similarity  $\tau$  between the query graph and each graph in a cluster exceeds a user-defined threshold, the query graph is added to the respective cluster; otherwise a new singleton cluster is created containing the query graph. The second approach is the well-known iterative k-means clustering approach BIRCH<sup>5</sup> [254] using molecular fingerprints generated with Openbabel<sup>6</sup> to cluster the graph data. As distance measure the Tanimoto distance was used. BIRCH requires two input parameter: the distance threshold  $t$  and the tree branching factor  $b$ .

### 3.3.2.2 Qualitative Analysis

A cluster analysis was performed on two data sets whose target values are categorical. The first data set consists of 107 standard anti-cancer agents (SACA) whose class labels corresponding to their mechanisms of action have been clearly classified [135, 237]. The second data set is a QSAR data set containing 2637 graph structures from five series of compounds: cyclooxygenase-2 (COX-2) inhibitors, benzodiazepine receptor (BZR) ligands, estrogen receptor (ER) ligands, dihydrofolate reductase (DHFR) inhibitors, and monoamine oxidase (MAO) inhibitors [216]. The purpose of the experiment is to test if the clusters obtained by SCAP agree well with the corresponding known class labels. The

---

<sup>4</sup> <http://www.daylight.com/dayhtml/doc/theory/theory.finger.html>

<sup>5</sup> The implementation available at <http://roberto.perdisci.com/projects/jbirch> was used.

<sup>6</sup> <http://openbabel.sourceforge.net>

**Table 3.7:** Clustering results for SCAP and PSCG.

Data set	Method		$\theta = 0.4$	$\theta = 0.5$	$\theta = 0.6$
SACA	SCAP	# clusters	32	27	24
		# singletons	18	29	39
		cluster purity	0.803	0.884	0.912
		Omega Index	0.164	0.310	0.362
	PSCG	# clusters	28	26	23
		# singletons	12	22	29
		cluster purity	0.720	0.865	0.911
		Omega Index	0.156	0.277	0.372
	SCAP <sub>CluFit</sub>	# clusters	32	28	22
		# singletons	14	23	29
		cluster purity	0.766	0.849	0.920
		Omega Index	0.151	0.360	0.366
QSAR	SCAP	# clusters	141	254	367
		# singletons	8	16	48
		cluster purity	0.800	0.882	0.892
		Omega Index	0.196	0.117	0.093
	PSCG	# clusters	123	178	245
		# singletons	8	19	38
		cluster purity	0.777	0.813	0.851
		Omega Index	0.178	0.110	0.089
	SCAP <sub>CluFit</sub>	# clusters	107	156	242
		# singletons	7	6	25
		cluster purity	0.762	0.833	0.865
		Omega Index	0.198	0.124	0.123

approach was further compared to PSCG [202]. As an external measure for clustering validation two metrics were used: cluster purity and the Omega Index [55]. For each cluster, purity is defined as the ratio between the number of graphs by the dominating class in that cluster and the total number of graphs in that cluster. Formally,

$$\text{purity}(C,L) = \frac{1}{N} \sum_k \max_j |C_k \cap l_j| \quad (3.8)$$

where  $C = \{C_1, C_2, \dots, C_K\}$  is the set of clusters and  $L = \{l_1, l_2, \dots, l_J\}$  is the set of classes. Here,  $C_k$  is interpreted as the set of graphs in  $C_k$  and  $l_j$  as the set of graphs in  $l_j$  in Equation 3.8. Larger cluster purities correspond to better agreement between the clustering results and the class labels, with 1.0 indicating perfect concordance. Since the index favors small clusters, with the degenerate case of a singleton cluster resulting in a maximal cluster purity score, singleton clusters were excluded in the purity calculation. The Omega Index generalizes the adjusted Rand index to compare non-disjoint cluster memberships. In case of disjoint clustering solutions, the Omega Index is identical to the

original adjusted rand index. The highest possible score of 1.0 indicates that two solutions perfectly agree on how each pair of graphs is clustered. Table 3.7 shows the clustering results for different values of  $\theta$  for the comparison methods. For a fair comparison with PSCG, the similarity threshold  $\theta$  was additionally adjusted in the second clustering stage (i.e.,  $\theta$  was lowered by 0.05 on the SACA data and by 0.1 on the QSAR data) such that approximately the same number of clusters and singletons as PSCG are achieved. In the result tables, the modification is referred to as SCAP<sub>CluFit</sub>. The results show that SCAP is able to discriminate the known structure classes in both data sets very well. Graphs in the same cluster not only share common subgraphs but are also strongly associated with specific classes. Across different values for  $\theta$  it was observed that for higher values a finer but cleaner grouping of the structures at the cost of generating a larger number of smaller clusters is achieved. The graphs in each class are more cleanly discriminated from other graphs in the data set.

### 3.3.2.3 Comparison to FP Clustering

SCAP was further compared with the FP clustering methods described in Section 3.3.2.1.3. For the experiments with BIRCH, the distance threshold was varied from 0.4 to 0.7. The branching factor was left at its default value. Tables 3.8 and 3.9 show the results for different parameter settings. The results demonstrate that the incremental FP method is not able to discriminate the known structure classes in the SACA data very well, indicated by low values of cluster purity and the Omega Index. Increasing the value of the Tanimoto threshold results in a finer but cleaner grouping of the structures at the cost of generating a larger number of singletons. Comparing the BIRCH clustering approach to SCAP, the values for the Omega Index are observed to be constantly higher for SCAP. However, for some parameter settings the clusters produced by BIRCH result in larger cluster purity values. This might be due to the fact that BIRCH generates many more clusters of smaller size. Forcing SCAP to produce a comparable number of clusters by increasing  $\theta$  (see Table 3.10) results in higher values for both validation metrics. The results show the advantage of considering the graph topology for clustering. Clustering approaches based on fingerprints, on the other hand, suffer to some extent from a loss of information with respect to the actual graph topology. Overall, the results suggest that there is a clear advantage from the use of the more sophisticated scaffold-based clustering algorithm SCAP, emphasizing the need to develop more complex algorithms that consider the structure a graph directly, as SCAP does.

### 3.3.2.4 Comparison to DP Clustering

SCAP was compared with a graph-based clustering based on variational Dirichlet process (DP) mixture models and frequent subgraph mining by Tsuda and Kurihara [225] (see Section 3.3.2.1.2 for a description). The goal of the experiment is to investigate if the DP clustering approach is also able to rediscover the known structure classes in the SACA

**Table 3.8:** Clustering results for the incremental FP clustering ( $\tau$ : Tanimoto similarity threshold).

Data set		$\tau = 0.4$	$\tau = 0.5$	$\tau = 0.6$
SACA	# clusters	19	18	15
	# singletons	24	36	43
	cluster purity	0.649	0.726	0.879
	Omega Index	0.043	0.113	0.301
QSAR	# clusters	118	195	293
	# singletons	22	42	76
	cluster purity	0.741	0.792	0.820
	Omega Index	0.160	0.103	0.080

**Table 3.9:** Clustering results for BIRCH ( $t$ : distance threshold).

Data set		$t = 0.7$	$t = 0.6$	$t = 0.5$	$t = 0.4$
SACA	# clusters	25	28	26	26
	# singletons	21	32	42	47
	cluster purity	0.821	0.879	0.908	0.882
	Omega Index	0.109	0.094	0.072	0.055
QSAR	# clusters	596	725	886	943
	# singletons	20	50	98	150
	cluster purity	0.880	0.887	0.897	0.910
	Omega Index	0.010	0.007	0.005	0.005

**Table 3.10:** Clustering results for SCAP using  $\theta = 0.7$ .

Data set	# clusters	# singletons	cluster purity	Omega Index
SACA	20	48	0.949	0.157
QSAR	634	166	0.917	0.120

database described in Section 3.3.2.2. In this experiment, the parameters required by DP clustering were chosen as follows. The number of features  $m$  was varied from 50 to 5000 and set  $\alpha = 0.01, 0.1, 1, 10$ . Table 3.11 shows the experimental results for  $\alpha = 0.1$ . The results for  $\alpha = 0.01, 1, 10$  are similar. The number of clusters were observed to increase along with the number of features for  $m \leq 500$ ; for  $m > 500$  the number of clusters decreases significantly. Compared to SCAP, DP clustering produces less clusters. In order to make the results more comparable to the results of the method, the user-specified parameters were varied. Nonetheless, it was not possible to parameterize the DP clustering method to obtain more than seven clusters. The DP clustering results indicate that the method is not able to discriminate the known structure classes in the SACA data set very well. The DP clusters are, in many cases, associated with different structure classes, indicated by

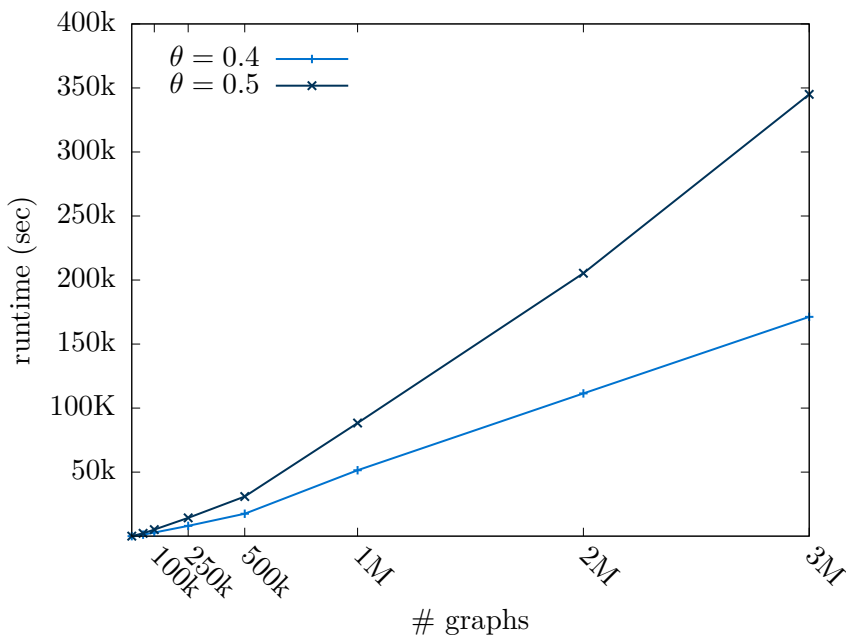
lower values for cluster purity and the Omega Index (Table 3.11). The results demonstrate that SCAP is better able to effectively grouping the 107 agents.

**Table 3.11:** Clustering results for DP Clustering.

# Features	50	100	500	1000	5000
# Clusters	6	7	7	6	2
Cluster purity	0.421	0.411	0.579	0.589	0.402
Omega Index	0.119	0.106	0.274	0.281	0.056

### 3.3.2.5 Experiments on Large Graph Data Sets

To study the scalability of SCAP, experiments were performed on different sized data sets. The data sets are subsets of the ChemDB database, containing nearly 5 million commercially available small molecules [48, 49]. Seven data sets were created from ChemDB of size 50K, 100K, 250K, 500K, 1M, 2M and 3M graphs, respectively, using random sampling. The mean graph size of these data sets is 27, the maximum graph size 435. The experiments were carried out on a SUN x4600 system with 32 AMD Opteron CPU cores (8 CPU sockets with 4 cpu cores). The processor in each node runs at 2.5 GHz with 2 GB of main memory.

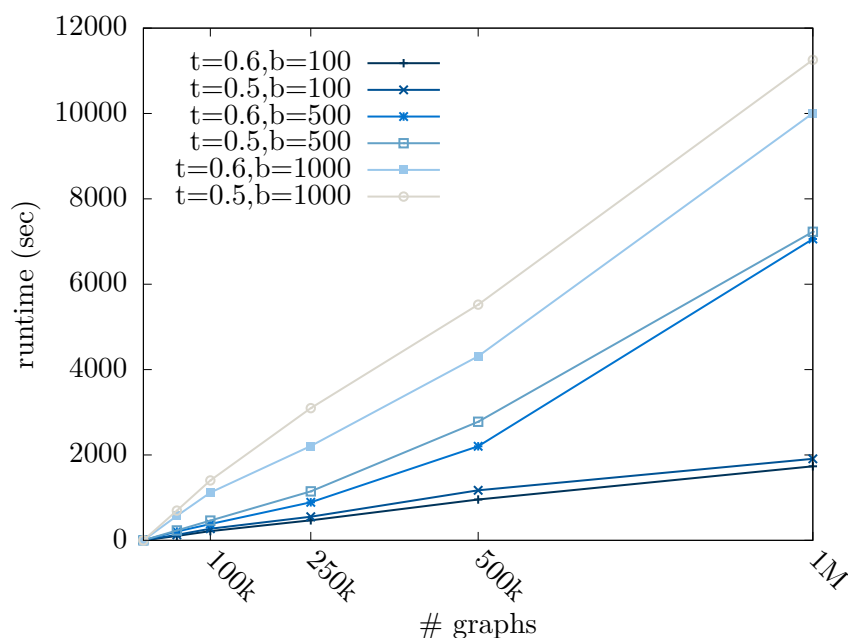


**Figure 3.24:** Runtime performance of the clustering approach SCAP on the data sets sampled from the ChemDB database comprising 100k to 3M graphs.

Figure 3.24 shows the runtime performance on the sampled data sets for  $\theta = \{0.4, 0.5\}$ . These values are suitable parameter settings under the aspect of producing a reasonable number of clusters. Whereas a too small value of  $\theta$  results in large, heterogeneous clusters, a too large value of  $\theta$  produces very few, small clusters or no clusters at all. Table 3.12



also reports the number of partitions produced by APreClus as well as the total number of clusters produced by SCAP for the three largest data sets. Compared to previous results of PSCG where the authors report that 300,000 graphs were clustered in 610,577 seconds for  $\theta = 0.4$  using a similar architecture, SCAP is able to cluster an order of magnitude more graphs in only a fraction of time. The results indicate that SCAP is able to handle large data sets containing millions of graphs, in reasonable time. SCAP is therefore well capable of structuring large graph databases such as real-world compound libraries as employed for virtual screening. The runtime of SCAP was also compared against BIRCH [254]. BIRCH requires two input parameter: the distance threshold  $t$  and the tree branching factor  $b$ . Figure 3.25 shows the runtime of BIRCH on the data sets up to size 1M for different parameter configurations. The results demonstrate the trade-off between precision and computation. Whereas the BIRCH approach is computationally more efficient, the results of the qualitative analysis presented in Section 3.3.2.3 are in favor of SCAP.



**Figure 3.25:** Runtime performance of BIRCH on the data sets sampled from ChemDB.

### 3.3.3 Conclusion

In this section, a scaffold-based graph clustering approach was presented that is able to cluster large databases containing millions of graphs. SCAP first employs a pre-clustering based on dynamic seed clustering to partition the data set into several smaller data sets using an abstraction-based similarity measure. In the second clustering stage, the resulting partitions are further clustered into a finer level of granularity using a variant of a recently proposed parallel structural graph clustering algorithm. Several experiments were designed to evaluate the effectiveness and efficiency of the approach on various real world data sets of molecular graphs. First of all, the results indicate that the clustering method is able

**Table 3.12:** Runtime (in sec), number of #partitions produced by APreClus and number of clusters generated by SCAP on the three large data sets sampled from ChemDB.

Data set	$\theta$	runtime (sec)	#partitions APreClus	#clusters SCAP
1,000,000	0.4	51,475	159	22,453
	0.5	88,332	211	41,549
2,000,000	0.4	111,443	181	25,020
	0.5	205,333	253	52,406
3,000,000	0.4	171,200	227	29,612
	0.5	345,034	290	69,441

to rediscover known structure classes in two molecular graph data sets. Second, the results on three large graph data sets show that SCAP is able to handle a much larger number of graph instances compared to the previously proposed structure-based clustering algorithms, i.e., an order of magnitude more instances in a fraction of time required before. Given these performance improvements, the proposed algorithm is applicable to the large structure databases used for virtual screening. To the best of my knowledge, it is for the first time possible to cluster millions of compounds within a reasonable time using an accurate scaffold-based similarity measure. This represents a step towards structuring chemical space which could be defined by the structures of PubChem [176] currently containing approximately 49.5 million chemical compounds. The presented approach is general enough to be applicable not only in the domain of molecular graphs, but also in the domain of general (non-molecular) graphs.

# CHAPTER 4

---

## Maximum Common Subgraph Based Locally Weighted Regression

---

This chapter investigates a simple, yet effective method for regression on graphs, in particular for applications in cheminformatics and for QSARs. The method combines Locally Weighted Learning (LWL) with MCS-based graph distances. More specifically, a variant of locally weighted regression on graphs (structures) is investigated that uses the maximum common subgraph for determining and weighting the neighborhood of a graph and feature vectors for the actual regression model. It is shown that this combination, LWL-MCS, outperforms other methods that use the local neighborhood of graphs for regression. The performance of this method on graphs suggests it might be useful for other types of structured data as well.

### 4.1 Introduction

The idea of using local neighborhoods of test instances to improve prediction quality has a long tradition in statistics and machine learning [52]. Recently, local learning methods have experienced renewed interest in the form of local models, i.e., high-quality models of small regions of the input space that often have the advantage of being easier to predict and easier to interpret by domain experts [192]. Several local models together can make up a global model, or global models are the fallback solution (default) when no local model is applicable. A more traditional, but related approach is locally weighted learning [10]. Although it has been known to be highly effective for regression for a long time, the combination with other learning schemes, for instance, classification, has not been investigated thoroughly. Compared to local models, one would expect that locally weighted learning is more time-consuming, but also potentially worth the effort, because it should be able to adjust its predictions to the specifics of test instances.

Quite surprisingly, both locally weighted learning and local models have not been studied systematically yet for structured data like graphs, although the approach makes sense for at least two reasons. First, it makes sense to structure the input space before any predictive model is inferred, because the structural composition and diversity of typical data sets for graph classification and regression have a serious impact on the predictive performance of

methods. A closer look reveals that there exist structural islands in many data sets, i.e., subsets of instances that share a large common structural scaffold. Second, for structured objects, local models or local neighborhoods are an opportunity, because the wealth of possible descriptors and similarity/dissimilarity measures enable the use of one view for determining the neighborhood, and a different view for actually making the prediction. For these reasons, and because there is nothing yet known about the effectiveness of this method, this chapter studies locally weighted regression on graphs and compares it experimentally to other methods using local neighborhoods.

The method is studied in the context of so-called QSARs, i.e., models relating chemical structure to biological activity. Both local neighborhoods of test instances (chemical structures) and structural islands for local models are defined by the size of common subgraphs. More specifically, an MCS-based distance measure is used for locally weighted learning, whereas a common structural scaffold with a minimal size is required for the local models. The actual predictive models are then built using a feature-vector representation of the chemicals where the features encode standard chemical descriptors. Combining structural and non-structural descriptors in this way makes particular sense for the envisaged application area, because the MCS retrieves structures with a large common structural scaffold and is thus better suited to measure structural similarity between molecules, whereas chemical descriptors are better suited to determine the actual biological activity of structurally similar molecules. In a large-scale experimental comparison, it is shown that this learning scheme consistently outperforms other learning schemes using local graph neighborhoods.

The chapter is organized as follows: After discussing related work in Section 4.2, MCS-based locally weighted regression is presented in Section 4.3. Section 4.4 discusses experimental results quantitatively and qualitatively, before Section 4.5 concludes the chapter.

## 4.2 Related Work

The idea of local learning and local models has been around in statistics for a long time. One of the pioneers of using and weighting information close to a test (query) instance was William S. Cleveland who, in 1979, proposed a method for estimating non-linear regression models in this way [52]. This method, called LOESS, also known as locally weighted polynomial regression, was further developed by Cleveland and Devlin in the 1980s. The basic concept of LOESS is to fit a polynomial using weighted least squares by giving more weight to objects close to the test instance. In the subsequent section, locally weighted (linear) regression is extended towards graph neighborhoods defined in terms of maximum common subgraphs. More recently, a method combining local and global models was proposed by Rüping [192]. In his study, a local model algorithm is presented that learns a global classifier plus local models. The goal is to reduce the complexity of the global model, ensure the prediction quality of the combined model and provide guarantees that the combined and global model will differ only up to a user-specified degree. This chapter experimentally compares locally weighted learning on graphs with

local models defined in terms of structural graph clusters [38]. The clusters are obtained by the previously proposed clustering method, PSCG. Another family of methods making use of local neighborhoods are kernel methods. Kernels defined on graphs [87] typically compare sets of common graph elements like chains, trees or subgraphs. As the goal of this work is to exploit large structural scaffolds in structure databases by the maximum common subgraph, locally weighted learning on graphs is compared with the empirical kernel map based on the MCS, because it is the closest kernel method possible. The idea behind the Empirical Kernel Map [222] is to redefine each input instance as the vector of similarities to all training examples. To show the advantage of combining structural and non-structural descriptors, this chapter further investigates the empirical kernel map based on both the MCS and a set of non-structural descriptors.

Another topic of this chapter, the integration of multiple sources of information for learning, has been approached in several ways. It is a central topic in inductive logic programming (ILP) and statistical relational learning (SRL) [69]. In most approaches, facts and rules from multiple sources are encoded in a logic programming representation as part of the background knowledge. However, typically no distinction is made between different types of data, and all parts of the background knowledge are treated equally. The only exception are approaches that distinguish between the logical structure (e.g., of clauses) and the numerical values of variables separately, like the NuRMI system by Alphonse *et al.* [7]. It is, at this point, unclear how this could be transferred to the setting described in this chapter.

In multi-view learning (and, similarly, co-training) [23], groups of features are usually taken into account alternately, where intermediate results are adjusted incrementally. This is different from the LWL approach that consists of exactly two consecutive steps. After considering the numerical features (physicochemical properties), it would not make sense to go back to the graph view again in this setting, because the main idea is to refine the predictions using those features once structural information is exploited already. The assumption is that structural similarity is not sufficient anymore at one point, and other, distinct information (e.g., in the form of physicochemical properties) is required to differentiate further. Also, not the whole, uniformly weighted data set is used in both views, but weighted neighborhoods of individual instances. Moreover, LWL-MCS belongs, different from most multi-view learning methods, to the field of lazy learning: the structural neighborhood of a test instance is determined individually, on demand, at testing time.

The work is also related to methods combining different kernels (multiple kernel learning [133]) or distance or similarity measures [190]. Most of the approaches to multiple kernel learning focus on classification and positive semi-definite kernels, whereas this work focuses on regression and wishes to incorporate information about the maximum common subgraph, which does not naturally give a valid kernel (see above). As for distance learning, a comparison with a recently proposed method [190] was performed and it was found that it is substantially outperformed by LWL-MCS. Since the focus of the experimental comparison in this chapter is on other methods that take into account structural

neighborhoods of graphs, the results of this preliminary comparison are not shown.

## 4.3 Method

### 4.3.1 Notation and Definitions

As opposed to the other chapters of this thesis, in this chapter, the size of a graph is defined by the number of its edges, i.e.,  $|E|$ . Given two graphs  $g_1$  and  $g_2$ , a graph  $g$  is called a Maximum Common Subgraph (MCS) of  $g_1$  and  $g_2$ , denoted by  $MCS(g_1, g_2)$ , if  $g$  is a common subgraph of  $g_1$  and  $g_2$  and there exists no other common subgraph of  $g_1$  and  $g_2$  with more edges than  $g$ .

The MCS between two graphs can be classified further by distinguishing between the *connected* and *disconnected* case. A connected MCS is an MCS where each vertex is connected to every other vertex by at least one path in the graph (i.e., the MCS consists of a single subgraph). A disconnected MCS comprises one or more subgraph components. In this chapter, the MCS is assumed to be disconnected.

### 4.3.2 MCS Algorithm

Computation of the MCS has received considerable attention in theoretical computer science. Conte *et al.* [57] give a comprehensive overview of the various algorithms. When processing chemical compounds, certain characteristics allow for fast processing in most cases, even though in general MCS is NP-complete. Cao *et al.* [46] explore these options in great detail. Generally the graphical 2D structures of chemical compounds are sparse graphs, where the average number of incident edges at each node is very low. The most common atoms are carbon with at most four bonds or edges, nitrogen with at most three bonds and oxygen with at most two bonds. Additionally, all nodes and all edges are labelled by atom and bond types respectively, further limiting the number of options for computing matching sub-graphs in a pair of compounds. Most problematic in terms of runtime are multiply connected ring structures. Still, as was mentioned in [198], many compounds (e.g. about 95% of all compounds in the NCI collection) are actually outer-planar graphs, for which polynomial-time algorithms exist.

For the experiments reported below all MCS-based similarity matrices were precomputed using an approximate depth-first search-based algorithm [46] with a reasonably low timeout value (see Table 4.1). Precomputation is the most efficient option, as similarities for all pairs of compounds are only computed once, but can be re-used over and over again in multiple experiments. Assuming the size of the MCS can be represented by a 16 bit number, a short, a fully precomputed similarity matrix for a data set of 100,000 compounds will need about 10 gigabytes of memory, an amount easily available in server class machines these days. The data sets used in this study are actually much smaller than that, so memory is not an issue at all. On the other hand, large pharmaceutical companies have data sets comprising a few million compounds. While precomputation in memory

is not an option for such large data sets, precomputation could still be done using disk space. The similarity matrix for one million examples would need about one terabyte of disk space. Thanks to the way LWL works, access would not be random, but sequential, retrieving the similarities for all compounds versus a single target compound at a time. This sequential or streaming type of access to disk-based data is still rather efficient on modern hardware. If even larger data sets would need to be handled, a hybrid approach could be employed where precomputation identifies the “expensive” pairs and caches only these, and all “cheap” similarity computations would be done on demand.

To give some indication of the runtimes seen during precomputation, Table 4.1 lists the percentage of similarity computations that needed more than 10 milli-seconds to finish, as well as those that needed more than 1.28 seconds to finish, for each of the data sets used in the experiments reported later. It is very obvious that there are large differences between these data sets, as the figures vary between 5% and 50% for the 10 ms limit, and between less than 1% and up to 15% for the 1.28 second limit. These observations relate well to the number of compounds comprising multiply connected ring structure, which were mentioned earlier as the main source of inefficiency in MCS computation.

**Table 4.1:** Percentage of MCS computations that do not finish within either ten milliseconds or 1.28 seconds for each of the data sets used in the experiments reported later.

Data set	10 ms	1.28 secs
4QSAR COX2	44.355	9.641
4QSAR DHFR	31.640	4.257
CPD MOUSE	4.669	0.707
CPD RAT	4.873	0.821
FDAMDD	34.903	12.668
ISS MOUSE	6.037	0.768
ISS RAT	5.799	0.971
Suth COX2	50.497	14.793
Suth DHFR	42.675	9.149
Suth ER_TOX	45.107	7.626

### 4.3.3 MCS-based Distance Measure

This section presents the MCS-based distance measure that is used by the LWL approach to compute the neighborhood of a test instance and to weight the instances in this neighborhood. For a given graph data set  $D$ , the  $MCS(g_i, g_j)$  is calculated from all possible graph pairs  $g_i$  and  $g_j$  in  $D$  using the method described in Section 4.3.2. The pairwise MCSs between the graphs are represented in a symmetric matrix  $M(D)$ . The  $(i, j)$ th matrix element of  $M(D)$  is given by

$$M(D)_{i,j} = \begin{cases} |MCS(g_i, g_j)| & \text{if } i \neq j, 1 \leq i, j \leq |D| \\ |g_i| & \text{if } i = j \end{cases} \quad (4.1)$$

The size of a graph  $g_i$  is denoted by  $|g_i|$ . The similarity measure by Wallis [232] is used to define the similarity between two graphs

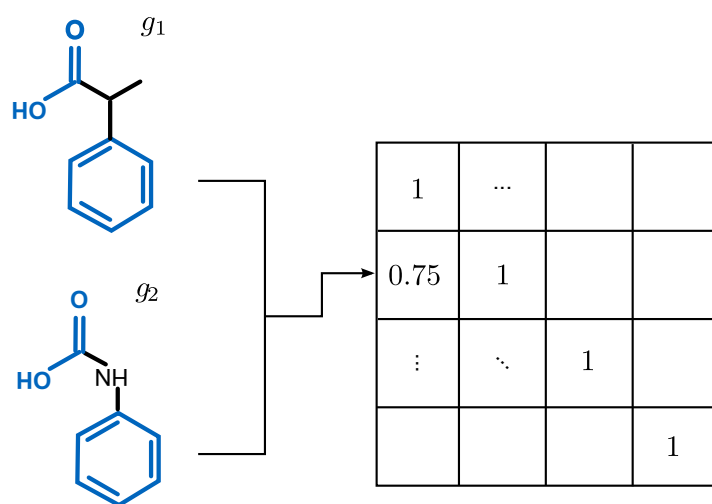
$$s(g_i, g_j) = \frac{|MCS(g_i, g_j)|}{|g_i| + |g_j| - |MCS(g_i, g_j)|}, \quad (4.2)$$

where a value of 0 (1) denotes that the graphs are maximally (minimally) dissimilar. The similarity matrix is then defined as

$$M_s(D)_{i,j} = \begin{cases} s(g_i, g_j) & \text{if } i \neq j, 1 \leq i, j \leq |D| \\ 1 & \text{if } i = j. \end{cases} \quad (4.3)$$

#### 4.3.4 MCS-based Locally Weighted Regression

MCS-based locally weighted linear regression (LWL-MCS) is a locally weighted version of linear regression which uses an MCS-based distance measure to determine the neighborhood of a test instance. It uses a local linear (ridge) regression model to fit a subset of the training instances that is in the neighborhood of the test instance whose class value is to be predicted. The linear regression model is one of the most important and widely-used models in statistics. Linear regression is a simple, yet successful regression algorithm which has been widely used in statistical applications for decades. In previous research locally weighted learning was successfully combined with linear regression [10]. Motivated by this idea, linear regression is used in locally weighted learning with an MCS-based distance measure. Training instances in the neighborhood of a test instance are weighted by applying a linear kernel to the MCS-based distance measure. This means that less weight



$$s(g_1, g_2) = \frac{|MCS(g_1, g_2)|}{|g_1| + |g_2| - |MCS(g_1, g_2)|} = \frac{9}{11 + 10 - 9} = 0.75$$

**Figure 4.1:** Sample similarity matrix calculation. The MCS between the graph structures  $g_1$  and  $g_2$  is marked blue.



is assigned to instances that are further from the test instance. The subset of the training data used to training each locally weighted linear regression model are determined by a  $k$ -nearest-neighbor algorithm. The actual predictive models are then built using a feature-vector representation of the training data. A regression prediction is obtained from the linear regression model taking the attribute values of the test instance as input.

Assume  $D_{Trg} = (x_i, y_i)$ ,  $1 \leq i \leq n$ , represents the training data set, where  $x_i$  are the data points and  $y_i$  their labels, respectively. Let  $x_q$  be a given test instance. The  $k$  nearest neighbors  $(x_1, \dots, x_k)$  of the test instance, sorted in increasing order of the distance, are defined in terms of an MCS-based distance. More precisely, the MCS-based distance  $d(x_q, x_i)$  of  $x_q$  to the  $i$ th nearest neighbor  $x_i$  is defined as:

$$\begin{aligned} d(x_q, x_i) &= 1 - M_s(D)_{q,i} \\ &= 1 - \frac{|MCS(x_q, x_i)|}{|x_q| + |x_i| - |MCS(x_q, x_i)|} \end{aligned} \quad (4.4)$$

where  $M_s(D)_{q,i}$  is defined in Equation 4.2.

Figure 4.2 shows an example of determining the  $k$  nearest neighbors of a given test instance  $x_q$  from a set of graph instances.

Let  $K$  be a weighting kernel function with  $K(y) = 0$  for all  $y \geq 1$ . Then, the weight  $w_i$  of each instance  $x_i$  is set to

$$w_i = K\left(\frac{d(x_q, x_i)}{d(x_q, x_k)}\right) \quad (4.5)$$

This means that instance  $x_k$  receives weight zero, all instances that are further away from the test instance also receive weight zero, and an instance identical to the test instance receives weight one. This work uses a linear weighting kernel  $K_{linear}$  defined as

$$K_{linear}(d) = 1 - d, \text{ for } d \in [0,1] \quad (4.6)$$

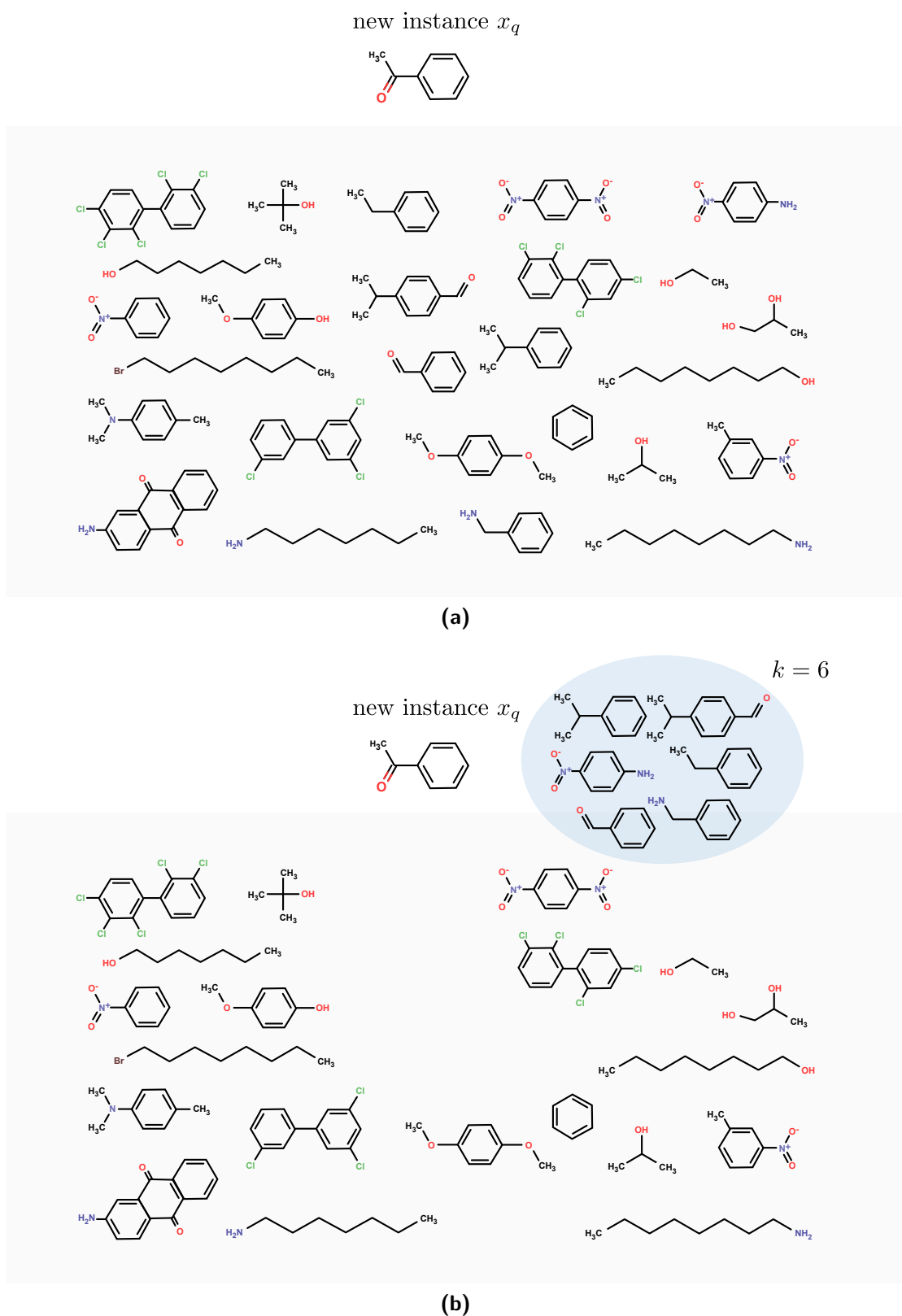
In other words, the weight is decreased linearly with the distance. The weights are then scaled so that the total weight of the instances used to generate the linear ridge regression model is approximately  $k$ . Thus, the rescaled weights  $w'_i$  are computed as follows:

$$w'_i = \frac{w_i \times k}{\sum_{l=0}^n w_l} \quad (4.7)$$

where  $n$  is the total number of training instances. Given a query instance  $x_q$ , to obtain the regression applicable to this query instance, the following cost function is minimized:

$$E(x_q) = \sum_{i=1}^k w_i (y_i - \hat{y}_i)^2 + \lambda \|\beta\|^2 \quad (4.8)$$

where  $\lambda$  is a regularization parameter.



**Figure 4.2:** Example of determining the  $k$  nearest neighbors (here:  $k = 6$ ) of a given test instance  $x_q$  from a training set of graph instances.

The training data can be represented by:

$$X\beta = y \quad (4.9)$$

where  $X$  is a matrix whose  $i$ th row represents the  $i$ th training instance  $x_i^T$  and  $y$  is a vector whose  $i$ th element represents the label of the  $i$ th training instance  $y_i$ . Thus, the dimensionality of  $X$  is  $n \times d$  where  $n$  is the number of data points and  $d$  is the number of features.

For a regression scheme, the weighted design matrix  $X'$  formed from the  $k$  nearest neighbors and weighted label vector  $y'$  can be formed using the unweighted quantities  $X$  and  $y$  as

$$X' = W'X \text{ and } y' = W'y. \quad (4.10)$$

$W'$  is a diagonal matrix with diagonal elements  $w'_i$  and zeros elsewhere. In the case of linear ridge regression,

$$X' = \begin{pmatrix} w'_1 & w'_1 x_1 \\ w'_2 & w'_2 x_2 \\ \vdots & \vdots \\ w'_k & w'_k x_k \end{pmatrix} \text{ and } y' = \begin{pmatrix} w'_1 y_1 \\ w'_2 y_2 \\ \vdots \\ w'_k y_k \end{pmatrix}. \quad (4.11)$$

The LWLR prediction for the point  $x_q$  is obtained by

$$\hat{y}_q = \hat{\beta}^T x_q, \text{ where } \hat{\beta} = (X'^T X' + \lambda I)^{-1} X'^T y'. \quad (4.12)$$

The pseudocode of LWL-MCS is given in Algorithm 10.

---

**Algorithm 10** LWL-MCS

---

- 1: Given: test instance  $x_q$ ,  $n$  training points  $(x_i, y_i)$
  - 2: Compute Prediction:
  - 3: a) Determine the  $k$  nearest neighbors of  $x_q$  in terms of the MCS-based distance
  - 4:  $d(x_q, x_i) = 1 - \frac{|MCS(x_q, x_i)|}{|x_q| + |x_i| - |MCS(x_q, x_i)|}$
  - 5: b) Compute the diagonal weight matrix  $W'$
  - 6: where  $w'_i = \frac{w_i \times k}{\sum_{l=0}^n w_l}$  with  $w_i = K \left( \frac{d(x_q, x_i)}{d(x_q, x_k)} \right)$
  - 7: c) Build the matrix  $X' = W'X$  and  $y' = W'y$
  - 8: d) Compute the locally linear model  $\hat{\beta} = (X'^T X' + \lambda I)^{-1} X'^T y'$
  - 9: e) The prediction for  $x_q$  is  $\hat{y}_q = \hat{\beta}^T x_q$
-

## 4.4 Experimental Results

This section empirically compares the performance of LWL-MCS, introduced in Section 4.3.4, against six methods.

1. **Linear SVM with the MCS-based empirical kernel map (EKM-MCS)**: The method learns a linear SVM on the empirical kernel map based on the MCS.
2. **Linear SVM with the empirical kernel map based on the MCS and a chemical feature set (EKM-COMB)**: The method learns a linear SVM on the empirical kernel map based on the MCS and a standard chemical feature set.
3. **A locally weighted linear regression** method which uses a feature-vector representation of the graphs to determine the nearest neighbors of a query graph and applies a linear weighted kernel shape based on an Euclidean distance to give the most weight to the nearest neighbors. In this work, the method will be called LWL-EUC. Note the difference between LWL-EUC and LWL-MCS: LWL-MCS combines different kinds of information. It uses an MCS-based distance measure for neighborhood determination and neighborhood weighting, whereas the actual prediction models are built using a feature-vector representation of the training instances. In contrast, LWL-EUC only uses the feature-vectors for both neighborhood determination and weighting and model building.
4. **Local Model Learning (LoMoGraph) with cluster size weighting (CS) [38]**: A method that combines clustering and classification or regression for making predictions on graph structured data. The approach consists of two steps: First, a structural clustering procedure is applied to find groups in a structural space that consist of similar graphs. More specifically, the graph structures in the clusters share a common structural scaffold that makes up a specific proportion,  $\theta$ , of the size of each graph in the cluster. Second, one local model is learned per structural cluster. In the prediction step, the query graph is assigned to one or more clusters using the structural clustering procedure. Based on this assignment, the prediction is made. Since the structural clustering procedure is overlapping and non-exhaustive, a graph can fall into no cluster, one cluster or multiple clusters. If it falls into no cluster, a global model is applied for prediction. If the query graph falls into a single cluster, the local model based on this cluster is used for prediction, and if it is assigned to multiple clusters, weighted local models are used dependent on cluster membership. The weight for a cluster is linearly dependent on its size. Thus, larger weights are assigned to larger clusters assuming that the more graphs a cluster has, the more reliable the corresponding model is.
5. **LoMoGraph with equal size weighting (EQ) [38]**: The method differs from CS (method 4) only in the weighting scheme. The predicted value of a query graph is the sum of the weighted predicted values.

6. **Global Model Learning (GL):** The method uses all the training data to learn a linear regression model.

To avoid overfitting the data, user parameters are optimized by internal cross-validation. LWL-MCS, LWL-EUC, EKM-MCS and EKM-COMB implement an internal grid search using only the training data to determine the best parameter values via inner 10-fold cross-validation. For LWL, the number of neighbors,  $k$ , was varied from 25 to 300 using a step size of 25 and the ridge regression parameter  $R$  was varied in the range  $\{1, 10, 25, 50, 100, 150, 200\}$ . For EKM-MCS and EKM-COMB, the complexity constant  $C$  is optimized in  $\{10^{-3}, 10^{-2}, 10^{-1}, 10^0, 10^1, 10^2\}$ . The parameter combination resulting in the lowest mean absolute error was then used for building the final model. It is observed that in the majority of cases the optimum parameter combinations lie within the grid, not at the border of the grid. In other words, this means that good or even optimal parameter combinations were found. For the remaining parameters, WEKA's [97] default parameter setting were used. As for the local model methods, the parameters that were used for clustering were defined based on a set of criteria: The similarity coefficient  $\theta$  of the clustering procedure was chosen so that the local models consist of minimally 5% and maximally 20% of the training data. The rationale behind this choice is that a too small value of  $\theta$  results in large, heterogeneous clusters whereas a too big value of  $\theta$  produces very few, small clusters or no clusters at all. In both cases the predictivity of LoMoGraph would be negatively affected. Another parameter called minimum cluster size controls how many graphs a cluster must have at least so that a local model can be learned. This parameter was chosen larger or equal to 20 as a lower bound for the number of graphs that are needed to train meaningful models. Moreover, the ridge regression parameter  $R$  was varied in the range  $\{1, 10, 25, 50, 100, 150, 200\}$ . Performance estimates are obtained using 100 times hold-out validation with a training set fraction of (i) 66% and (ii) 90% of the data. This means that 2/3 (9/10) of the data are used for training a model while the remaining 1/3 (1/10) is reserved for testing. To quantify predictive accuracy, the relative mean error, a standard measure in regression settings, is chosen. The Wilcoxon signed-rank test is applied to test for significant differences at a significance level of 5%.

The chemical domain is employed as application area by using real data sets of molecular graphs. The data sets are available at [www.cheminformatics.org](http://www.cheminformatics.org) and the same data sets as used by Buchwald *et al.* [38] (see Table 4.2). All results are based on the same base learning algorithm, i.e., linear (ridge) regression [104] and on the same feature vectors. For each data set, standard chemical descriptors (e.g., molecular weight, LogP, topological diameter, hydrogen bond acceptor/donor, polar surface area, number of atoms/bonds) were used that were computed using the cheminformatics library JOELib2 (<http://www-ra.informatik.uni-tuebingen.de/software/joelib>).

Tables 4.3 and 4.4 show the detailed experimental results in terms of relative mean absolute error, standard error and runtime performance for each algorithm on each data set. The first line for each data set show the performance of LWL-MCS as baselines to

**Table 4.2:** Overview of the data sets used for assessing the LWL-MCS method.  $n$  denotes the number of molecules in the respective data set.

Data set	$n$	Reference
4QSAR COX2	282	4QSAR database [217]
4QSAR DHFR	362	4QSAR database [217]
CPD MOUSE	442	ACD DSSTox databases [93]
CPD RAT	580	ACD DSSTox databases [93]
ISS MOUSE	316	Benigni/Vari Carcinogenicity [21]
ISS RAT	375	Benigni/Vari Carcinogenicity [21]
Suth COX2	414	Sutherland data set [216]
Suth DHFR	672	Sutherland data set [216]
Suth ER TOX	410	Sutherland data set [216]
FDAMDD	1216	ACD DSSTox databases [162]

compare against. Significance of differences between LWL-MCS and LWL-EUC, EKM-MCS, EKM-COMB, CS, EQ and GL respectively, is judged by a Wilcoxon signed-rank test at a confidence level of 95%. A summary of the results can also be seen in the bar charts shown in Figures 4.3 and 4.4. The experimental results show that LWL-MCS significantly outperforms LWL-EUC in 8 out of 10 cases for  $f_D = 66\%$  and in 7 out of 10 cases for  $f_D = 90\%$ . There is only a small number of cases where LWL-EUC performs best. In particular, for the Sutherland ER TOX and ISS MOUSE data set (for  $f_D = 66\%$  and  $f_D = 90\%$ ) as well as for the Suth COX2 data set (for  $f_D = 90\%$ ), LWL-EUC seems to be the better choice. Note, however, that only one of these three wins is statistically significant. For all remaining data sets LWL-MCS is significantly better than LWL-EUC. These results show that taking into account structurally related molecules to query molecules in combination with chemical descriptors improves in most of the cases the predictive performance in a statistically significant way. Using chemical descriptors alone, ignoring explicit structure information, will retrieve a set of structurally more diverse molecules, which lead to inferior predictions and make interpretation harder for the expert chemist. Using molecular graphs by themselves, ignoring chemical descriptors, will retrieve similar structures, but may overlook important binding site effects [160]. In contrast to LWL-EUC, LWL-MCS uses a combination of both chemical and structure descriptors. Whereas the MCS is better suited to measure structural similarity between molecules, chemical descriptors are better suited to discriminate between structurally similar molecules with regard to their actual biological activities.

The comparison with EKM-MCS shows that LWL-MCS significantly outperforms the MCS empirical kernel map in 8 out of the 10 data sets for both  $f_D = 66\%$  and  $f_D = 90\%$ . Extending the Empirical Kernel Map approach by additionally taking into account chemical descriptors the performance advantage of LWL-MCS diminishes. In 7 out of 10 cases, LWL-MCS shows a statistically significant improvement compared to EKM-COMB for  $f_D = 66\%$ . For  $f_D = 90\%$ , 5 wins, 4 draws and 1 loss are observed using LWL-MCS. The results suggest that EKM-COMB yields better prediction performance compared to

**Table 4.3:** Mean absolute errors (MAE) with standard errors and the results of the Wilcoxon signed-rank test with a 95% confidence level between LWL-MCS and LWL-EUC, EKM-MCS, EKM-COMB, CS, EQ and GL respectively. Abbreviations:  $f_D$  = fraction of data set used for training; Wil = Wilcoxon signed-rank test, W/L = wins/losses.

Data set	Method	MAE [ $f_D = 66\%$ ]	Wil (W/L)	MAE [ $f_D = 90\%$ ]	Wil (W/L)
4QSAR COX2	LWL-MCS	0.65252 ± 0.00652		0.63270 ± 0.01099	
	LWL-EUC	0.65842 ± 0.00596	++ (62/38)	0.64790 ± 0.01194	== (56/44)
	EKM-MCS	0.68752 ± 0.00552	++ (74/26)	0.68284 ± 0.01171	++ (72/28)
	EKM-COMB	0.66618 ± 0.00610	++ (63/37)	0.63679 ± 0.01100	== (52/48)
	CS	0.66796 ± 0.00598	++ (71/29)	0.65675 ± 0.01821	== (59/41)
	EQ	0.67089 ± 0.00606	++ (72/28)	0.68934 ± 0.04501	++ (60/40)
	GL	0.69208 ± 0.00634	++ (86/14)	0.67668 ± 0.01099	++ (72/28)
4QSAR DHFR	LWL-MCS	0.61831 ± 0.00444		0.60833 ± 0.00884	
	LWL-EUC	0.64055 ± 0.00499	++ (77/23)	0.63281 ± 0.00945	++ (68/32)
	EKM-MCS	0.61485 ± 0.00419	== (49/51)	0.61603 ± 0.00841	== (56/44)
	EKM-COMB	0.61127 ± 0.00455	-- (42/58)	0.60098 ± 0.00818	== (41/59)
	CS	0.65722 ± 0.00600	++ (83/17)	0.65909 ± 0.01058	++ (81/19)
	EQ	0.65863 ± 0.00665	++ (83/17)	0.66154 ± 0.01067	++ (80/20)
	GL	0.65066 ± 0.00476	++ (84/16)	0.64704 ± 0.00911	++ (74/26)
CPD MOUSE	LWL-MCS	0.75381 ± 0.00478		0.71661 ± 0.00986	
	LWL-EUC	0.77212 ± 0.00485	++ (74/26)	0.74309 ± 0.00954	++ (75/25)
	EKM-MCS	0.80529 ± 0.00516	++ (94/6)	0.78438 ± 0.00922	++ (86/14)
	EKM-COMB	0.79113 ± 0.00494	++ (88/12)	0.76894 ± 0.00957	++ (78/22)
	CS	0.82652 ± 0.00445	++ (100/1)	0.80785 ± 0.01026	++ (93/7)
	EQ	0.82643 ± 0.00445	++ (100/1)	0.80751 ± 0.01025	++ (93/7)
	GL	0.83234 ± 0.00439	++ (99/1)	0.81250 ± 0.01015	++ (9/6)
CPD RAT	LWL-MCS	0.87851 ± 0.00453		0.82912 ± 0.00863	
	LWL-EUC	0.91396 ± 0.00432	++ (95/5)	0.88585 ± 0.00963	++ (85/15)
	EKM-MCS	0.93815 ± 0.00503	++ (95/5)	0.91812 ± 0.01067	++ (86/14)
	EKM-COMB	0.91594 ± 0.00473	++ (84/16)	0.89104 ± 0.00981	++ (83/17)
	CS	0.96375 ± 0.00433	++ (100/0)	0.93886 ± 0.01052	++ (97/3)
	EQ	0.96377 ± 0.00434	++ (100/0)	0.93859 ± 0.01051	++ (96/4)
	GL	0.98125 ± 0.00407	++ (100/0)	0.95755 ± 0.01029	++ (100/0)
ISS MOUSE	LWL-MCS	0.75470 ± 0.00575		0.73286 ± 0.01003	
	LWL-EUC	0.74874 ± 0.00597	== (46/54)	0.72582 ± 0.01196	== (44/56)
	EKM-MCS	0.78007 ± 0.00594	++ (86/14)	0.77894 ± 0.01118	++ (80/20)
	EKM-COMB	0.78803 ± 0.00599	++ (86/14)	0.77536 ± 0.01045	++ (76/24)
	CS	0.82052 ± 0.00566	++ (97/3)	0.79935 ± 0.01102	++ (91/9)
	EQ	0.81973 ± 0.00567	++ (96/4)	0.79793 ± 0.01100	++ (91/9)
	GL	0.82662 ± 0.00571	++ (96/4)	0.81208 ± 0.01135	++ (91/9)
ISS RAT	LWL-MCS	0.89128 ± 0.00592		0.85751 ± 0.01191	
	LWL-EUC	0.91542 ± 0.00590	++ (79/21)	0.88418 ± 0.01158	++ (67/33)
	EKM-MCS	0.92283 ± 0.00571	++ (78/22)	0.87540 ± 0.01051	++ (62/38)
	EKM-COMB	0.90249 ± 0.00580	++ (57/43)	0.86287 ± 0.01053	== (56/44)
	CS	0.93018 ± 0.00584	++ (88/12)	0.90689 ± 0.01204	++ (78/22)
	EQ	0.92915 ± 0.00588	++ (85/15)	0.90460 ± 0.01205	++ (75/25)
	GL	0.95365 ± 0.00562	++ (95/5)	0.93690 ± 0.01145	++ (86/14)
Suth COX2	LWL-MCS	0.61090 ± 0.00482		0.60006 ± 0.01652	
	LWL-EUC	0.62060 ± 0.00429	++ (77/28)	0.59036 ± 0.00827	++ (71/29)
	EKM-MCS	0.59428 ± 0.00401	-- (37/63)	0.57554 ± 0.00909	== (56/44)
	EKM-COMB	0.58943 ± 0.00398	-- (30/70)	0.55685 ± 0.00849	-- (41/59)
	CS	0.62180 ± 0.00423	++ (66/34)	0.59526 ± 0.00831	++ (70/30)
	EQ	0.62111 ± 0.00427	++ (67/33)	0.59398 ± 0.00830	++ (73/27)
	GL	0.63966 ± 0.00563	++ (80/20)	0.61014 ± 0.00749	++ (72/28)
Suth DHFR	LWL-MCS	0.54921 ± 0.00262		0.52557 ± 0.00592	
	LWL-EUC	0.57875 ± 0.00276	++ (93/7)	0.55893 ± 0.00607	++ (87/13)
	EKM-MCS	0.60307 ± 0.00271	++ (100/0)	0.58643 ± 0.00577	++ (90/10)
	EKM-COMB	0.60925 ± 0.00304	++ (100/0)	0.59002 ± 0.00503	++ (93/7)
	CS	0.64540 ± 0.00326	++ (100/0)	0.60911 ± 0.00586	++ (94/6)
	EQ	0.64400 ± 0.00326	++ (100/0)	0.60722 ± 0.00576	++ (94/6)
	GL	0.68201 ± 0.00286	++ (100/0)	0.66562 ± 0.00527	++ (100/0)
Suth ER_TOX	LWL-MCS	0.83777 ± 0.00488		0.79448 ± 0.01088	
	LWL-EUC	0.82392 ± 0.00531	-- (31/69)	0.78053 ± 0.01107	== (45/55)
	EKM-MCS	0.90229 ± 0.00530	++ (95/5)	0.87545 ± 0.01074	++ (86/14)
	EKM-COMB	0.83593 ± 0.00528	== (51/49)	0.79222 ± 0.01054	== (51/49)
	CS	0.87209 ± 0.00577	++ (78/22)	0.83548 ± 0.01051	++ (73/27)
	EQ	0.87257 ± 0.00578	++ (77/23)	0.83802 ± 0.01057	++ (74/26)
	GL	0.90017 ± 0.00482	++ (97/3)	0.87262 ± 0.01110	++ (81/19)
FDAMDD	LWL-MCS	0.62019 ± 0.00213		0.59895 ± 0.00490	
	LWL-EUC	0.64402 ± 0.00224	++ (97/3)	0.62501 ± 0.00497	++ (87/13)
	EKM-MCS	0.66924 ± 0.00211	++ (100/0)	0.65202 ± 0.00513	++ (93/7)
	EKM-COMB	0.68599 ± 0.00232	++ (100/0)	0.66542 ± 0.00542	++ (97/3)
	CS	0.69865 ± 0.00229	++ (100/0)	0.67665 ± 0.00502	++ (98/2)
	EQ	0.69456 ± 0.00229	++ (100/0)	0.67110 ± 0.00501	++ (99/1)
	GL	0.75061 ± 0.00224	++ (100/0)	0.74424 ± 0.00531	++ (100/0)

++,==,-- statistically significant improvement, or degradation

**Table 4.4:** Mean runtime (training and testing time in sec) of one time hold-out validation with standard deviations for LWL-MCS and LWL-EUC, EKM-MCS, EKM-COMB, CS, EQ and GL. The abbreviations correspond to Table 4.3.

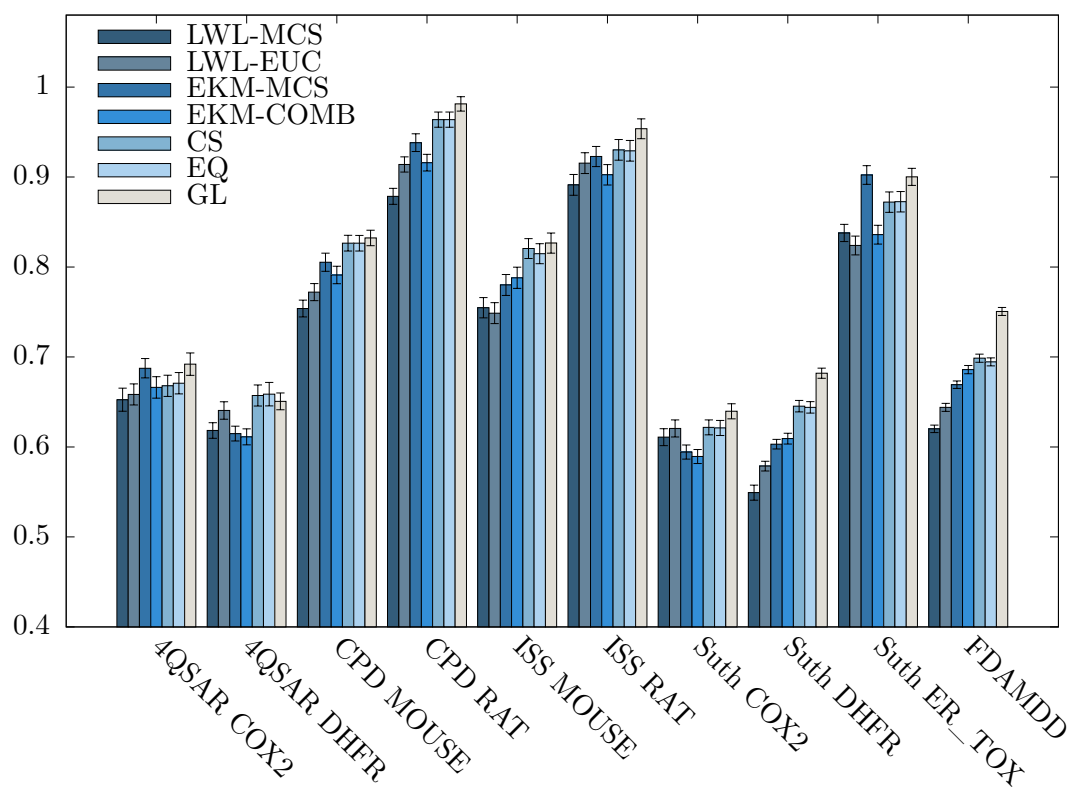
Data set	Method	t [ $f_D = 66\%$ ]	t [ $f_D = 90\%$ ]
4QSAR COX2	LWL-MCS	188.41 ± 3.18	228.49 ± 2.62
	LWL-EUC	76.38 ± 1.00	121.91 ± 1.44
	EKM-MCS	51.20 ± 5.64	97.73 ± 9.23
	EKM-COMB	156.31 ± 20.19	357.48 ± 29.84
	CS, EQ, GL	81.07 ± 7.94	95.17 ± 9.45
4QSAR DHFR	LWL-MCS	274.20 ± 7.64	322.82 ± 4.62
	LWL-EUC	114.71 ± 2.87	167.56 ± 3.96
	EKM-MCS	110.42 ± 11.86	246.04 ± 18.63
	EKM-COMB	304.41 ± 28.69	820.21 ± 69.52
	CS, EQ, GL	126.07 ± 9.84	184.21 ± 13.42
CPD MOUSE	LWL-MCS	450.31 ± 6.58	549.99 ± 2.90
	LWL-EUC	188.56 ± 3.05	265.62 ± 2.19
	EKM-MCS	234.87 ± 31.57	870.38 ± 0.11
	EKM-COMB	329.47 ± 74.14	1314.22 ± 115.53
	CS, EQ, GL	174.16 ± 7.76	270.37 ± 10.06
CPD RAT	LWL-MCS	742.57 ± 6.26	810.01 ± 7.52
	LWL-EUC	253.80 ± 4.01	360.78 ± 2.75
	EKM-MCS	554.07 ± 47.46	1869.56 ± 180.28
	EKM-COMB	663.36 ± 52.07	2631.27 ± 268.62
	CS, EQ, GL	282.15 ± 13.11	446.36 ± 16.92
ISS MOUSE	LWL-MCS	255.16 ± 1.86	314.17 ± 4.82
	LWL-EUC	120.24 ± 1.38	178.48 ± 1.92
	EKM-MCS	64.71 ± 6.31	282.76 ± 22.17
	EKM-COMB	70.33 ± 8.46	372.32 ± 30.99
	CS, EQ, GL	97.91 ± 5.11	145.69 ± 6.26
ISS RAT	LWL-MCS	338.51 ± 4.21	429.99 ± 2.51
	LWL-EUC	150.22 ± 1.74	224.27 ± 4.96
	EKM-MCS	122.08 ± 11.99	443.76 ± 37.32
	EKM-COMB	133.08 ± 12.92	600.08 ± 60.16
	CS, EQ, GL	133.43 ± 6.58	199.43 ± 9.06
Suth COX2	LWL-MCS	389.10 ± 6.53	432.79 ± 6.16
	LWL-EUC	135.01 ± 4.06	194.28 ± 3.73
	EKM-MCS	246.72 ± 25.23	576.46 ± 46.10
	EKM-COMB	540.76 ± 56.04	1356.16 ± 124.03
	CS, EQ, GL	135.78 ± 15.64	146.68 ± 13.04
Suth DHFR	LWL-MCS	792.44 ± 9.38	1001.70 ± 18.57
	LWL-EUC	249.47 ± 3.90	372.17 ± 4.70
	EKM-MCS	1232.64 ± 148.81	2972.67 ± 300.90
	EKM-COMB	2034.21 ± 158.88	5427.62 ± 354.86
	CS, EQ, GL	347.67 ± 20.08	483.39 ± 18.62
Suth ER_TOX	LWL-MCS	395.38 ± 3.81	458.13 ± 4.93
	LWL-EUC	156.38 ± 3.26	221.10 ± 1.89
	EKM-MCS	249.21 ± 27.32	602.03 ± 45.19
	EKM-COMB	409.58 ± 37.27	1132.35 ± 84.65
	CS, EQ, GL	195.04 ± 23.98	273.89 ± 35.78
FDAMDD	LWL-MCS	3159.39 ± 120.76	3263.67 ± 29.01
	LWL-EUC	638.87 ± 6.53	964.28 ± 5.58
	EKM-MCS	4133.40 ± 277.01	11345.37 ± 584.25
	EKM-COMB	1662.30 ± 95.56	26449.16 ± 1247.72
	CS, EQ, GL	2267.15 ± 179.19	3157.60 ± 237.64

EKM-MCS. This means that taking into account non-structural descriptors in addition to structural descriptors improves in many cases the predictive performance in a statistically significant way.

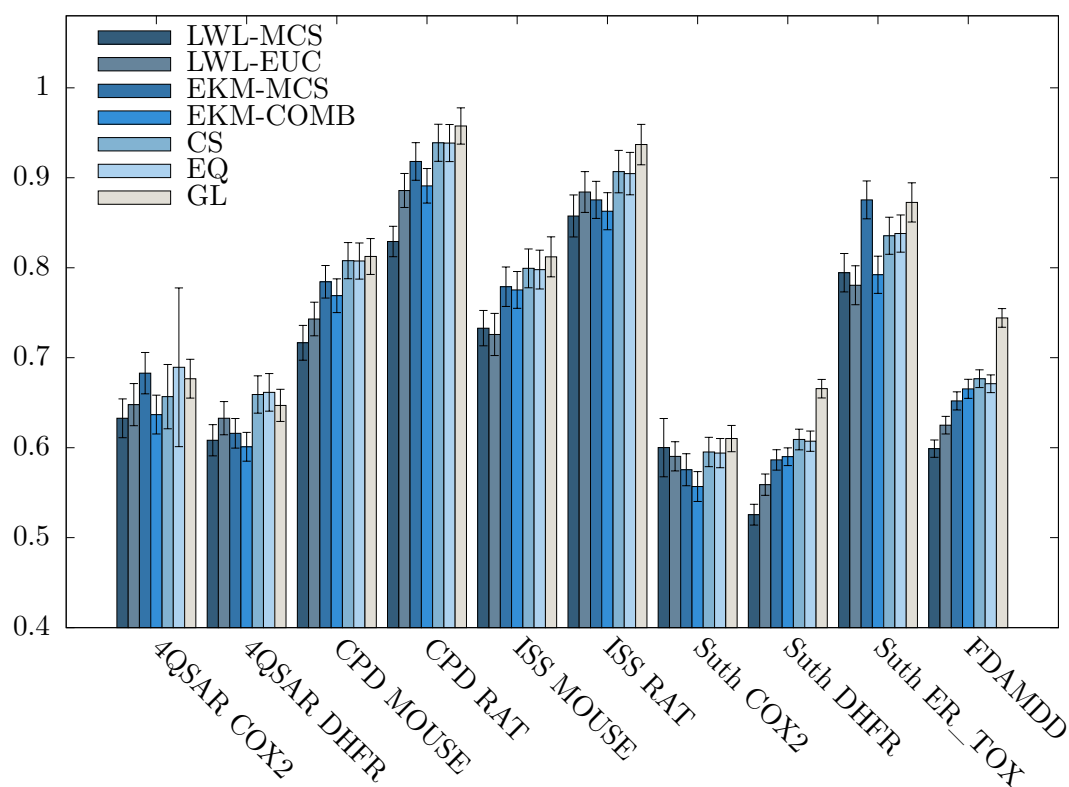


For the remaining three comparison methods (CS, EQ and GL), it is observed that LWL-MCS outperforms all of them in 10 out of 10 cases for  $f_D = 66\%$ . Using 90% of the data for training, LWL-MCS significantly outperforms CS in 9 of 10 cases and EQ and GL in all cases. However, the runtime results in Table 4.4 suggest a clear advantage of the local model learning approaches over LWL-MCS. The results demonstrate the trade-off between the runtimes of the algorithms and the quality of the results achieved. Whereas methods which perform lazy learning, like LWL-MCS, achieve better performance regarding prediction accuracy, methods performing eager learning, like local model learning, show better performance regarding runtime, but at the same are limited in their expressive power. While there is no one rule for what is acceptable, there are some applications, e.g., in the domain of graphs, where users are willing to accept higher runtime for significant accuracy improvements.

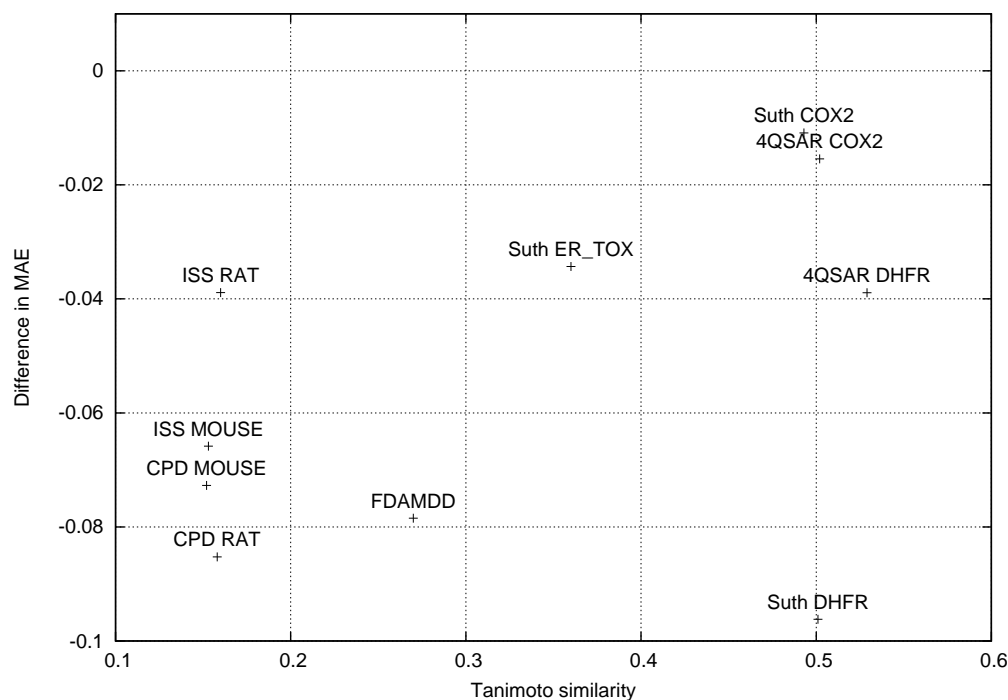
In the following, the performance difference between LWL-MCS and the local model methods is analyzed on (i) the COX2 data sets, (ii) the CPD and ISS data sets, (iii) FDAMDD and (iv) Suth DHFR. For the COX2 data sets, a relatively small difference in performance compared to the remaining data sets (see results for CS in Figure 4.5) is observed. The data sets contain extremely similar molecules, often differing in only one atom. Hence, the structural clustering procedure applied by the local model methods results in a small number of relatively large, structurally homogeneous clusters so that the majority of molecules are predicted with local models. This leads to only slight differences in performance between LWL-MCS and the local model approaches. For the CPD and ISS data sets, a comparison between the mean absolute errors shows a clear performance advantage for LWL-MCS. Primarily, this positive effect is explained as a result of the structurally heterogeneity of the data sets consisting of many small molecules ( $\sim$  up to 10 atoms), not particularly suitable to find many large homogeneous structural clusters. For these data sets CS and EQ produce rather few, relatively small clusters, so that the majority of molecules are predicted with global models. On the FDAMDD data set, LWL-MCS achieves a strong gain over local models. This data set is the largest one, comprising structurally more heterogeneous molecules. Consequently, local model methods use a relatively low similarity threshold resulting in large, heterogeneous clusters. Hence, local models are often applicable, but still inferior, due to the relative structural inhomogeneity of each cluster. LWL-MCS achieves its largest gain on the Suth DHFR data set. This data consists of structurally more heterogeneous molecules resulting in few and relatively small structural clusters. Hence, inferior global models need to be applied more often, causing the local model approach to underperform in comparison to LWL-MCS (similarly to CPD and ISS). Figure 4.5 shows a strong relationship between the average Tanimoto similarity and the advantage of LWL-MCS: gains increase for lower similarities. The only exceptions are the ISS RAT and Suth DHFR data sets, where the latter can be viewed as an outlier.



**Figure 4.3:** Mean absolute errors and 95% confidence intervals for the comparison methods using 66% of the data for training.



**Figure 4.4:** Mean absolute errors and 95% confidence intervals for the comparison methods using 90% of the data for training.



**Figure 4.5:** Relation between the Tanimoto similarity and the difference in MAE between LWL-MCS and CS ( $f_D = 66\%$ ).

## 4.5 Discussion and Conclusion

The chapter investigated locally weighted linear regression based on maximum common subgraph based distances (LWL-MCS). Clearly, LWL-MCS is not a completely new method, but a novel combination that, quite surprisingly, has not been considered so far. The experimental evaluation showed that the method outperforms other methods building on the neighborhood of graphs for regression. Determining the neighborhood of graph instances based on the MCS and making the actual prediction based on feature vectors apparently works particularly well for the application area of quantitative structure-activity relationships. The MCS retrieves structures with a large common structural scaffold, for which differences in the activities can be explained by differences in the physicochemical properties. However, given the performance of this conceptually simple learning scheme compared to other methods using the local neighborhood of graphs, it appears worthwhile to study similar approaches on other types of structured data (like sequence, tree, logical or database representations). This could be similarly successful whenever such structural similarity (longest common subsequence, maximum common subtree, least general generalization) can be complemented by feature vectors with orthogonal information.



# CHAPTER 5

---

## The Structural Cluster Kernel

---

In recent years, graph kernels have received considerable interest within the machine learning and data mining community. This chapter introduces a novel approach enabling kernel methods to utilize additional information hidden in the structural neighborhood of the graphs under consideration. The novel SCK incorporates similarities induced by a structural clustering algorithm to improve state-of-the-art graph kernels. The approach taken is based on the idea that graph similarity can not only be described by the similarity between the graphs themselves, but also by the similarity they possess with respect to their structural neighborhood. The novel kernel is applied in a supervised and a semi-supervised setting to regression and classification problems on a number of real-world data sets of molecular graphs. The experimental results show that the structural cluster similarity information can indeed leverage the prediction performance of the base kernel, particularly when the data set is structurally sparse and consequently structurally diverse. By additionally taking into account a large number of unlabeled instances the performance of the structural cluster kernel can further be improved.

### 5.1 Introduction

The topic of graph similarity and in particular kernel approaches have attracted considerable interest in recent years [87, 166, 191, 230]. To determine the similarity of two graphs, most approaches decompose the graphs in different ways: either into a potentially very large set of smaller subgraphs or related graph features, or into one or more larger common subgraphs (connected or disconnected). This chapter investigates the question whether the structural neighborhood of two graphs can also contribute to similarity searches and consequently to improve prediction performance. To determine the structural neighborhood of a graph, the previously proposed structural graph clustering method PSCG is used.

In the work presented here, a novel kernel called SCK is proposed, which in addition to existing kernel approaches measures the similarity between two graphs, by their assignment to structural clusters found with PSCG. The SCK first employs the structural clustering

algorithm to determine small, structurally homogeneous regions in the input space, and then uses the pairwise similarities between these regions to define a similarity measure for graphs. The approach taken here is to extend two state-of-the-art graph kernels using this structural distance measure: the WDK [166] and the NSPDK [61].

To study the effectiveness of the SCK, the prediction performance is measured in both the regression and classification setting, by employing several real-world data sets of molecular graphs within the experiments. To show the advantage of combining graph similarity and structural cluster similarity, the SCK approach is compared with the base kernels using graph similarity alone. Furthermore, the SCK is compared against two approaches, (i) an approach that also employs structural clustering during model construction and learns one local model per structural cluster and (ii) an approach called LWL-MCS, a variant of locally weighted regression on graphs (structures) that uses the maximum common subgraph for determining and weighting the neighborhood of a graph and feature vectors for the actual regression model. The performance of the SCK approach is further investigated in the semi-supervised setting, where the base kernel is deformed by a cluster kernel encoding similarities between both labeled and unlabeled examples.

This chapter is organized as follows: In Section 5.2, the proposed structural cluster kernel is introduced. Section 5.3 presents and discusses the experimental results, before Section 5.4 gives a conclusion.

## 5.2 Method

### 5.2.1 Structural Cluster Kernel

This section introduces a novel kernel, called structural cluster kernel, that leverages information of a clustering algorithm to improve a base kernel representation. The main idea is to change the similarity metric of a base kernel so that the relative similarity between two points is higher if the points are in the same cluster. The kernel uses a combination of two similarity measures: (1) a base kernel that computes structural similarity between pairs of

---

#### Algorithm 11 Structural Cluster Kernel

---

- 1: Given: training points  $D_{Trg} = \{(x_1, y_1), \dots, (x_t, y_t)\}$  and test points  $D_{Tst} = \{x_{t+1}, \dots, x_n\}$ ,  $x_i \in \mathbb{R}^n$ ,  $i = 1, \dots, n$
  - 2: a) Cluster training points using PSCG
  - 3: c) Build cluster matrix on the training set
  - 4:  $K_{Cl}(x_i, x_j) = \frac{1}{|n_{x_i}| |n_{x_j}|} \sum_{C_k: x_i \in C_k} \sum_{C_l: x_j \in C_l} K(C_k, C_l)$ ,  $i, j \in \{1, \dots, t\}$
  - 5: d) Build the SCK on the training data set by taking the product between the base kernel  $K_b$  and the cluster kernel  $K_{Cl}$
  - 6:  $K_{SC}(x_i, x_j) = K_{Cl}(x_i, x_j) \times K_b(x_i, x_j)$ ,  $i, j \in \{1, \dots, t\}$
  - 7: e) Compute cluster assignments for all test points
  - 8: f) Compute  $K_{SC}(x_j, x_i)$  between each test point  $x_j$  and all training points  $x_i$ ,  $i = 1, \dots, t$ .
-

graphs and (2) a cluster-based similarity measure that describes how close examples are to each other in terms of the similarities between the clusters they belong to. The similarity between two clusters is computed by taking the average of the similarities between the cluster instances. In application to molecule regression and classification, the WDK and NSPDK (see Section 2.3.2.2 and 2.3.2.3) are used as base kernel. For the cluster-based kernel, the structural clustering approach introduced in section 3.2 that clusters a data set of graphs based on structural similarity. The cluster similarity information is used to improve pointwise similarities, based on which the final kernel is constructed.

In the following, the steps which are necessary to build the structural cluster kernel are described. Let  $D_{Trg} = \{(x_1, y_1), \dots, (x_t, y_t)\}$  denote the training data, where  $x_i \in X$  represent the data points and  $y_i$  their labels, respectively. Further, let  $D_{Tst} = \{x_{t+1}, \dots, x_n\}$  denote the set of test points. First, the training set is clustered by the structural clustering procedure PSCG presented in Section 3.2. The resulting clusters are used to build a kernel representing the pairwise similarities between all clusters. In this kernel representation, each of the pairwise sets of the structural clusters is seen as a single data point, and a higher level kernel is designed so as to compare the two clusters. The similarity between two clusters is computed by taking the average of the sum of the pairwise similarities between all graph instances in both clusters. The kernel  $K(C_i, C_j)$  is defined as

$$K(C_i, C_j) = \begin{cases} \frac{1}{|C_i||C_j|} \sum_{x_k \in C_i} \sum_{x_l \in C_j} K_b(x_k, x_l) & \text{if } i \neq j \\ 1 & \text{if } i = j, \end{cases} \quad (5.1)$$

where  $K_b(x_k, x_l)$  represents the base kernel and  $C_i, C_j \in \{C_1, \dots, C_p\}$ . As mentioned earlier, the WDK and NSPDK are used as base kernel to compute the pairwise similarities between graphs. In the next step, a kernel representation  $K_{Cl}(x_i, x_j)$  is built based on the averaged pairwise similarities between the clusters  $x_i$  and  $x_j$  belong to.  $K_{Cl}(x_i, x_j)$  is defined as

$$K_{Cl}(x_i, x_j) = \begin{cases} \frac{1}{|n_{x_i}||n_{x_j}|} \sum_{C_k: x_i \in C_k} \sum_{C_l: x_j \in C_l} K(C_k, C_l) & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \quad (5.2)$$

where  $n_{x_i}$  denotes the number of clusters containing  $x_i$ ,  $n_{x_j}$  denotes the number of clusters containing  $x_j$  and  $C_k, C_l \in \{C_1, \dots, C_p\}$ . Thus, the points are mapped to a feature space where the pointwise similarities are equal to the cluster similarities in the input space. The points belonging to the same cluster will result in matrix entries close to one, whereas for the points from different clusters, the entries will be close to zero. Figure 5.1 illustrates the cluster kernel concept.

The cluster similarity weights  $K_{Cl}(x_i, x_j)$  are combined with the values of the base kernel  $K_b(x_i, x_j)$ , thus forming the final kernel matrix. To sum up, the new structural cluster kernel is

$$K_{SC}(x_i, x_j) = K_b(x_i, x_j) \times K_{Cl}(x_i, x_j) \quad (5.3)$$

We are faced with two problems in the construction of the above structural cluster kernel: (i) the base kernel matrix has to be positive semi-definite and (ii) the structural cluster kernel must be positive semi-definite. The first requirement is obvious, since the WDK and NSPDK are used as base kernels, which are known to be valid kernels. In the following, a proof sketch is provided to show that the structural cluster kernel is a valid kernel.

**5.2.1.0.1 Proof Sketch:**  $K_{Cl}$  is a valid kernel, since each kernel value  $K_{Cl}(x_i, x_j)$  contains the average sum of pairwise similarities between all clusters, which in turn encompass the average sum of all training instances  $x_i \in \{x_1, \dots, x_t\}$ .

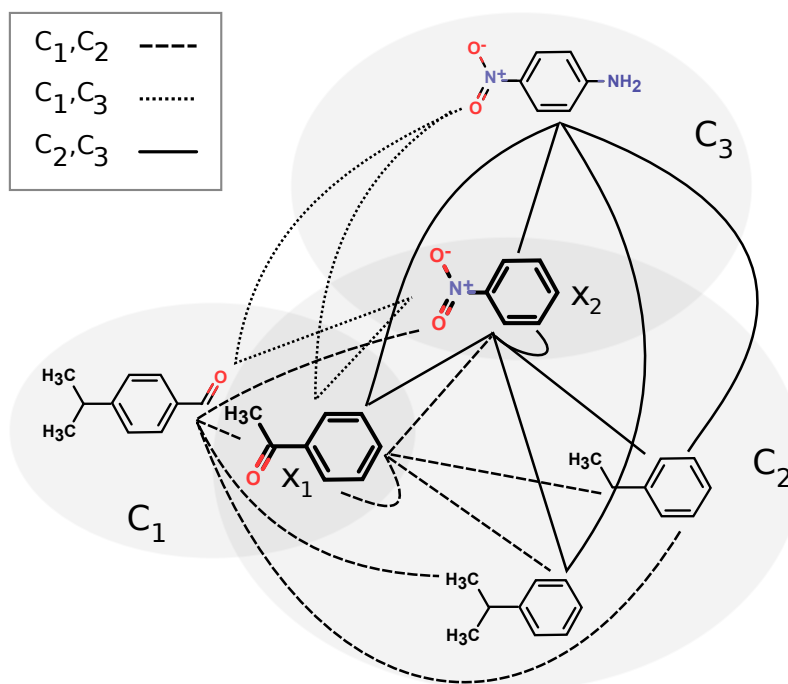
For each pair of clusters, one kernel is defined that returns the average similarity between the two clusters for the first instance in cluster one and the second in cluster two. For all other instances, it returns zero. As the sum of two valid kernels is again a valid kernel, the resulting function is a valid kernel as well. Applying the kernel to two instances, only the clusters to which the respective two instances are assigned are considered. Consequently, most of the summands are equal to zero:

$$\begin{aligned} K_{Cl}(x_i, x_j) &= \frac{1}{|n_{x_i}| |n_{x_j}|} \sum_{C_k: x_i \in C_k} \sum_{C_l: x_j \in C_l} K(C_k, C_l) \\ &= \frac{1}{|n_{x_i}| |n_{x_j}|} \sum_{C_k: x_i \in C_k} \sum_{C_l: x_j \in C_l} K(C_k, C_l) \\ &\quad + \underbrace{\frac{1}{|m_{x_i}| |m_{x_j}|} \sum_{C_k: x_i \notin C_k} \sum_{C_l: x_j \notin C_l} K(C_k, C_l)}_0 \\ &\quad + \underbrace{\frac{1}{|n_{x_i}| |m_{x_j}|} \sum_{C_k: x_i \in C_k} \sum_{C_l: x_j \notin C_l} K(C_k, C_l)}_0 \\ &\quad + \underbrace{\frac{1}{|m_{x_i}| |n_{x_j}|} \sum_{C_k: x_i \notin C_k} \sum_{C_l: x_j \in C_l} K(C_k, C_l)}_0, \end{aligned} \quad (5.4)$$

where  $n_{x_i}$  denotes the number of clusters containing  $x_i$  and  $m_{x_i}$  denotes the number of clusters not containing  $x_i$ .

In kernel methods, for predicting the label of a new test point kernel function calculations need to be performed only between the test points and the training points. For computing the kernel entries, each test point first needs to be assigned to one or more clusters using the structural clustering procedure to compute  $K_{Cl}(x_i, x_t)$ . Based on this cluster assignment the similarity  $K_{Cl}(x_i, x_t)$  between the test point  $x_t$  and all training points  $x_i$  is computed by



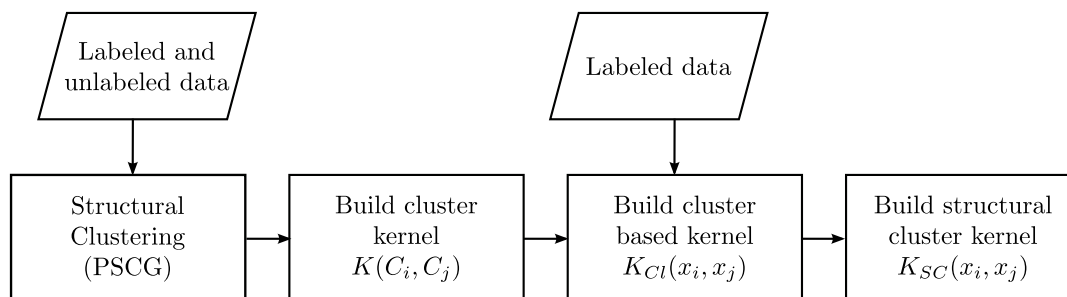


**Figure 5.1:** Illustration of the cluster kernel concept. The cluster-based similarity  $K_{Cl}(x_1, x_2)$  between the highlighted structures  $x_1$  and  $x_2$  is computed based on the averaged pairwise similarities between the clusters they belong to.  $x_1$  belongs to  $C_1$  and  $C_2$ ,  $x_2$  to  $C_2$  and  $C_3$ . Thus, the pairwise similarities between the cluster instances of cluster  $C_1C_2$ ,  $C_1C_3$ ,  $C_2C_2$  (which equals 1) and  $C_2C_3$  need to be computed.

averaging the pairwise similarities between all clusters  $x_t$  and  $x_i$  are assigned to (Equation 5.2). The kernel matrix  $K_{SC}$  is extended by taking the inner product between  $K_{Cl}(x_i, x_t)$  and  $K_b(x_i, x_t)$  between each test point  $x_t$  and all training points  $x_i$ ,  $i = 1, \dots, t$ . The steps needed for the calculation of the structural cluster kernel are shown in Algorithm 11.

### 5.2.2 Semi-Supervised Setting

In semi-supervised learning, one tries to improve a classifier trained on labeled data by exploiting a relatively large set of unlabeled data. If unlabeled data is added to the relatively small labeled data set, it is expected that the new similarity, obtained via structural clustering and the use of unlabeled data, induces a better representational space for classification and regression than using only the labeled data. Therefore, the kernel construction in Section 5.2.1 is extended by involving a large number of unlabeled data. The structural cluster kernel is constructed as follows: First, both the labeled and unlabeled training data are clustered with the structural clustering procedure to determine small, structurally homogeneous neighborhoods of the input space. The resulting clusters are then used to build a kernel representing the pairwise similarities between all clusters. As in the supervised setting, the cluster similarity information is used to improve pointwise similarities between the labeled data samples, based on which the final structural cluster kernel is constructed. Figure 5.2 illustrates the workflow.



**Figure 5.2:** Flowchart illustrating the construction of the structural cluster kernel in the semi-supervised setting.

### 5.3 Experimental Results

This section studies the performance of the proposed structural cluster kernel in a supervised setting. Next, the performance of the cluster kernel is investigated in a semi-supervised setting using large amounts of unlabeled data to augment the labeled data in order to test if the prediction performance can be improved. For all experiments, the chemical domain is employed as application area by using real data sets of molecular graphs. In Table 5.1 an overview of the data sets is provided.

**Table 5.1:** Overview of the data sets used for assessing the structural cluster kernel.  $n$  denotes the number of molecular graphs in the respective data set.

Data set	$n$	class.(SAR)/ regr.(QSAR)	Reference
4QSAR COX2	282	regression	4QSAR database [217]
4QSAR DHFR	362	regression	4QSAR database [217]
CPD MOUSE	442	regression	ACD DSSTox databases [93]
CPD RAT	580	regression	ACD DSSTox databases [93]
ISS MOUSE	316	regression	Benigni/Vari Carcinogenicity [21]
ISS RAT	375	regression	Benigni/Vari Carcinogenicity [21]
Suth COX2	414	regression	Sutherland data set [216]
Suth DHFR	672	regression	Sutherland data set [216]
Suth ER TOX	410	regression	Sutherland data set [216]
FDAMDD	1216	regression	ACD DSSTox databases [162]
Biodeg	328	regression	Biodegradability data set [73]
Tox09	1213	regression	Environmental Toxicity Prediction Challenge 2009 [45]
ER_LIT	381	regression	Sutherland data set [216]
CYP INH 2C9	700	classification	Yap and Chen [248]
CYP SUB 2C9	700	classification	Yap and Chen [248]
Fontaine	435	classification	Fontaine <i>et al.</i> [80]
NCI AIDS	1000	classification	DTP AIDS Antiviral Screen [54]
CPDB MUT	684	classification	Mutagenicity data set [102]

### 5.3.1 Supervised Setting

This section empirically compares the performance of the structural cluster kernel approach against five methods.

1. **WDK**: The Weighted Decomposition Kernel is used to build a classification or regression model. Section 2.3.2.2 provides a detailed description of the WDK.
2. **NSPDK**: The Neighborhood Subgraph Pairwise Distance Kernel (see Section 2.3.2.3) is used to build a classification or regression model.
3. **LoMoGraph**: LoMoGraph [38] combines clustering and classification or regression for making predictions on graph structured data. The approach consists of two steps: First, the structural clustering procedure PSCG is applied to find groups of graphs in a structural space that share a common structural scaffold with a minimum size. The sizes of these common subgraphs are used as a measure of similarity between the graphs. A graph is assigned to a cluster provided that there exists at least one common subgraph, whose size is equal or greater than a user-defined threshold  $\theta$ . Second, one local model is learned per structural cluster using a feature-vector representation of the graphs where the features encode standard chemical descriptors in the setting of molecular graphs. In the prediction step, the query graph is assigned to one or more clusters using PSCG. Based on this assignment, the prediction is made. Since the structural clustering procedure is overlapping and non-exhaustive, a graph can fall into no cluster, one cluster or multiple clusters. If it falls into no cluster, a global model is applied for prediction. If the query graph falls into a single cluster, the local model based on this cluster is used for prediction, and if it is assigned to multiple clusters, weighted local models are used dependent on cluster membership. The weight for a cluster is linearly dependent on its size. Thus, larger weights are assigned to larger clusters, assuming that the more graphs a cluster has, the more reliable the corresponding model is.
4. **LoMoGraph WDK**: The method combines LoMoGraph with WDK. More precisely, one local model is learned per structural cluster based on the WDK.
5. **LoMoGraph NSPDK**: The method combines LoMoGraph with NSPDK, i.e., one local model is learned per structural cluster based on the NSPDK.

The structural cluster kernel approach was investigated using both NSPDK and WDK as base kernel. For SCK with NSPDK and SCK with WDK, not only the approach with the diagonal elements in the kernel matrices  $K(C_i, C_j)$  (Equation 5.1) and  $K_{Cl}(x_i, x_j)$  (Equation 5.2) set to one was investigated, but also a second approach, where the diagonal elements are computed in the same way as the non-diagonal elements. These four approaches are referred to as SCK NSPDK (d=1), SCK NSPDK (d $\neq$ 1), SCK WDK (d=1) and SCK WDK (d $\neq$ 1).

In the experiments, regression and classification were performed using the SVM algorithm. Several user parameters were optimized by internal cross-validation. For SCK WDK, SCK NSPDK, WDK, NSPDK, LoMoGraph WDK, and LoMoGraph NSPDK, the trade-off between training error and margin,  $C$ , was selected from  $\{10^{-3}, 10^{-2}, 10^{-1}, 10^0, 10^1, 10^2\}$ . Further, the radius  $r$  for WDK was optimized in the range  $\{1, 2, 3, 4\}$ . The parameter combination resulting in the lowest mean absolute error (the highest accuracy) was then used for building the final model. All other SVM parameters were left at their default values. For NSPDK, the maximum radius  $r^*$  was set to 2, and the maximum distance  $d^*$  to 5. For the SCK approaches, the similarity coefficient  $\theta$  used by PSCG was set to 0.5. For FDAMDD and NCI AIDS an exception was made setting  $\theta$  to 0.3 to take into account the size and structural heterogeneity of the data sets. As for LoMoGraph, the parameters that were used for clustering were defined based on a set of criteria: the similarity coefficient of PSCG was chosen such that the local models consist of minimally 5% and maximally 20% of the training data. The rationale behind this choice is that a too small value of  $\theta$  results in large, heterogeneous clusters whereas a too big value of  $\theta$  produces very few, small clusters or no clusters at all. In both cases the predictivity of LoMoGraph would be negatively affected. For the experiments on SCK, the same values were used for  $\theta$  for all data sets. Another parameter called minimum cluster size controls how many graphs a cluster must have at least so that a local model can be learned. This parameter was chosen greater than or equal to 20 as a lower bound for the number of graphs that are needed to train meaningful models.

Performance estimates were obtained using 100 times hold-out validation with a training set fraction of 66%. This means that 2/3 of the data were used for training a model while the remaining 1/3 were reserved for testing. To quantify predictive accuracy, the relative mean error (regression) and classification accuracy (classification) were chosen, which are standard measures in regression and classification settings. The Wilcoxon signed-rank test and the corrected resampled t-test [172] were applied to test for significant differences at a significance level of 5%.

Tables 5.2, 5.3, 5.4 and 5.5 show the detailed experimental results in terms of relative mean absolute error (regression) and accuracy (classification) for the various methods on all data sets. The results for LoMoGraph were taken from the original publication [38]. Since not all data sets were used in this paper, the table contains missing values. In the same tables the second column shows the performance of the respective SCK method as baseline to compare against. For better illustration, the reference method is highlighted in italic. It is indicated whether the respective SCK method is significantly better or worse than the comparison methods at  $p < 0.05$  using both the Wilcoxon signed-ranked test and the corrected resampled t-test. In the following, the results are discussed based on the more conservative corrected resampled t-test. Overall, the experimental results show that the structural cluster kernel with NSPDK as base kernel performs always better than all comparison methods using WDK as base kernel. This demonstrates that NSPDK is a much more powerful base kernel compared to WDK. Moreover, it is observed that the

**Table 5.2:** Mean absolute errors with standard deviations of SCK NSPDK, NSPDK, LoMoGraph NSPDK and LoMoGraph on the regression data sets. Statistically significant results are reported using both the Wilcoxon signed-rank test and the corrected resampled t-test (separated by a '|').

Data set	<i>SCK NSPDK</i> ( $d \neq 1$ )	SCK NSPDK ( $d=1$ )	NSPDK	LoMoGraph NSPDK	LoMoGraph
4QSAR COX2	$0.607 \pm 0.055$	$0.625 \pm 0.055$ ●	$0.601 \pm 0.054$ ○	$0.628 \pm 0.055$ ●	$0.868 \pm 0.135$ ●●
4QSAR DHFR	$0.551 \pm 0.035$	$0.572 \pm 0.037$ ●●	$0.541 \pm 0.033$ ○	$0.562 \pm 0.035$ ●	$0.955 \pm 0.102$ ●●
CPD MOUSE	$0.751 \pm 0.041$	$0.760 \pm 0.041$ ●	$0.764 \pm 0.040$ ●	$0.775 \pm 0.039$ ●●	$1.276 \pm 0.154$ ●●
CPD RAT	$0.868 \pm 0.045$	$0.870 \pm 0.044$ ●	$0.887 \pm 0.043$ ●●	$0.877 \pm 0.042$	$1.529 \pm 0.116$ ●●
ISS MOUSE	$0.738 \pm 0.046$	$0.740 \pm 0.046$	$0.753 \pm 0.045$ ●●	$0.761 \pm 0.048$ ●●	$1.197 \pm 0.111$ ●●
ISS RAT	$0.860 \pm 0.050$	$0.863 \pm 0.047$	$0.900 \pm 0.046$ ●●	$0.873 \pm 0.051$ ●	$1.371 \pm 0.109$ ●●
Suth COX2	$0.559 \pm 0.039$	$0.586 \pm 0.041$ ●●	$0.552 \pm 0.038$ ○○	$0.573 \pm 0.040$ ●	$0.905 \pm 0.137$ ●●
Suth DHFR	$0.504 \pm 0.026$	$0.519 \pm 0.030$ ●●	$0.493 \pm 0.026$ ○○	$0.498 \pm 0.027$ ○	$0.941 \pm 0.066$ ●●
Suth ER TOX	$0.828 \pm 0.046$	$0.842 \pm 0.045$ ●	$0.820 \pm 0.042$	$0.847 \pm 0.052$ ●	$1.216 \pm 0.141$ ●●
FDAMDD	$0.629 \pm 0.029$	$0.647 \pm 0.023$ ●●	$0.612 \pm 0.024$ ○○	$0.621 \pm 0.026$ ○	$0.951 \pm 0.051$ ●●
Biodeg	$0.844 \pm 0.051$	$0.875 \pm 0.055$ ●●	$0.867 \pm 0.050$ ●●	$0.874 \pm 0.053$ ●●	-
Tox09	$0.345 \pm 0.015$	$0.386 \pm 0.016$ ●●	$0.342 \pm 0.015$	$0.367 \pm 0.016$ ●●	-
ER_LIT	$0.492 \pm 0.039$	$0.499 \pm 0.035$ ●	$0.495 \pm 0.039$	$0.495 \pm 0.041$	-

●,○ statistically significant improvement, or degradation of SCK NSPDK ( $d \neq 1$ ) over the other methods

**Table 5.3:** Mean absolute errors with standard deviations of SCK WDK, WDK, LoMoGraph WDK and LoMoGraph on the regression data sets. Statistically significant results are reported using both the Wilcoxon signed-rank test and the corrected resampled t-test (separated by a '|').

Data set	<i>SCK WDK</i> ( $d=1$ )	SCK WDK ( $d \neq 1$ )	WDK	LoMoGraph WDK	LoMoGraph
4QSAR COX2	$0.673 \pm 0.088$	$0.691 \pm 0.056$ ●●	$0.683 \pm 0.055$	$0.676 \pm 0.053$	$0.868 \pm 0.135$ ●●
4QSAR DHFR	$0.669 \pm 0.054$	$0.805 \pm 0.132$ ●●	$0.733 \pm 0.051$ ●●	$0.702 \pm 0.047$ ●	$0.955 \pm 0.102$ ●●
CPD MOUSE	$0.827 \pm 0.052$	$0.874 \pm 0.085$ ●	$0.888 \pm 0.063$ ●●	$0.880 \pm 0.052$ ●●	$1.276 \pm 0.154$ ●●
CPD RAT	$1.002 \pm 0.048$	$1.070 \pm 0.103$ ●	$1.129 \pm 0.154$ ●●	$1.043 \pm 0.048$ ●●	$1.529 \pm 0.116$ ●●
ISS MOUSE	$0.798 \pm 0.049$	$0.826 \pm 0.053$ ●●	$0.850 \pm 0.056$ ●●	$0.859 \pm 0.062$ ●●	$1.197 \pm 0.111$ ●●
ISS RAT	$0.977 \pm 0.064$	$1.018 \pm 0.074$ ●●	$1.031 \pm 0.061$ ●●	$1.022 \pm 0.062$ ●●	$1.371 \pm 0.109$ ●●
Suth COX2	$0.612 \pm 0.042$	$0.620 \pm 0.047$ ●	$0.603 \pm 0.041$ ○	$0.610 \pm 0.044$	$0.905 \pm 0.137$ ●●
Suth DHFR	$0.625 \pm 0.036$	$0.639 \pm 0.086$ ●	$0.633 \pm 0.032$ ●	$0.610 \pm 0.031$ ○	$0.941 \pm 0.066$ ●●
Suth ER TOX	$0.993 \pm 0.064$	$1.301 \pm 0.439$ ●●	$1.175 \pm 0.071$ ●●	$1.130 \pm 0.076$ ●●	$1.216 \pm 0.141$ ●●
FDAMDD	$0.727 \pm 0.029$	$0.809 \pm 0.077$ ●●	$0.833 \pm 0.329$ ●●	$0.727 \pm 0.029$	$0.951 \pm 0.051$ ●●
Biodeg	$1.011 \pm 0.072$	$1.051 \pm 0.119$ ●	$1.110 \pm 0.071$ ●●	$1.086 \pm 0.080$ ●●	-
Tox09	$0.440 \pm 0.021$	$0.643 \pm 0.168$ ●●	$0.464 \pm 0.017$ ●●	$0.425 \pm 0.017$ ○	-
ER_LIT	$0.594 \pm 0.040$	$0.590 \pm 0.058$ ○	$0.609 \pm 0.040$ ●	$0.591 \pm 0.043$ ●	-

●,○ statistically significant improvement, or degradation of SCK WDK ( $d=1$ ) over the other methods

choice of setting the diagonal entries in the kernel matrix has a different effect on both SCK methods. Whereas setting the diagonal entries of the kernel matrix unequal to one leads to

**Table 5.4:** Classification accuracies with standard deviations of SCK NSPDK, NSPDK, LoMoGraph NSPDK and LoMoGraph on the classification data sets. Statistically significant results are reported using both the Wilcoxon signed-rank test and the corrected resampled t-test (separated by a '|').

Data set	<i>SCK NSPDK</i> ( $d \neq 1$ )	SCK NSPDK ( $d=1$ )	NSPDK	LoMoGraph NSPDK	LoMoGraph
CYP INH	$76.33 \pm 2.35$	$75.62 \pm 2.52$ ●	$75.55 \pm 2.50$ ●	$75.88 \pm 2.56$ ●	$74.08 \pm 2.55$ ●●
CYP SUB	$76.84 \pm 2.28$	$75.11 \pm 2.84$ ●●	$75.95 \pm 2.17$ ●	$76.21 \pm 1.99$ ●	$71.37 \pm 2.46$ ●●
Fontaine	$95.56 \pm 1.57$	$95.37 \pm 1.57$ ●	$95.71 \pm 1.66$	$95.62 \pm 1.60$	$92.08 \pm 2.01$ ●●
NCI AIDS	$90.13 \pm 1.58$	$89.86 \pm 1.69$ ●	$90.58 \pm 1.35$ ●	$89.02 \pm 1.62$ ●●	$84.93 \pm 1.61$ ●●
CPDB MUT	$76.71 \pm 2.10$	$75.15 \pm 2.20$ ●●	$77.25 \pm 2.12$ ○	$73.85 \pm 2.50$ ●●	-

●,○ statistically significant improvement, or degradation of SCK NSPDK ( $d \neq 1$ ) over the other methods

**Table 5.5:** Classification accuracies with standard deviations of SCK WDK, WDK, LoMoGraph WDK and LoMoGraph on the classification data sets. Statistically significant results are reported using both the Wilcoxon signed-rank test and the corrected resampled t-test (separated by a '|').

Data set	<i>SCK WDK</i> ( $d=1$ )	SCK WDK ( $d \neq 1$ )	WDK	LoMoGraph WDK	LoMoGraph
CYP INH	$74.05 \pm 3.02$	$70.78 \pm 3.85$ ●●	$75.05 \pm 2.51$ ○	$75.42 \pm 2.42$ ○	$74.08 \pm 2.55$
CYP SUB	$72.07 \pm 3.84$	$70.63 \pm 4.63$ ●	$75.77 \pm 2.38$ ○○	$75.74 \pm 2.46$ ○○	$71.37 \pm 2.46$ ●
Fontaine	$94.01 \pm 1.86$	$94.48 \pm 1.83$ ○	$94.41 \pm 1.54$ ○	$93.03 \pm 5.14$ ●	$92.08 \pm 2.01$ ●●
NCI AIDS	$84.28 \pm 2.04$	$83.67 \pm 2.20$ ●	$79.97 \pm 8.32$ ●●	$82.27 \pm 1.82$ ●●	$84.93 \pm 1.61$ ●
CPDB MUT	$72.80 \pm 2.48$	$71.51 \pm 2.88$ ●	$73.29 \pm 2.57$	$70.85 \pm 2.64$ ●	-

●,○ statistically significant improvement, or degradation of SCK WDK ( $d=1$ ) over the other methods

better predictive performance for SCK NSPDK, setting the diagonal entries equal to one results in better predictive performance for SCK WDK. In the following, the performance of the SCK approaches is analyzed on the different data sets. On the COX2 data sets, no performance improvement of SCK NSPDK and SCK WDK is observed over the respective base kernel. The data sets contain extremely similar molecules, often differing in only one atom. Hence, the base kernel cannot be improved by the similarities induced by the structural clustering procedure. For the CPD, ISS and Biodeg data sets, a comparison between the mean absolute errors shows a clear performance advantage of SCK using both WDK and NSPDK as base kernel. Primarily, this positive effect can be explained as a result of the structurally heterogeneity of the data sets consisting of many small molecules ( $\sim$  up to 10 atoms). Hence, the NSPDK alone is not suited to determine similarity between graphs. As a consequence, for these data sets the pairwise similarities between the small, structurally homogeneous neighborhoods can contribute to similarity and consequently to predictive performance. On FDAMDD, the proposed structural cluster kernel with NSPDK as base kernel yields performance degradation compared to NSPDK. This shows that taking into account the similarities induced by PSCG has an adverse effect on the

predictive performance. Although for this data set a significant performance gain of SCK over the base kernel can be achieved by using WDK as base kernel, SCK WDK still has a higher mean absolute error compared to NSPDK. This demonstrates that NSPDK is much more powerful compared to WDK. For NCI AIDS and both CYP data sets the results on classification are clearly in favor of SCK NSPDK. On these data sets the structural cluster kernel with NSPDK improves over all other compared methods. However, for the corrected resampled t-test only four of the nine wins are statistically significant. Using WDK as base kernel, SCK can only achieve strong performance improvements on NCI AIDS. On the remaining classification data sets taking into account similarities induced by PSCG has either no significant effect or an adverse effect on predictive accuracy compared to the baseline methods (except for LoMoGraph on the Fontaine data set). In summary, the structural cluster kernel approach SCK is comparative to other methods, yet shows a strong performance increase on structurally more sparse data sets, i.e., chemically and structurally more diverse data sets. On these data sets the base kernel alone is not suited to determine similarities between graphs due to the high structural heterogeneity within the data set. Hence, the structural neighborhood of two graphs can substantially contribute to graph similarity and therefore to predictive performance of the constructed models.

### 5.3.2 Semi-Supervised Setting

This section investigates whether incorporating unlabeled data in the clustering process can positively contribute to predictive performance. Since semi-supervised methods potentially give the greatest benefit when a large amount of unlabeled data is used, the structural cluster kernel approach was tested in large-scale experiments, enriching the training data by a large number of molecules from the vast chemical space. To this end, the ChemDB database was employed, which contains nearly 5 M commercially available small molecules [48, 49], as a source of unlabeled data, randomly sampling 100,000 structures from it. Since in the supervised setting, SCK NSPDK ( $d \neq 1$ ) performs always better than or equal to all methods using WDK as base kernel as well as to SCK NSPDK ( $d=1$ ), LoMoGraph NSPDK and LoMoGraph, SCK was only compared in the semi-supervised setting against SCK NSPDK ( $d \neq 1$ ) and NSPDK. As in the supervised setting, the SVM complexity constant,  $C$ , was selected from  $\{10^{-3}, 10^{-2}, 10^{-1}, 10^0, 10^1, 10^2\}$ . Further, the same parameter setting were used for the NSPDK and the similarity coefficient  $\theta$  of PSCG.

The experimental results are shown in Tables 5.6 and 5.7 and in the bar charts in Figures 5.3 and 5.4. For completeness, the bar charts also depict the results for LoMoGraph NSPDK and LoMoGraph. The following discussion is based on the corrected resampled t-test. The results show that in the semi-supervised setting, SCK NSPDK achieves a strong performance gain on all data sets over the supervised approach: 10 of the 18 wins are statistically significant. For regression, the best results can be achieved on the toxicity data sets consisting of structurally more heterogeneous graphs. The results indicate that incorporating a large set of unlabeled data into the structural clustering process has a def-

inite positive effect on the predictive performance. As opposed to the supervised setting, SCK NSPDK can improve over the base kernel on the FDAMDD data set. This data set is the largest one, comprising structurally heterogeneous molecules. Hence, exploiting a large set of unlabeled data in the clustering step can contribute to graph similarity. The strongest performance gains with respect to NSPDK can be achieved on the classification data sets. Whereas in the supervised setting SCK NSPDK was not able to gain significantly with respect to the base kernel on the classification data sets, the semi-supervised approach shows significant improvements over NSPDK in three out of five cases.

**Table 5.6:** Mean absolute errors with standard deviations of SCK NSPDK in both the semi-supervised and supervised setting and NSPDK on the regression data sets. Statistically significant results are reported using the Wilcoxon signed-rank test and the corrected resampled t-test (separated by a '|').

Data set	<i>SCK NSPDK Semi-Sup</i>	SCK NSPDK (d≠1)	NSPDK
4QSAR COX2	$0.606 \pm 0.055$	$0.607 \pm 0.055$	$0.601 \pm 0.054$ ◊
4QSAR DHFR	$0.548 \pm 0.033$	$0.551 \pm 0.035$	$0.541 \pm 0.033$ ◊
CPD MOUSE	$0.746 \pm 0.043$	$0.751 \pm 0.041$ ●	$0.764 \pm 0.040$ ● ●
CPD RAT	$0.861 \pm 0.046$	$0.868 \pm 0.045$ ● ●	$0.887 \pm 0.043$ ● ●
ISS MOUSE	$0.731 \pm 0.049$	$0.738 \pm 0.046$ ● ●	$0.753 \pm 0.045$ ● ●
ISS RAT	$0.850 \pm 0.050$	$0.860 \pm 0.050$ ● ●	$0.900 \pm 0.046$ ● ●
Suth COX2	$0.556 \pm 0.040$	$0.559 \pm 0.039$ ●	$0.552 \pm 0.038$
Suth DHFR	$0.501 \pm 0.025$	$0.504 \pm 0.026$	$0.493 \pm 0.026$ ◊ ◊
Suth ER TOX	$0.808 \pm 0.041$	$0.828 \pm 0.046$ ● ●	$0.820 \pm 0.042$ ● ●
FDAMDD	$0.608 \pm 0.020$	$0.629 \pm 0.029$ ● ●	$0.612 \pm 0.024$ ●
Biodeg	$0.840 \pm 0.051$	$0.844 \pm 0.051$ ● ●	$0.867 \pm 0.050$ ● ●
Tox09	$0.339 \pm 0.014$	$0.345 \pm 0.015$ ● ●	$0.342 \pm 0.015$ ●
ER_LIT	$0.492 \pm 0.039$	$0.492 \pm 0.039$	$0.495 \pm 0.039$ ●

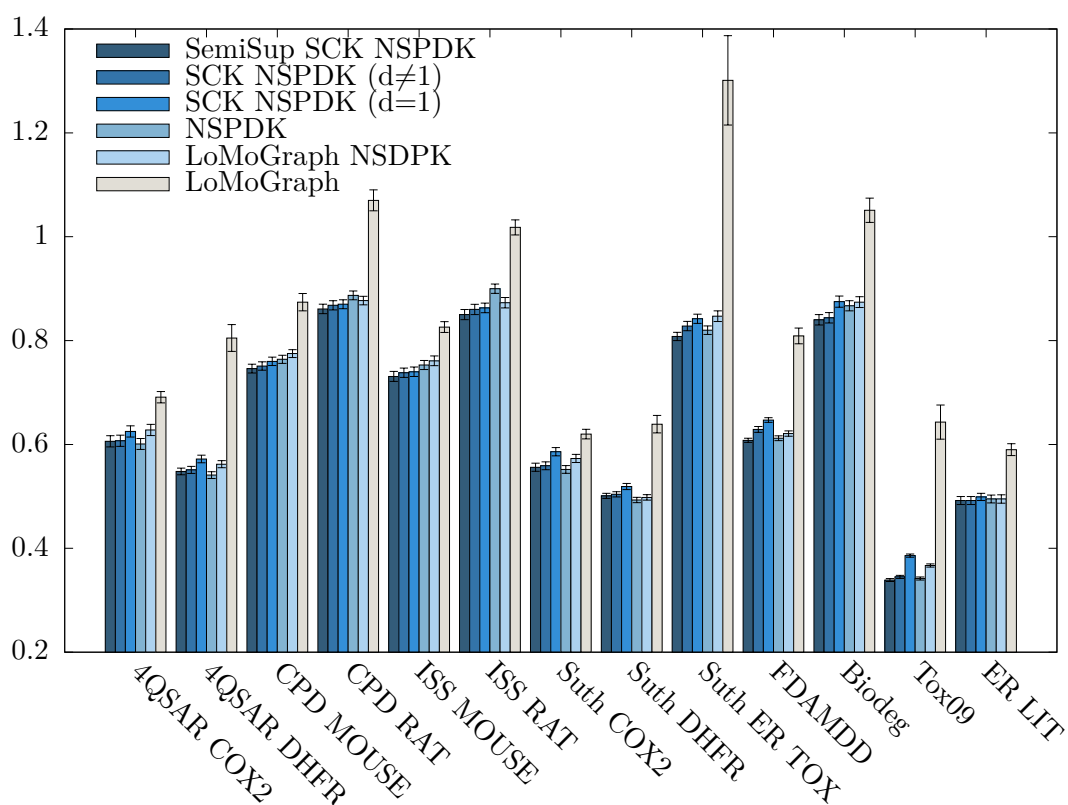
●,◊ statistically significant improvement, or degradation of SCK NSPDK Semi-Sup over the other methods

**Table 5.7:** Classification accuracies with standard deviations of SCK NSPDK in both the semi-supervised and supervised setting and NSPDK on the classification data sets. Statistically significant results are reported using the Wilcoxon signed-rank test and the corrected resampled t-test (separated by a '|').

Data set	<i>SCK NSPDK Semi-Sup</i>	SCK NSPDK (d≠1)	NSPDK
CYP INH	$77.37 \pm 2.26$	$76.33 \pm 2.35$ ● ●	$75.55 \pm 2.50$ ● ●
CYP SUB	$78.78 \pm 2.16$	$76.84 \pm 2.28$ ● ●	$75.95 \pm 2.17$ ● ●
Fontaine	$95.80 \pm 1.42$	$95.56 \pm 1.57$	$95.71 \pm 1.66$
NCI AIDS	$90.92 \pm 1.00$	$90.13 \pm 1.58$ ●	$90.58 \pm 1.35$ ●
CPDB MUT	$78.47 \pm 2.40$	$76.71 \pm 2.10$ ● ●	$77.25 \pm 2.12$ ● ●

●,◊ statistically significant improvement, or degradation of SCK NSPDK Semi-Sup over the other methods





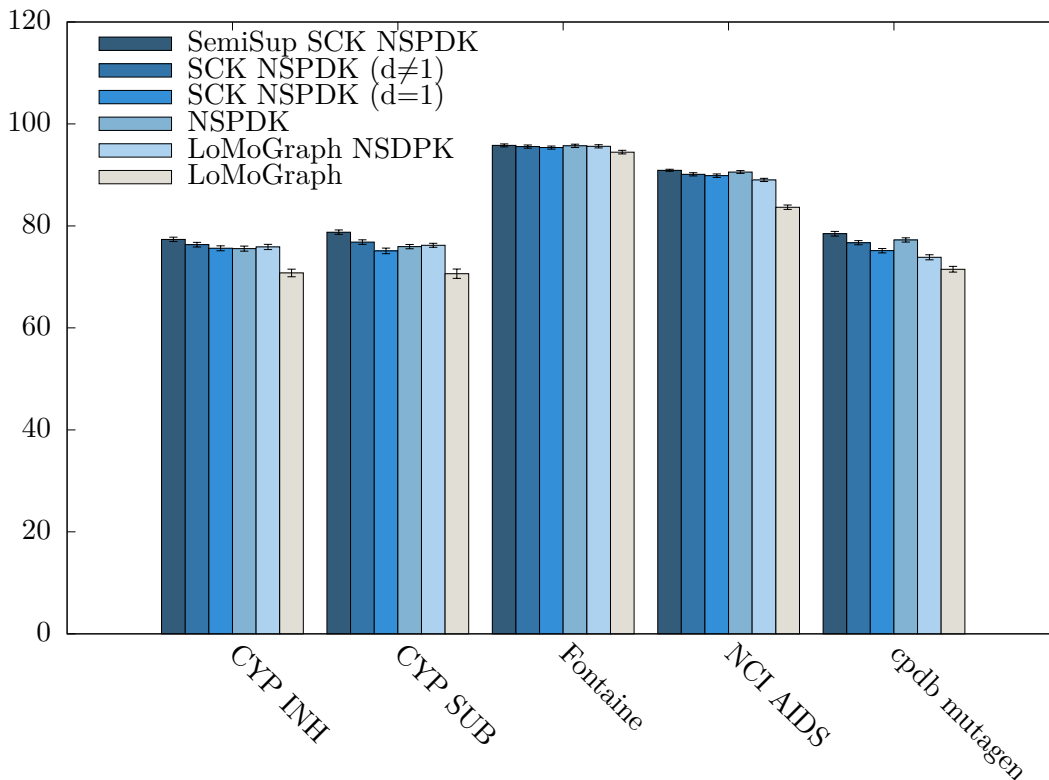
**Figure 5.3:** Mean absolute errors with 95% confidence intervals on the different comparison methods for the regression data sets in Table 5.1.

### 5.3.3 Comparison to Locally Weighted Learning

The structural cluster kernel was further compared against a simple, yet effective method for regression on graphs that combines LWL with MCS-based graph distances. The approach called LWL-MCS was introduced in Chapter 4. To recapitulate, LWL-MCS is a variant of locally weighted regression on graphs that uses the maximum common subgraph for determining and weighting the neighborhood of a graph and feature vectors for the actual regression model.

For LWL-MCS, regression was performed using linear ridge regression. LWL-MCS implements an internal grid search using only the training data to determine the best parameter values via inner 10-fold cross-validation. The number of neighbors,  $k$ , was varied from 25 to 300 using a step size of 25 and the ridge regression parameter  $R$  was varied in the range  $\{1, 10, 25, 50, 100, 150, 200\}$ . The parameter combination resulting in the lowest mean absolute error was then used for building the final model. As for SCK, performance estimates were obtained using 100 times hold-out validation with a training set fraction of 66%. To quantify predictive accuracy, the relative mean error was chosen. The Wilcoxon signed-rank test was applied to test for significant differences at a significance level of 5%.

For the experiments, the first ten regression data sets provided in Table 5.1 were used. The experiments are only conducted on the regression data sets, since the LWL-MCS method was developed for regression on graphs. For each data set, standard chemical



**Figure 5.4:** Classification accuracies with 95% confidence intervals on the different comparison methods for the classification data sets in Table 5.1.

descriptors (e.g., molecular weight, LogP, topological diameter, hydrogen bond acceptor/donor, polar surface area, number of atoms/bonds) computed using the cheminformatics library JOELib2 [236] were used. In the experiments, LWL-MCS was compared against the SCK in the semi-supervised and supervised setting. For the supervised setting, the approach that computes the diagonal elements in the kernel matrix in the same way as the non-diagonal elements is used.

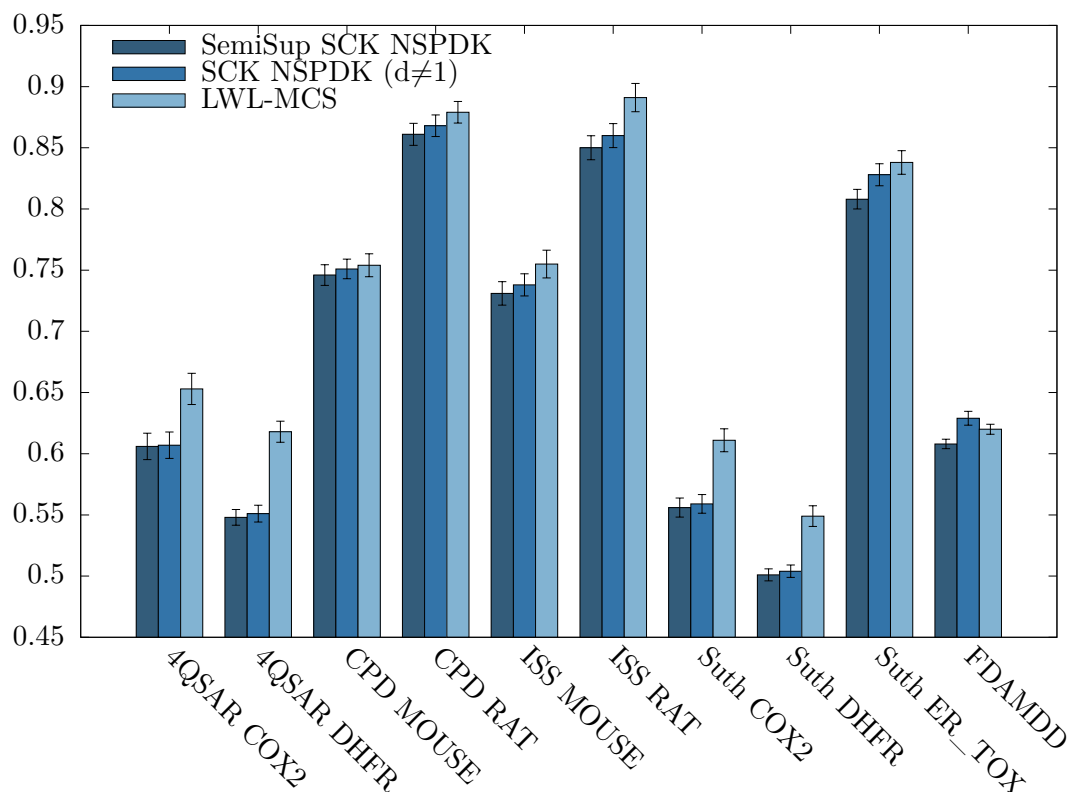
The experimental results are shown in the bar chart in Figure 5.5. The following discussion is based on the corrected resampled t-test. The results show that in the semi-supervised setting, SCK NSPDK achieves a strong performance gain on all data sets over LWL-MCS: four of the ten wins are statistically significant. The best results can be achieved on the data sets consisting of structurally more homogeneous graphs. In the supervised setting, three of the nine wins of SCK NSPDK over LWL-MCS are statistically significant. Only on the FDAMDD data set, LWL-MCS can significantly improve over SCK NSPDK in the supervised setting. A closer look at the results in Table 5.1 reveals that the performance difference between LWL-MCS and SCK is smaller on the data sets containing structurally more heterogeneous molecules, i.e., on the CPD and ISS data sets and on FDAMDD. On these data sets, the strategy of LWL-MCS to determine the neighborhood of graph instances based on the MCS and to make the actual prediction based on feature vectors seems to work particularly well. Still, LWL-MCS does not reach the

performance of the SCK approach.

**Table 5.8:** Mean absolute errors with standard deviations of SCK NSPDK in both the semi-supervised and supervised setting and LWL-MCS on the regression data sets. Statistically significant results are reported using the Wilcoxon signed-rank test and the corrected resampled t-test (separated by a '|').

Data set	<i>SCK NSPDK Semi-Sup</i>	SCK NSPDK ( $d \neq 1$ )	LWL-MCS
4QSAR COX2	$0.606 \pm 0.055$ ●	$0.607 \pm 0.055$ ●	$0.65252 \pm 0.00652$
4QSAR DHFR	$0.548 \pm 0.033$ ●●	$0.551 \pm 0.035$ ●●	$0.61831 \pm 0.00444$
CPD MOUSE	$0.746 \pm 0.043$ ●	$0.751 \pm 0.041$ ●	$0.75381 \pm 0.00478$
CPD RAT	$0.861 \pm 0.046$ ●	$0.868 \pm 0.045$ ●	$0.87851 \pm 0.00453$
ISS MOUSE	$0.731 \pm 0.049$ ●	$0.738 \pm 0.046$ ●	$0.75470 \pm 0.00575$
ISS RAT	$0.850 \pm 0.050$ ●	$0.860 \pm 0.050$ ●	$0.89128 \pm 0.00592$
Suth COX2	$0.556 \pm 0.040$ ●●	$0.559 \pm 0.039$ ●●	$0.61090 \pm 0.00482$
Suth DHFR	$0.501 \pm 0.025$ ●●	$0.504 \pm 0.026$ ●●	$0.54921 \pm 0.00262$
Suth ER TOX	$0.808 \pm 0.041$ ●	$0.828 \pm 0.046$ ●	$0.83777 \pm 0.00488$
FDAMDD	$0.608 \pm 0.020$ ●●	$0.629 \pm 0.029$ ○	$0.62019 \pm 0.00213$

●,○ statistically significant improvement, or degradation of SCK NSPDK Semi-Sup and SCK NSPDK over LWL-MCS



**Figure 5.5:** Mean absolute errors with 95% confidence intervals on the different comparison methods for the first ten regression data sets in Table 5.1.

## 5.4 Conclusion

This chapter proposed a novel graph kernel approach that incorporates similarity information based on structural graph clustering to improve state-of-the-art graph kernels. The proposed kernel is based on the idea that graph similarity can not only be determined by the similarity of the graphs alone, i.e., their structure, but also by the similarity of the graphs' structural neighborhood. The performance of the structural cluster kernel was investigated for regression and classification by using several real-world data sets of molecular graphs. In the experiments, a comparison to the weighted decomposition kernel, the neighborhood subgraph pairwise distance kernel, a learning method combining clustering with classification or regression for the prediction task and a method that combines Locally Weighted Learning with Maximum Common Subgraph based graph distances for regression on graphs was conducted. The results demonstrate that the proposed kernel approach yields an increase in performance on a number of data sets, in particular on structurally more diverse data sets. The performance of the SCK was further investigated in the semi-supervised setting, by enriching relatively small labeled data sets by a large set of unlabeled data instances from the vast chemical space. The results show that within the semi-supervised setting the proposed approach achieves gains in performance when compared to the supervised version as well as to the pure base kernel, in particular for classification. I believe that the approach presented is general as such, and can also be employed in conjunction with a variety of different kernels and clustering approaches and is therefore not restricted to graph mining alone.

# CHAPTER 6

---

## Mining Support Vectors of the Structural Cluster Kernel

---

Statistical machine learning algorithms building on patterns found by pattern mining algorithms have to cope with large solution sets and thus the high dimensionality of the feature space. Vice versa, pattern mining algorithms are frequently applied to irrelevant instances, thus causing noise in the output. Solution sets of pattern mining algorithms also typically grow with increasing input data sets. This chapter proposes an approach to overcome these limitations. The approach extracts information from trained support vector machines, in particular their support vectors and their relevance according to their coefficients. It uses the support vectors along with their coefficients as input to pattern mining algorithms able to handle weighted instances. The experiments in the domain of graph mining and molecular graphs show that the resulting models are not significantly less accurate than models trained on the full data sets, yet require only a fraction of the time using much smaller sets of patterns.

### 6.1 Introduction

Two of the most important families of classification methods for structured data (like graphs, trees, and sequences) are kernel-based and pattern-based methods. Kernel-based methods, on the one hand, are often superior in terms of predictivity, but frequently lack the possibility of interpretation and extraction of relevant features. Pattern-based methods, on the other hand, are often less accurate, but offer the advantage of explicit feature spaces, for instance, for the inspection of feature usage or feature weights. As pattern-based classification methods build upon the output of pattern-mining algorithms, they are heavily affected by the typically huge solution sets produced by those algorithms. Despite enormous progress in the area of condensed representations and compression methods for pattern sets (see, e.g., BBRCs [163] and KRIMP [231]), the output size may, in some cases, remain too large for practical application.

Kernel-based methods from the first family of methods, based on SVMs, are often considered as state-of-the-art classification methods in machine learning. An important advantage of SVMs is that their classification decision is based on a subset of the training

examples, referred to as the support vectors. However, a drawback of SVMs is their black box character. The generated non-linear models typically lack interpretability. Therefore, the topic of opening the black box or making SVMs interpretable has received considerable attention in the literature, e.g. in areas such as credit evaluation, graph reconstruction and others [159, 14, 239]. Martens *et al.*, for example, attempted to mimic the behavior of SVM models and to add comprehensibility to them by extracting rules from the trained models [159]. Another attempt to extract information from SVMs was made by Bakir *et al.* [14]. The authors trained an SVM regression model to represent the inverse mapping from the feature space to the input space, thus obtaining a pre-image function. This enables sampling of novel instances into the input space. Weston *et al.* [239] considered the task of learning dependencies between a general class of objects. Their approach referred to as Kernel Dependency Estimation uses a kernel principal component analysis (PCA) to implicitly model correlations among both inputs and outputs. Kernel Dependency Estimation decouples output correlations by first applying Kernel PCA over the outputs and then learning the mappings from the input space to dimension-reduced space by ridge regression. A pre-image calculation is required in order to recover the output in the original representation. Whereas pre-image methods are well-established for vectorial data, their usage for graph data seems limited so far. In particular, there is no well-known approach for deriving an explicit graph-based feature representation from a trained SVM with a graph kernel.

In the light of this, the chapter addresses the following question: Can we make use of trained SVM models to obtain more compact pattern-based classification models? There are at least two possible ways of doing so: by analyzing a trained SVM model together with the training set, or by using the models as oracles to label instances [62].

In this chapter, the former of the two approaches is studied. As the wish is take advantage of any SVM with a graph-based kernel and all the information regarding the classification is actually contained in the graph, the feature space of the graph kernel is not necessarily assumed to correspond to the pattern language of the graph miner.

This work investigates this question and a possible solution in the context of graphs and, in particular, molecular graphs. Many graph kernels have been proposed in this domain, including the random walk graph kernel [88, 229], the optimal assignment kernel [82], the shortest-path graph kernel [32], the subtree pattern kernel [211], the neighborhood subgraph pairwise distance kernel [61] and the structural cluster kernel proposed in Chapter 5. As mentioned above, a major drawback of kernel approaches is the lack of interpretability, as it may be difficult to figure out which features play an important role in classification. On the other hand, graph pattern-based approaches build graph classifiers based on different types of graph substructure features. The basic idea is to extract frequent substructures, local graph fragments, or cyclic patterns and trees and use them as descriptors to represent the graph data. A major problem with pattern mining is that it usually generates too many patterns on training data, many of which are redundant. When the input data sets attain considerable size, the mining process becomes computa-

tionally expensive or simply infeasible. Further, pattern mining algorithms are frequently applied to irrelevant instances. Thus, the interpretation of the results turns out to be a difficult task as the interesting patterns are blurred into the huge amount of outputted patterns.

The approach proposed in this chapter aims to overcome some of the aforementioned limitations. More specifically, the approach is motivated by the question whether a small set of representative patterns can be derived from a set of relevant structures extracted from a given data set such that the classifier trained on this pattern set still leads acceptable prediction performance. To this end, graph mining techniques are combined with graph kernel-based classifiers. To extract a set of relevant graph instances from a given input data, an SVM is trained on the recently proposed graph kernel, called the structural cluster kernel (SCK). The basic idea of the SCK is to improve a base graph kernel by incorporating similarity information induced by a structural graph clustering algorithm. The extracted graph instances, i.e., the support vectors, and their according weights reflecting the relevance of each support vector are then used as input to a pattern mining algorithm that can cope with weighted instances. More specifically, an extension of the pattern mining algorithm Backbone Refinement Class Mining (BBRC) [163] that mines compact sets of subgraph descriptors from a set of weighted graphs is employed. By considering only the support vectors for the mining process, i.e., the instances that are critical for classification, it is expected that the resulting models achieve similar or at least not significantly worse accuracy than the model trained on the full data set. The experiments in the domain of molecular graphs indeed show that on most of the data sets, in particular on data sets comprising structurally more homogeneous graphs, the models trained on the support vectors are not significantly less accurate than models trained on the full data sets, yet require only a fraction of the time while using less patterns.

The chapter is organized as follows: The problem definition is presented in Section 6.2. Section 6.3 presents the approach to information extraction from SVMs for pattern-based classification along with necessary background information about the employed methods. In Section 6.4, an overview of the data sets and baseline methods used in the chapter is given as well as a description of the experimental setup. Further, the Section presents and discusses experimental results quantitatively and qualitatively. Finally, a conclusion is given in Section 6.5.

## 6.2 Problem Definition

The proposed approach investigates the question whether trained SVM models can be employed to obtain more compact pattern-based classification models. To do so, the approach extracts information from trained support vector machines, i.e., their support vectors and their relevance according to their coefficients (weights). It uses the support vectors along with their coefficients as input to pattern mining algorithms able to handle weighted instances. The motivation for this is the following: (a) Only the support vectors,

i.e., the training instances relevant for the classification according to the trained SVM, are being subjected to the pattern mining. (b) These training instances enter the graph mining not with uniform weight, but a weight corresponding to their contribution to the classification according to the SVM’s objective and loss function.

Formally, the problem is framed as follows. Consider the binary classification scenario, and a training data set  $D_{Trg} = \{(x_1, y_1), \dots, (x_t, y_t)\}$ , where  $x_i \in X$  represent the data points and  $y_i \in \{-1, 1\}$  their corresponding class labels. Further, let  $D_{Tst} = \{x_{t+1}, \dots, x_n\}$  denote the set of test instances. By training an SVM on the training data  $D_{Trg}$ , the support vectors  $s = \{s_1, s_2, \dots, s_k\} \subseteq D_{Trg}$  are obtained. Given the set of support vectors  $s$  along with their corresponding weights  $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_k\}$ , the goal of this approach is to extract a small set of representative subgraph patterns  $\{q_1, q_2, \dots, q_m\}$  ( $m \in \mathbb{N}$ ) correlated with the class labels from only the support vectors, such that the classifier trained on the reduced pattern set still yields acceptable, i.e., not significantly worse, classification accuracy.

## 6.3 Method

In this section, the proposed approach to information extraction from SVMs for pattern-based classification is presented. The section starts by introducing the employed approach to pattern mining for mining a compact sets of subgraph descriptors from a set of weighted graphs.

### 6.3.1 Backbone Refinement Class Mining

BBRC Mining [163] is an algorithmic approach to mine compact sets of subgraph descriptors in the search space of chemical structure graphs, creating compressed representations of chemical structure. It can be applied to class-labeled 2D graph databases and combines feature generation and feature selection into one step. The extended version employed in this work, allows to weight individual instances of the graph database according to their importance. BBRC mining creates a sparse selection from the search space of frequent and significant subtrees, based on a weighted minimum frequency, structural and statistical constraints. In the work presented here, weights ( $\beta$ ) for individual instances can be positive real numbers and zero and are not normalized in any way. Furthermore, the minimum frequency constraint can be set to any positive real number, reflecting each instance’s importance in the data set and hence also of the induced patterns. It has very high compression potential, which has been shown theoretically [163] from the unweighted version of BBRCs. Empirical results confirmed the compression results in practice, while retaining good database coverage. Moreover, it has been shown that the structural constraints produce structurally diverse features with low co-occurrence rates. BBRC descriptors compare favorably to other compressed representations in the context of classification models. In classification tasks with either nearest-neighbor or SVM models, the accuracy of models



based on BBRC descriptors was equal to models with the complete set of frequent and significant subtrees, but significantly better than that of other compressed representations. The algorithm performs substructure selection with regard to the endpoint under investigation, and calculates substructure associations to the endpoint in the form of  $p$ -values from a chi-squared distribution test. In the following, the basic concepts of BBRC are recapitulated.

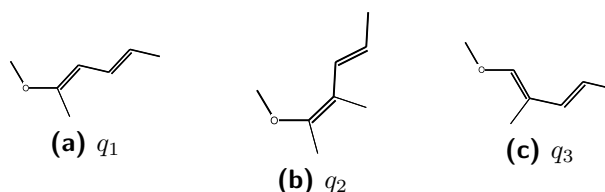
First, the definition of Backbone Refinement Classes is given.

Undirected, labeled graphs are partially ordered via the refinement relation. Here, only acyclic graphs (paths and trees) are considered. Specifically, any path has a sequence, and sequences can be lexicographically ordered. Accordingly, any tree has a backbone, which is defined as the longest path with the lexicographically lowest sequence within the tree. An immediate tree refinement is an addition of a node and an edge to a tree such that it remains a tree, i.e., not possesses a cycle. The *Backbone Refinement Classes* are considered, where the members of each such class are tree refinements of each other and share the same backbone. The backbone refinement class relation is denoted by  $\preceq_b$ . Note that the classes are not disjoint for the same backbone (but they are across different ones). For example, in Figure 6.1,  $q_1$  and  $q_3$  are in different classes, but  $q_2$  is in the respective classes of both  $q_1$  and  $q_3$ .

A categorical target class labeling function for the graphs in the database is assumed, enabling significance tests on trees by calculating their weighted occurrences in the database and therefore in the weighted target classes.

The problem of weighted BBRC Mining can be defined as follows. Given a graph database, corresponding weights, a user-defined minimum support and user-defined minimum  $\chi^2$  value, for all backbone refinement classes within the database, find the smallest (according to  $\preceq_b$ ) of the most significant trees in each class that is *frequent* and *significant* with respect to their weighted occurrences in the target classes. The complexity of BBRC mining is upper-bounded by the complexity of regular tree mining [174]. However, running times are significantly lower for practical applications.

For significance testing, BBRC employs the  $\chi^2$  distribution test using weighted instances. Given a subgraph  $q$ , the original weights  $\beta$ , and  $I$  disjoint target classes  $G_i$ , whose weighted member graphs make up the database, a  $I \times 2$  contingency table is sought that lists  $q$ 's weighted support values per target class in the first column and the overall



**Figure 6.1:** Three example trees with the same backbone (bold). Its sequence is 'c:c:c-C=C-O-C' (reflecting that the fragments include part of an aromatic ring). It also holds that  $q_1 \preceq_b q_2$  and  $q_3 \preceq_b q_2$ , but neither  $q_1 \preceq_b q_3$  nor  $q_3 \preceq_b q_1$ . Therefore,  $q_1$  and  $q_3$  are not in the same Backbone Refinement Class.

weighted distribution of target classes in the second column, as in Table 6.1.

These data serve to check whether  $q$ 's weighted support values differ significantly from the overall weighted class distribution. The  $\chi_d^2$  function is used for distribution testing, defined as

$$\chi_d^2(x, y) = \sum_{i=1}^I \frac{(k_i - E(k_i))^2}{E(k_i)}, \quad (6.1)$$

where  $k_i = \sum_{x_j \in G_i} \text{cover}(q, x_j) \beta_{x_j}$  with  $\text{cover}(q, x_j) \in \{0, 1\}$  denoting whether  $q$  satisfies  $x_j$  (1) or not (0). Furthermore,  $E(k_i) = \frac{|G_i|k}{|G|}$  is the expected value of the weighted  $k_i$ , calculates the sum of squares of deviations from the expected weighted support for all target classes  $G_i$  (where  $|G_i|$  is given by  $|G_i| = \sum_{x_j \in G_i} \beta_{x_j}$ , i.e., the weighted occurrence of all instances for class  $G_i$ ). Similarly,  $|G|$  reflects the weighted occurrences of all instances in the database. The function value is then compared against the  $\chi^2$  distribution function to conduct a significance test with  $I - 1$  degrees of freedom and obtain a  $p$ -value  $p(q)$ . It is possible to calculate an upper bound for the  $\chi^2$  values of refinements of a pattern [171], which can be used for antimonotonic pruning. Using a static, user-defined upper bound threshold is referred to as *static upper bound pruning*. To speed up the search, this threshold (*dynamic upper bound adjustment*) may be increased. For any frequent subtree  $q$ , let  $\chi^2(q, R)$  and  $\chi_u^2(q, R)$  denote the  $\chi^2$  value for  $q$  and  $\chi^2$  upper bound for refinements of  $q$ , respectively. Let  $u_{max}(q) = \max\{\chi^2(p, R) \mid p \preceq_b q\}$ . Then, if  $u_{max}(q) > u$ ,  $u$  may be increased to  $u_{max}(q)$ , since only the maximum class element is searched.

### 6.3.2 Graph Mining On Support Vectors

In this section, the approach for extracting information from support vector machines for pattern-based classification is introduced. The approach extracts information from trained support vector machines, in particular their support vectors and their relevance according to their coefficients (weights). Both the support vectors and their corresponding weights are then used as input to the pattern mining algorithm BBRC (see Section 6.3.1 for a detailed description) that can handle weighted instances. BBRC reflects the weights in its distribution test expressing individual importance of the instances. The intuition behind this approach is the following. It is known that only the support vectors determine

**Table 6.1:** Contingency table for subgraph  $q$ .

	$q$	$all$
class 1	$k_1$	$ G_1 $
class 2	$k_2$	$ G_2 $
...	...	...
class $I$	$k_I$	$ G_I $
$\Sigma$	$k$	$ G $

the final decision boundary of SVM, whereas instances other than support vectors have no contribution to determine the decision boundary. The corresponding weights reflect the relative importance of a graph instance in discriminating the classes. The higher the weight of an instance, the more influential it is. Thus, by incorporating only the instances that are important for classification along with their weights in the pattern mining process, it is expected that the resulting models yield similar predictive performance compared to the models trained on the full data set.

Formally, let  $D_{Trg} = \{(x_1, y_1), \dots, (x_t, y_t)\}$  denote the training data, where  $x_i$  represent the graph instances and  $y_i \in \{-1, 1\}$  their class labels, respectively. Further, let  $D_{Tst} = \{x_{t+1}, \dots, x_n\}$  represent the test instance. To extract relevant information from the given training data, the approach employs the structural cluster kernel SCK presented in Chapter 5. The graph kernel is based on the assumption that graph similarity can not only be determined by the similarity of the graphs alone, i.e., their structure, but also by the similarity of the graphs' structural neighborhood. Following this idea, the SCK incorporates similarity information induced by a structural graph clustering algorithm (see Section 3.2) to improve a base kernel. Chapter 5 evaluated the SCK using two different state-of-the-art graph kernels. In this chapter, the NSPDK is used, since this kernel yields better performance results with respect to predictive accuracy than the WDK.

The approach in this chapter starts by constructing the structural cluster kernel SCK from the training data  $D_{Trg}$  and by training an SVM on the resulting kernel. Given the model trained on the SCK, the graph instances are extracted whose corresponding SVM coefficients are nonzeros, i.e., the support vectors, along with their weights.

Formally, let  $s = \{s_1, s_2, \dots, s_k\}$  denote the set of extracted support vectors associated with the training data  $D_{Trg}$  and let  $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_k\}$  represent their corresponding weights. The weights are normalized such that the sum of all weights equals the number of training data, i.e., each weight  $\alpha_j$ ,  $1 \leq j \leq k$ , is normalized according to  $\beta_j = \frac{|D_{Trg}|}{\sum_l \alpha_l}$ . Next, the support vectors and the modified weights  $\beta$  are used as input for BBRC to derive a set of class-correlated subgraph patterns  $\{q_1, q_2, \dots, q_m\}$ . As described in Section 6.3.1, BBRC incorporates weights for instances in the  $\chi^2$  distribution test expressing individual importance of the instances.

Given the resulting subgraph patterns, a feature vector is constructed for each support vector representing the mined subgraph patterns. Formally, each instance is represented from the set of support vectors,  $s_i$ , by a binary feature vector  $f_{s_i} = [f_{s_i}^1, f_{s_i}^2, \dots, f_{s_i}^m]$  corresponding to the set of mined subgraph patterns  $\{q_1, q_2, \dots, q_m\}$ . Each element  $j \in \{1, \dots, m\}$  in the feature vector  $f_{s_i}$  indicates the presence and absence of the corresponding subgraph pattern  $q_j$  in the graph object  $s_i$ . Next, a linear SVM is trained on the support vectors using the feature vectors constructed from the derived subgraph patterns.

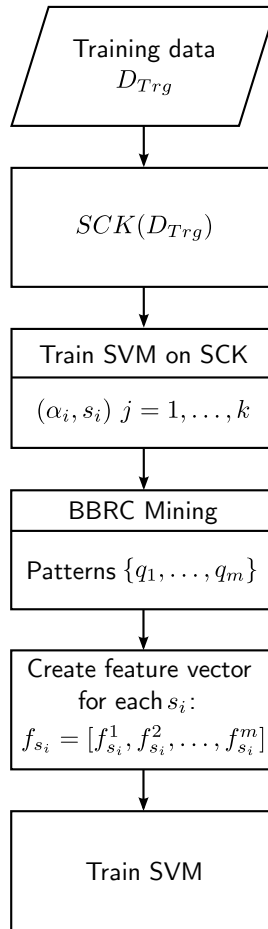
Given a test set  $D_{Tst}$ , the subgraph patterns  $\{q_1, q_2, \dots, q_m\}$  obtained by performing graph mining on the support vectors are matched back onto the test instances. For each test instance, a feature vector is created indicating the occurrence of each subgraph pattern in the test instance. The selected features are then evaluated by the accuracy of

classification.

To summarize, given the training points  $D_{Trg} = \{(x_1, y_1), \dots, (x_t, y_t)\}$ , where  $x_i \in X$ ,  $1 \leq i \leq t$ , represent the data points and  $y_i \in \{-1, 1\}$  ( $i = 1, \dots, n$ ) their corresponding class labels, and test points  $D_{Tst} = \{x_{t+1}, \dots, x_n\}$ , the approach performs the following steps:

1. Construct the SCK on the training data  $D_{Trg}$  and train an SVM with the SCK.
2. Extract the set of support vectors  $s = \{s_1, s_2, \dots, s_k\}$  associated with  $D_{Trg}$  and their corresponding weights  $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_k\}$  obtained by training an SVM with the SCK.
3. Normalize each weight  $\alpha_j$ ,  $1 \leq j \leq k$ , according to the formula  $\beta_j = \frac{|D_{Trg}|}{\sum_l \alpha_l}$  and use the extracted support vectors  $s$  and their weights  $\beta$  as input for BBRC to derive a set of subgraph patterns  $\{q_1, \dots, q_m\}$ .
4. For each support vector  $s_i$ , create a feature vector  $f_{s_i} = [f_{s_i}^1, f_{s_i}^2, \dots, f_{s_i}^m]$  corresponding to the set of derived subgraph patterns  $\{q_1, \dots, q_m\}$ .
5. Use the feature vectors associated with the support vectors to train a linear SVM.
6. For each test instance  $x_j \in D_{Tst}$ :
  - a) Match the subgraph patterns  $\{q_1, \dots, q_m\}$  back onto  $x_j$ .
  - b) Create a feature vector  $f_{x_j} = [f_{x_j}^1, f_{x_j}^2, \dots, f_{x_j}^m]$  corresponding to the matched subgraph patterns.
  - c) Classify the test instance  $x_j$  according to the prediction of the model built in step 5.

Figure 6.2 illustrates the steps of the approach in a flowchart.



**Figure 6.2:** Flowchart of the proposed approach for extracting information from support vector machines for pattern-based classification.

## 6.4 Experiments

In this section, the performance of the method proposed in this chapter is studied. The goal is to investigate whether reducing the entire training data to a set of  $k$  support vectors that have the highest contribution to classification still yields acceptable classification performance. The section presents the baseline methods, the data sets, the experimental setup and the results.

### 6.4.1 Baseline Methods

To investigate the effectiveness of the proposed approach in terms of its ability to reduce the training set size while maintaining the generalization performance, it was compared against a method that calculates a set of subgraph patterns from the entire training data  $D_{Trg}$ . The derived subgraph patterns are then used as features to build a classification model using a linear SVM. Similar to the proposed approach, the method applies BBRC for computing the subgraph patterns.

Further, the approach is compared against a method that builds a classification model

on a reduced feature set obtained by conducting feature selection on the training data. The following two approaches are employed for feature selection. The first approach to feature selection, referred to as  $FS_{\text{top-k}}$ , produces a ranked list of attributes using a linear SVM as attribute evaluator and specifies a number of top-ranked attributes to retain. In the experiments, the number of top-ranked attributes corresponds to the number of features obtained by the proposed approach by using 60% of the training data as support vectors for model building. The second approach, referred to as FS, also produces a ranked list using a linear SVM as attribute evaluator. It then steps through this list evaluating each subsequently larger subset (i.e., top attribute, top two attributes etc.) using an SVM-based subset evaluator. For both approaches, a linear SVM is trained over the training data represented by the reduced feature set.

#### 6.4.2 Data Sets

For the experiments, the chemical domain is employed as the application area by using real data sets of molecular graphs. Table 6.2 provides an overview of the data sets. All of the data sets are associated with a classification endpoint, e.g., carcinogenicity.

#### 6.4.3 Experimental Setup

In all experiments, classification was performed using SVMs as the classifier. To build a classification model using the SCK,  $\nu$ -SVM is used as classifier [200] (see Section 2.3.4.3). The fraction of support vectors was controlled by choosing  $\nu$  such that the number of support vectors covers a specific fraction of the training samples. For the experiments, the number of support vectors is required to cover 50% and 60% of the training data, respectively. For each data set, three values for the minimum frequency (MF) parameter of BBRC were selected. For *NCI\_AIDS*, *dhfr* and *bloodbarr*  $MF = 6\%$ ,  $8\%$  and  $10\%$  were chosen, respectively. Due to the structural diversity of the *kazius* data set, smaller values for the MF were chosen, since higher values result in too few patterns. On the other hand, for the structural homogeneous data sets *er\_tox* and *Fontaine*, higher values for the

**Table 6.2:** Overview of the data sets used for assessing the method.  $n$  denotes the number of graph instances in the respective data set and *Tanimoto Sim.* describes the mean pairwise Tanimoto similarity using ChemAxon’s chemical fingerprint with default parameter setting.

Data set	$n$	Tanimoto Sim.	Proportion positive class
kazius [129]	4337	0.159	0.41
dhfr [216]	393	0.428	0.32
bloodbarr [145]	413	0.237	0.67
Fontaine [80]	435	0.461	0.64
er_tox [216]	446	0.416	0.41
NCI_AIDS [54]	1000	0.305	0.42

MF were chosen, since for smaller values BBRC generates too many features resulting in an increased computational complexity. To build a classification model on the subgraph patterns, a linear SVM was used. The trade-off between training error and margin,  $C$ , was optimized by internal cross-validation selecting from  $\{10^{-3}, 10^{-2}, 10^{-1}, 10^0, 10^1, 10^2\}$ . The parameter resulting in the the highest accuracy was then used for building the final model. All other SVM parameters were left at their default values. Performance estimates were obtained using 15 times hold-out validation using the same data for training and testing for all comparison methods. According to Nadeau and Bengio [172], a 15 holdout run provides good power, in terms of the probability of rejecting the null hypothesis when it is false, with reasonable computational effort, whereas going beyond 15 gives little additional power and is probably not worth the computational effort. To quantify predictive accuracy, the classification accuracy was chosen, which is a standard measure in classification settings. To test for significant differences between the methods the corrected resampled t-test [172] at the 5% significance level was used. Further, the number of computed subgraph patterns and the runtime for model building and prediction were investigated.

#### 6.4.4 Results

Table 6.3 shows the detailed experimental results in terms of classification accuracy, number of computed subgraph patterns and runtime performance (time for model building and prediction) for the various methods on all data sets. The corrected resampled t-test is used to indicate whether the proposed approach using both 50% and 60% support vectors is significantly better or worse than the method employing all the training data at  $p < 0.05$ . Further, the same table reports whether the feature selection approaches,  $FS_{top-k}$  and FS, are significantly better or worse than the proposed approach using 60% support vectors.

The results show that the proposed approach achieves the best performances in terms of prediction accuracy on the data sets comprising structurally more homogeneous graphs, i.e., on all data sets except for *kazius*. On these data sets using 60% of the training data as support vectors the approach yields similar predictive performance and at the same time employs less features and requires only a fraction of the time compared to the approach that uses the entire training data to build a model. On the other hand, the results on the *kazius* toxicity data set show that the models trained on the support vectors are significantly less accurate than models trained on the full data sets. Primarily, this negative effect can be explained as a result of the structurally heterogeneity of the examples. On this data set the proposed approach yet generates a much smaller pattern set and at the same time requires less time. However, the patterns generated from only the support vectors are not sufficient for classification, indicated by significantly worse classification accuracies. Reducing the fraction of support vectors to 50% results in a decrease in performance with respect to prediction accuracy. However, on the *dhfr*, *bloodbarr* and *er\_tox* data sets and on *Fontaine* for MF=22% the performance differences with respect to the approach using the full training data for graph mining are not statistically significant. For better

**Table 6.3:** Classification accuracies (ACC), number of attributes (#Feats) and runtime (in sec). MF denotes the minimum frequency parameter of BBRC.

<b>Fontaine</b>		MF=22%	MF=20%	MF=18%
$D_{Trg}$	ACC	91.26 ± 1.90	92.30 ± 1.96	92.34 ± 2.86
	#Feats	4864.3 ± 3486.3	4923.6 ± 3483.7	5868.6 ± 2952.2
	Time	25.5 ± 22.7	60.0 ± 69.9	21.9 ± 18.1
SV <sub>60%</sub>	ACC	90.59 ± 2.11	91.12 ± 1.87	91.40 ± 2.52
	#Feats	140.6 ± 119.0	237.3 ± 151.5	291.9 ± 169.5
	Time	2.8 ± 4.3	3.7 ± 5.5	4.5 ± 5.6
SV <sub>50%</sub>	ACC	89.14 ± 2.73	89.28 ± 2.66 ◦	89.64 ± 3.16 ◦
	#Feats	113.8 ± 120.3	143.7 ± 153.5	176.45 ± 159.3
	Time	4.0 ± 9.2	2.8 ± 5.2	4.7 ± 7.5
FS <sub>top-k</sub>	ACC	86.26 ± 8.05 ◦	90.26 ± 3.40	90.96 ± 2.57
	#Feats	140.6 ± 119.0	237.3 ± 151.5	291.9 ± 169.5
	Time	763.4 ± 871.8	1893.2 ± 3418.9	1135.4 ± 2086.8
FS	ACC	90.63 ± 1.72	91.08 ± 2.47	91.36 ± 1.84
	#Feats	65.7 ± 24.2	104.8 ± 45.3	144.8 ± 132.1
	Time	8771.4 ± 9873.8	9998.2 ± 13405.2	18068.5 ± 25696.9
<b>er_tox</b>		MF=12%	MF=10%	MF=8%
$D_{Trg}$	ACC	75.31 ± 2.62	75.26 ± 2.78	75.00 ± 2.75
	#Feats	7927.5 ± 2135.3	7998.0 ± 2171.3	8131.2 ± 1867.0
	Time	74.9 ± 31.1	75.0 ± 29.7	75.5 ± 32.1
SV <sub>60%</sub>	ACC	75.18 ± 2.68	75.18 ± 3.37	75.35 ± 3.87
	#Feats	5826.3 ± 2730.1	6796.1 ± 2823.1	7474.5 ± 3045.7
	Time	30.0 ± 18.9	31.9 ± 22.5	37.5 ± 23.1
SV <sub>50%</sub>	ACC	70.79 ± 5.49	71.14 ± 8.17	72.63 ± 3.88
	#Feats	2000.4 ± 3321.0	3473.6 ± 4487.6	3833.8 ± 5150.2
	Time	5.3 ± 13.0	7.6 ± 14.0	12.7 ± 26.4
FS <sub>top-k</sub>	ACC	74.66 ± 2.39	74.76 ± 2.46	74.83 ± 2.89
	#Feats	5826.3 ± 2730.1	6796.1 ± 2823.1	7474.5 ± 3045.7
	Time	2056.1 ± 1557.4	2882.7 ± 3329.3	3646.1 ± 2924.7
FS	ACC	74.61 ± 1.22	74.91 ± 1.25	75.10 ± 1.99
	#Feats	501.0 ± 159.4	727.0 ± 161.0	971.0 ± 280.0
	Time	28460.2 ± 23104.2	30508.4 ± 28020.3	34460.2 ± 30104.2
<b>bloodbarr</b>		MF=10%	MF=8%	MF=6%
$D_{Trg}$	ACC	71.87 ± 2.88	70.83 ± 4.00	71.87 ± 2.64
	#Feats	73.4 ± 27.0	141.3 ± 66.8	321.2 ± 173.4
	Time	15.4 ± 9.2	18.3 ± 7.6	29.0 ± 67.2
SV <sub>60%</sub>	ACC	72.81 ± 2.92	72.96 ± 3.10	73.43 ± 2.77
	#Feats	35.8 ± 10.8	52.1 ± 17.5	142.6 ± 137.5
	Time	1.8 ± 3.9	1.4 ± 1.3	2.2 ± 3.9
SV <sub>50%</sub>	ACC	69.31 ± 3.57	69.22 ± 2.91	70.02 ± 3.34
	#Feats	21.4 ± 10.0	41.8 ± 21.6	87.6 ± 50.7
	Time	1.8 ± 4.4	1.2 ± 2.1	2.2 ± 5.0
FS <sub>top-k</sub>	ACC	71.89 ± 2.96	70.35 ± 3.84 ◦	71.73 ± 3.43
	#Feats	35.8 ± 10.8	52.1 ± 17.5	142.6 ± 137.5
	Time	108.07 ± 282.33	138.65 ± 147.33	140.16 ± 27.19
FS	ACC	72.29 ± 2.98	70.50 ± 3.13	71.73 ± 2.74
	#Feats	42.3 ± 20.9	51.7 ± 25.9	123.3 ± 51.9
	Time	113.7 ± 248.2	137.8 ± 147.5	388.2 ± 533.4

◦, ◦ statistically significant improvement, or degradation



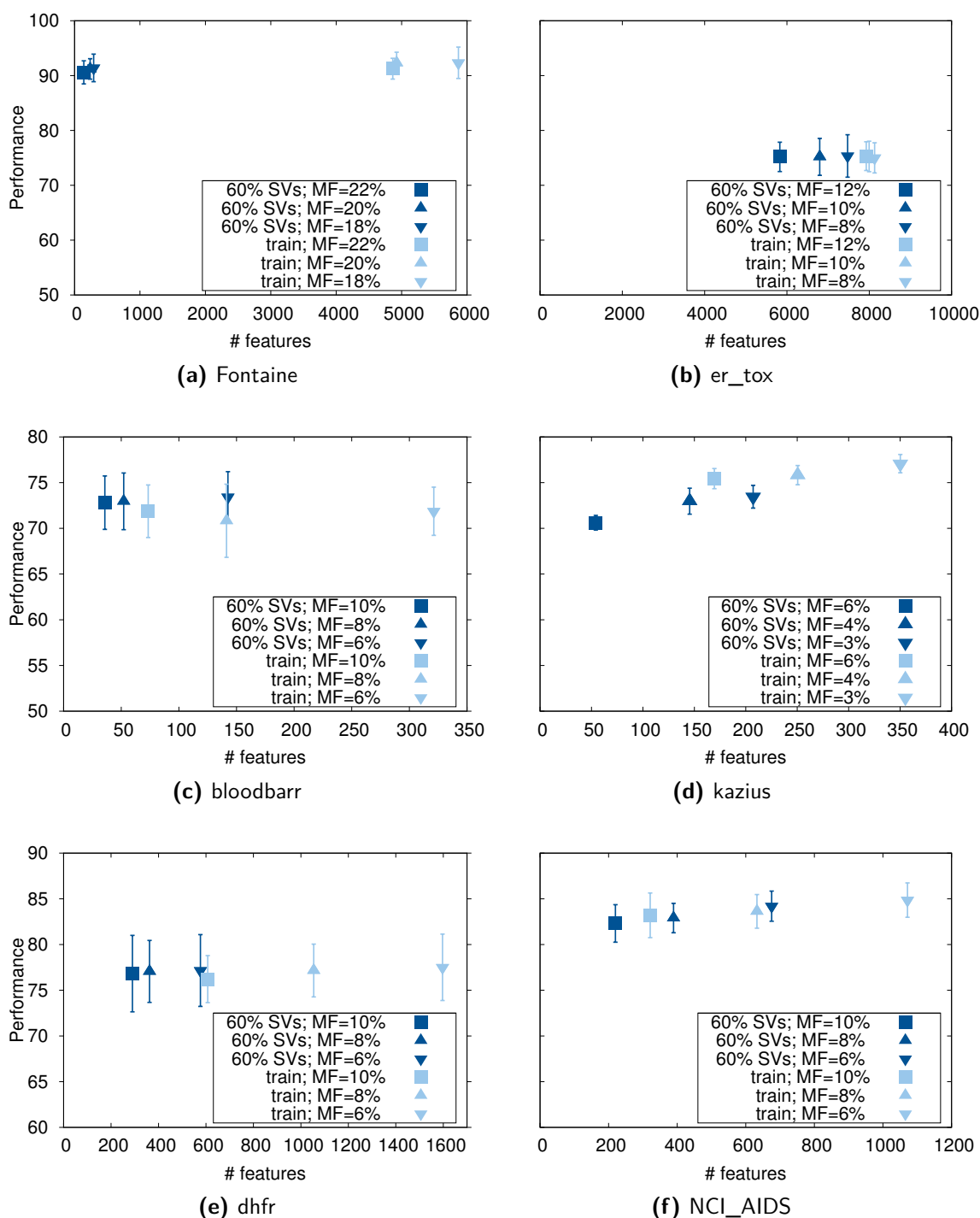
Table 6.3: (continued)

<b>kazius</b>		MF=6%	MF=4%	MF=3%
$D_{Trg}$	ACC	75.44 ± 1.10	75.82 ± 1.05	77.08 ± 1.00
	#Feats	169.5 ± 9.7	250.5 ± 15.4	350.2 ± 33.8
	Time	3122.5 ± 1456.9	4572.9 ± 1053.0	7016.5 ± 1155.50
SV <sub>60%</sub>	ACC	70.62 ± 0.80 ○	72.97 ± 1.42 ○	73.46 ± 1.25 ○
	#Feats	54.2 ± 12.1	145.5 ± 21.9	207.1 ± 42.7
	Time	70.3 ± 26.4	154.9 ± 64.5	227.4 ± 91.8
SV <sub>50%</sub>	ACC	67.56 ± 2.43 ○	67.85 ± 3.36 ○	68.78 ± 2.49 ○
	#Feats	29.3 ± 8.9	54.8 ± 15.6	95.0 ± 37.9
	Time	21.5 ± 13.5	40.5 ± 23.9	71.0 ± 42.3
FS <sub>top-k</sub>	ACC	74.17 ± 1.00 ●	75.07 ± 1.23 ●	76.31 ± 0.86 ●
	#Feats	54.2 ± 12.1	145.5 ± 21.9	207.1 ± 42.7
	Time	6443.9 ± 5036.9	6870.5 ± 12745.4	14107.0 ± 29832.0
FS	ACC	75.05 ± 0.84 ●	76.04 ± 0.93 ●	76.73 ± 0.73 ●
	#Feats	151.0 ± 10.2	165.2 ± 19.6	265.3 ± 68.59
	Time	12266.9 ± 8266.9	19819.4 ± 15778.5	32730.9 ± 28721.1
<b>dhfr</b>		MF=10%	MF=8%	MF=6%
$D_{Trg}$	ACC	76.22 ± 2.57	77.16 ± 2.89	77.51 ± 3.63
	#Feats	607.4 ± 161.4	1054.1 ± 280.5	1596.9 ± 366.0
	Time	20.7 ± 7.1	38.5 ± 60.7	41.8 ± 41.1
SV <sub>60%</sub>	ACC	76.82 ± 4.18	77.06 ± 3.39	77.16 ± 3.93
	#Feats	290.0 ± 152.4	362.0 ± 210.0	576.5 ± 315.1
	Time	3.7 ± 3.8	4.8 ± 4.4	5.7 ± 4.6
SV <sub>50%</sub>	ACC	75.97 ± 1.70	76.12 ± 3.08	76.31 ± 4.06
	#Feats	135.6 ± 85.3	213.5 ± 250.3	497.0 ± 377.0
	Time	2.5 ± 4.1	3.2 ± 4.5	3.0 ± 1.6
FS <sub>top-k</sub>	ACC	74.73 ± 4.28 ○	76.02 ± 3.47	76.61 ± 3.43
	#Feats	290.0 ± 152.4	362.0 ± 210.0	576.5 ± 315.1
	Time	88.9 ± 91.2	144.4 ± 140.6	303.6 ± 322.6
FS	ACC	74.98 ± 4.40	76.57 ± 3.45	76.96 ± 4.14
	#Feats	127.7 ± 86.9	163.7 ± 55.3	288.3 ± 377.5
	Time	876.0 ± 894.7	1451.4 ± 1544.1	3153.1 ± 3468.6
<b>NCI_AIDS</b>		MF=10%	MF=8%	MF=6%
$D_{Trg}$	ACC	83.20 ± 2.44	83.63 ± 1.84	84.86 ± 1.88
	#Feats	321.1 ± 83.3	632.6 ± 142.0	1071.6 ± 224.9
	Time	399.0 ± 113.0	471.7 ± 145.7	210.8 ± 95.9
SV <sub>60%</sub>	ACC	82.31 ± 2.05	82.90 ± 1.61	84.20 ± 1.65
	#Feats	219.9 ± 72.4	389.4 ± 107.8	675.5 ± 222.4
	Time	15.9 ± 6.4	20.7 ± 5.8	19.4 ± 3.2
SV <sub>50%</sub>	ACC	77.94 ± 2.77 ○	78.35 ± 1.44 ○	79.22 ± 1.86 ○
	#Feats	72.2 ± 22.2	96.8 ± 37.6	154.3 ± 79.9
	Time	6.5 ± 5.1	7.3 ± 5.2	8.6 ± 5.6
FS <sub>top-k</sub>	ACC	82.13 ± 1.94	82.42 ± 1.85	83.90 ± 1.83
	#Feats	219.9 ± 72.4	389.4 ± 107.8	675.5 ± 222.4
	Time	602.8 ± 1167.9	1010.1 ± 2155.3	1450.0 ± 2416.3
FS	ACC	82.26 ± 2.03	82.95 ± 1.89	84.25 ± 1.66
	#Feats	137.0 ± 61.1	238.1 ± 81.4	437.7 ± 98.9
	Time	4559.9 ± 12782.3	4408.7 ± 4050.0	13997.9 ± 13719.4

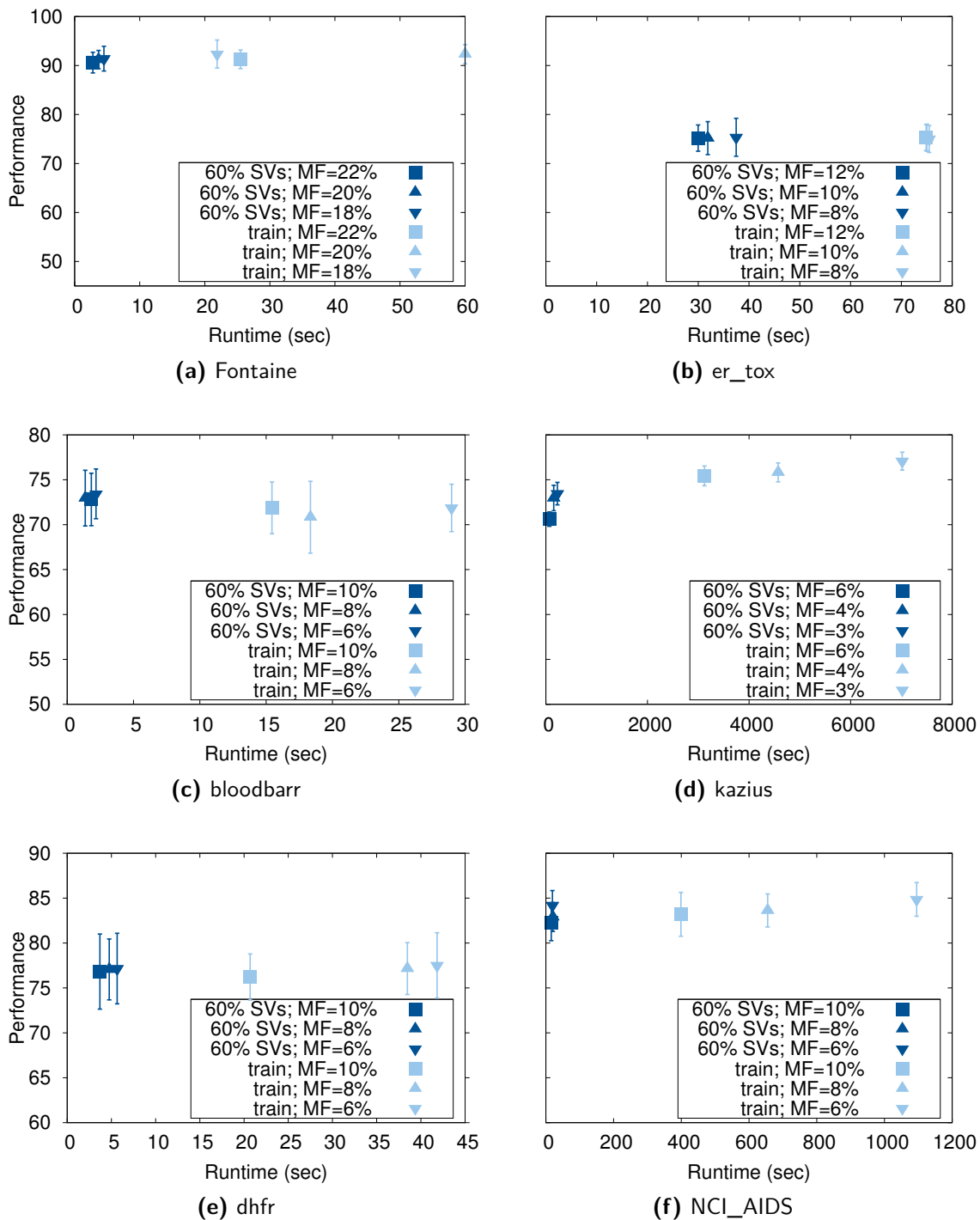
●,○ statistically significant improvement, or degradation

illustration, Figure 6.3 shows the relationship between the number of features and the prediction performance on all data sets for the method with the number of support vectors covering 60% of the training data and the method that employs all training data.

Comparing the proposed approach using 60% support vectors to both feature selection approaches,  $FS_{top-k}$  and FS, it can be observed that on all data sets except for *kazius* the proposed approach yields similar or even better predictive performance. At the same time, it approach requires far less time for training and testing than  $FS_{top-k}$ , while employing the same number of features. Even though the approach named FS produces less features than the proposed approach (except for *kazius*), it needs far more time for model building and prediction (up to factors of thousands). To summarize, the experimental results demonstrate that the classification models resulting from the proposed approach are on most data sets, more specifically on data sets comprising structurally more homogeneous graphs, not significantly less accurate than the models trained on the full data sets.



**Figure 6.3:** Relationship between the number of patterns (features) generated by BBRC and the predictive performance of the compared methods on all benchmark data sets. In each figure "train" denotes the method using the full data set as input for pattern mining, whereas "60% SVs" represents the approach incorporating only the support vectors and their corresponding weights into the mining process.



**Figure 6.4:** Relationship between the runtime and the predictive performance of the compared methods on all benchmark data sets. In each figure "train" denotes the method using the full data set as input for pattern mining, whereas "60% SVs" represents the approach incorporating only the support vectors and their corresponding weights into the mining process.

## 6.5 Conclusion

In the work presented here, an approach for extracting information from support vector machines for pattern-based classification was proposed. More specifically, the approach extracts information from trained support vector machines, in particular their support vectors and their relevance according to their coefficients. It uses the support vectors along with their coefficients as input to pattern mining algorithms able to handle weighted instances. The approach was evaluated on several real-world data sets of molecular graphs and compared it against a method that builds a classification model on a set of subgraph patterns derived from the full data sets. The results show that the models resulting from the proposed approach are on most data sets, i.e., on data sets containing structurally similar graphs, not significantly less accurate than models trained on the full data sets. At the same time the resulting models tend to be better interpretable, due to the smaller set of patterns generated by the mining process. Compared to using the full data set in the mining process, the proposed approach allows an analysis, which is up to an order of magnitude faster using often a substantially smaller number of features. This enables feasible parameter and feature optimization for given problems using tens of thousands of experiments in a fraction of the original time required.



# CHAPTER 7

## Conclusion and Outlook

This thesis covers scalable methods for clustering large graph databases as well as applications for graph classification and regression. First, three algorithms for clustering large databases of small graphs were introduced: i) structural graph clustering, ii) PSCG and iii) SCAP. These three methods cluster graph instances by common scaffolds, i.e., the existence of one sufficiently large subgraph shared by all cluster elements. Next, two methods exploiting local structural graph similarity neighborhoods for building regression and classification models were presented. The first method employs locally weighted learning, a form of lazy learning, in combination with a distance measure based on the maximum common subgraph. The second method exploits the clusters produced by PSCG to define a new graph kernel. Finally, an approach was presented that extracts knowledge from SVMs trained on the SCK. Figure 7.1 illustrates the structure of this thesis.

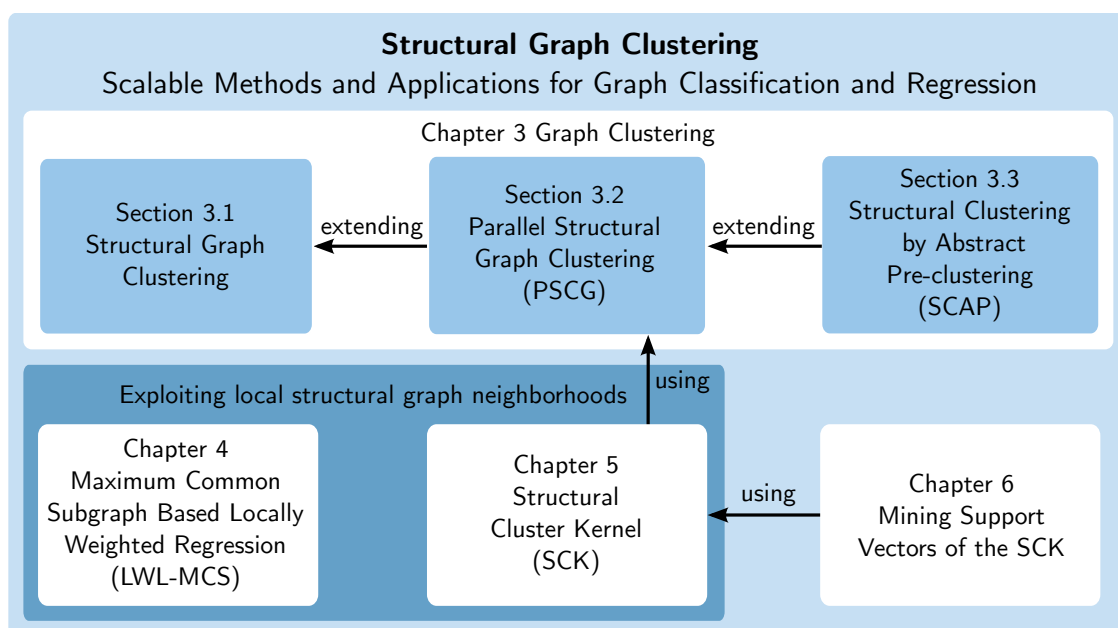


Figure 7.1: Overview of the contributions of the thesis.

## 7.1 Conclusion

Chapter 1 gave an introduction to the field of graph clustering, which is the task of grouping a set of graphs into clusters based on structural similarity. Graph clustering is an interesting and challenging research problem which has received much attention recently [4, 79, 181]. The identification of clusters in graph data is useful in many applications involving bio- and cheminformatics and the web. To motivate the presented work, several application areas with a focus on the domain of cheminformatics were introduced followed by an outline of the thesis.

Chapter 2 introduced the basics and notations for graph theory used throughout the thesis. Subsequently, a review on the topic of graph clustering and an overview of state-of-the-art algorithms for clustering vectorial data and graph data was provided. Next, the chapter gave an introduction to kernels and kernel methods with a particular focus on graph kernels. The last part of the chapter introduced the topic of knowledge extraction from SVMs and presented related work on this topic.

The main part of the thesis was presented in Chapter 3, where three new approaches for graph clustering were introduced with a special focus on scalability. All three approaches have in common that they consider clustering graph databases according to structural similarity. The first approach introduced the problem formulation of structural graph clustering providing the basics for the following two graph clustering approaches. The presented clustering approach works online and produces overlapping (non-disjoint) and non-exhaustive clusters. The resulting clusters encompass all graphs that share a sufficiently large common subgraph. The size of this common subgraph has to take at least a user-specified fraction of the size of each graph. Three factors contribute to the practically favorable performance of the approach: First, the use of a gSpan variant to compute a sufficiently large common subgraph, which is known to be effective on graphs of low density. Second, the possibility to terminate search as soon as such a subgraph is found. Third, the cluster exclusion criteria to avoid unnecessary subgraph mining runs. The experimental results on various real-world data sets of molecular graphs demonstrate that the proposed approach is able to rediscover known structure classes in data sets. Compared to fingerprint-based clustering, this approach yields larger and more representative cluster scaffolds contributing to an increase in intra-cluster homogeneity. Due to the introduction of several cluster exclusion criteria, the efficiency of the approach could further be improved resulting in a scalable, overlapping, non-exhaustive structural clustering approach that generates interpretable clusters in acceptable time.

The second part of Chapter 3 introduced a parallel version of the previously proposed structural graph clustering algorithm referred to as PSCG. Parallelization is an efficient technique to design fast algorithms for clustering huge data sets. Such algorithms are highly desirable due to the explosive growth of graph-structured data. In the domain of cheminformatics, for instance, graph databases representing chemical compounds routinely encompass several hundred thousands or even millions of graphs; thus, clustering methods



that are able to explore and structure the vast graph space are highly required. The parallelized graph clustering approach PSCG presented in Section 3.2 divides the cluster tasks among processors with minimal communication costs, thus making it distributable across a large number of computing nodes. The implementation of PSCG adopts the master-worker agenda paradigm for parallelization. To reduce the number of cluster membership tests, PSCG employs two clustering exclusion criteria, a set abstraction of graphs and a size-based clustering criterion. Further, to reduce subgraph mining running times for larger clusters, a cluster representative is defined for each cluster composed of the common cluster scaffold once this scaffold is unique. These optimizations together enable PSCG to achieve near-linear scalability and near-linear speed-up with the number of processors. Compared to previously proposed structure-based clustering algorithms, the algorithm is able to handle a much greater number of graph instances. In addition to obtaining time gain, the quality of the clusters is preserved in the parallel version. Given these performance improvements, PSCG is already applicable to the large structure databases from virtual screening. Further, PSCG is the first solution for the structural (i.e., scaffold-based) graph clustering problem that is able to handle data sets encompassing several hundred thousand graphs. The majority of previously proposed structural graph-based clustering algorithms, involving, e.g., the computation of the MCS, is hardly suitable for such data sets. The graph data sets covered in related papers typically contain only several hundred graphs [3, 107, 182], and hardly any effort has been spent on characterizing the performance of the clustering algorithms.

Due to the rapid and constant growth of graph databases the demand is increasing for clustering algorithms that are able to structure even larger libraries containing millions of graphs. An important requirement in this endeavor is scalability, as it allows us to cope with the large real-world compound libraries rapidly increasing in size. For this purpose, the third part of Chapter 3 introduced a scalable scaffold-based graph clustering approach that was designed for the purpose of clustering such massive graph data sets. The approach referred to as SCAP employs two clustering stages. First, a pre-clustering based on dynamic seed clustering is employed to partition the data set into several smaller data sets using an abstraction-based similarity measure. Next, the resulting partitions are clustered into a finer level of granularity using a scaffold-based structural graph clustering approach that produces overlapping (non-disjoint) and non-exhaustive clusters. More precisely, a modified version of PSCG is used that avoids cluster comparisons with all cluster members, which grow computationally more expensive with increasing cluster size, by defining a cluster representative for each cluster at an early clustering stage. To summarize, the main contribution of Section 3.3 is scalability, which is achieved by employing a pre-clustering step and by leveraging an incremental clustering algorithm that runs efficiently on very large data sets. Compared to previously proposed structure-based graph clustering algorithms, SCAP is able to handle a much larger number of graph instances, i.e., an order of magnitude more instances in a fraction of time required before. Further, SCAP has been shown to rediscover known structure classes in graph data sets. Given

these performance improvements, SCAP is applicable to the large structure databases. With this approach, it is for the first time possible to cluster millions of graphs within a reasonable time using an accurate scaffold-based similarity measure. There are many potential areas which could benefit from the use of the proposed graph clustering approach. In the area of cheminformatics, for instance, this represents an important step towards structuring the chemical space which could be defined by the structures of PubChem [176] currently containing approximately 49.5 million compounds.

Local structural graph similarity neighborhoods obtained, for instance, by the previously proposed graph clustering approaches, can be useful for a variety of purposes, for instance for building models for classification and regression. Chapters 4 and 5 presented two approaches that exploit local structural graph similarity neighborhoods for model building. The former of the two approaches presented in Chapter 4 investigates a simple, yet effective method for regression on graphs, in particular for applications in cheminformatics and for QSARs. The approach called LWL-MCS presents a variant of locally weighted regression on graphs that uses the maximum common subgraph for determining and weighting the neighborhood of a graph and feature vectors for the actual regression model. LWL-MCS belongs to the field of lazy learning, meaning that the structural neighborhood of a test instance is determined individually, on demand, at testing time. The presented combination outperforms other methods in the evaluation that use the local neighborhood of graphs for regression and works particularly well for the employed application area of QSARs. In this domain, the MCS retrieves structures with a large common structural scaffold, for which differences in the activities can be explained by differences in the physicochemical properties. Hence, the approach performs particularly well on structurally more heterogeneous graph data sets that really benefit from the combination of structural and chemical descriptors. On these data sets, using chemical descriptors alone while ignoring explicit structure information, the approach retrieves a set of structurally more diverse molecules. This leads to inferior predictions and makes interpretation harder for the expert chemist. On the other hand, using molecular graphs by themselves and ignoring chemical descriptors, the approach retrieves similar structures, but may overlook important binding site effects [160]. LWL-MCS uses a combination of both chemical and structure descriptors. Whereas the MCS is better suited to measure structural similarity between molecules, chemical descriptors are better suited to discriminate between structurally similar molecules with regard to their actual biological activities. Compared to local model learning, locally weighted learning is more time-consuming, but also potentially worth the effort, because it is able to adjust its predictions to the specifics of test instances. Though the approach works particularly well for the employed application area of QSARs, the performance of this method on graphs suggests that the approach might be useful for other types of structured data as well.

Chapter 5 introduced an approach that exploits structural graph neighborhoods to define a new kernel. More specifically, the novel kernel, called SCK, incorporates similarities induced by a structural graph clustering algorithm to improve state-of-the-art graph

kernels. The approach is motivated by the idea that graph similarity can not only be described by the similarity between the graphs themselves, but also by the similarity they possess with respect to their structural neighborhood. The SCK was investigated using both NSPDK and WDK as base kernel. For the clustering task, PSCG is used. In the evaluation, the novel approach was applied in a supervised and a semi-supervised setting to regression and classification problems on a number of real-world data sets of molecular graphs using SVMs. The experimental results indicate that the structural cluster similarity information can leverage the prediction performance of the base kernel, particularly when the data set is structurally sparse and consequently structurally diverse. The performance of the SCK approach was further investigated in the semi-supervised setting enriching the relatively small labeled graph data sets by a large set of unlabeled graph instances. The assumption is that if unlabeled data is added to the relatively small labeled data set, the new similarity, obtained via structural clustering and the use of unlabeled data, induces a better representational space for classification and regression than using only the labeled data. Compared to the supervised setting, the SCK in the semi-supervised setting yields performance gains, in particular for classification. The presented kernel-based approach is general as such, and can also be employed in conjunction with a variety of different kernels and clustering approaches and is therefore not restricted to graph mining alone.

Chapter 6 addressed the question whether incomprehensible trained SVM models can be used to obtain more compact pattern-based classification models. To do so, the proposed approach extracts information from the support vector machines trained on the SCK, in particular their support vectors and their relevance according to their coefficients. It uses the support vectors along with their coefficients as input to pattern mining algorithms able to handle weighted instances. The experiments in the domain of graph mining and molecular graphs show that the resulting models are not significantly less accurate than models trained on the full data sets, yet require only a fraction of the time using much smaller sets of patterns. The approach performs particularly well on data sets containing structurally similar graphs, i.e., on structurally homogeneous data sets.

## 7.2 Outlook

This thesis tackles the challenges in clustering large databases of graphs and applications for graph classification and regression. As the field of graph clustering is an interesting and challenging research problem that has received much attention recently, there exist many ways where future research could proceed.

The investigation of graph clustering in Chapter 3 is probably the part of the thesis which gives rise to the most opportunities for further research. It would be interesting and useful to suitably extend graph clustering to take into account multiple “views” on the data simultaneously to produce a more accurate and robust partitioning of the data. In many application domains of machine learning and data mining, such as bioinformatics, information retrieval, and social network analysis, it is natural to assume that there are

multiple views on the same data. Views are typically defined as sets of features or variables that together describe one aspect of the objects of interest. In many cases, it can be shown that working just with the union of variables (instead of defined views) decreases the performance on the task at hand (e.g., clustering, classification and regression). The increasing prevalence of multi-view data has given rise to the development of algorithms that employ multiple views simultaneously. Despite the interest in multi-view learning in general and multi-view clustering in particular, multi-view clustering in the domain of graphs has not received much attention so far. However, methods for multi-view graph clustering are currently called for in several application domains of data mining. For instance, there is a pressing need for multi-view clustering in many areas of cheminformatics, QSAR and predictive toxicology, where multiple endpoints are tested routinely and the task is not only prediction, but also the formation of *categories* homogeneous in both the structure and the biological profiles. Such categories are urgently required, for instance, in the context of REACH [186], the European Union regulation for the registration, evaluation, authorization and restriction of chemicals. Technically speaking, chemical structures can be represented by their graph structure, by a vector representing chemical fingerprints, by a vector representing various standard chemical descriptors (e.g., molecular weight, LogP, topological diameter and polar surface area) or by a binary vector indicating the biological activity against certain targets. Although these individual views might be sufficient on their own for a given learning task, they may often provide useful complementary information to each other which can lead to improved performance on the learning task at hand. The central axiom of QSAR is that the activity of molecules is reflected in their structure, i.e., structurally similar chemical structures should also have similar biological activity. Hence, by exploiting information from both the structural view as well as the biological view in the clustering process, one may expect to find a set of structurally homogeneous clusters reflecting similar biological or toxicological profiles. Though adding more views helps on average, adding a noisy view to a set of informative views might hurt the clustering accuracy for certain cases. Hence, another future research direction is to investigate how to carefully select the most informative views of a graph while downgrading the noisy ones. There are many more possible future research directions in the domain of graph clustering. From an application point of view, further work, e.g., in the domain of cheminformatics, could also investigate the effects of preprocessing steps, e.g., downweighting longer chains (acyclic substructures) or reduced graph representations (transforming cycles, in chemical terms: rings, into special nodes).

The LWL-MCS approach presented in Chapter 4 also offers a wide range of potential extensions. Due to the good performance of this conceptually simple learning scheme compared to other methods using the local neighborhood of graphs, it appears worthwhile to study similar approaches on other types of structured data (like sequence, tree, logical or database representations). This could be similarly successful whenever such structural similarity (longest common subsequence, maximum common subtree, least general generalization) can be complemented by feature vectors with orthogonal information. Further,

it might be worthwhile to study the approach in the context of classification.

Promising directions for further research can also be found in the structural cluster kernel approach presented in Chapter 5. The empirical evaluation was limited to a specific clustering approach, i.e., PSCG and to two base kernel, i.e., the WDK and the NSPDK. However, as the SCK approach is general as such and not restricted to a special base kernel or clustering method, it may also be employed in conjunction with a variety of different kernels and clustering approaches which might work equally well or even better. Further, the approach is not restricted to graph mining alone, but may be useful in many other domains.

Chapter 6 addressed the question whether one can make use of trained SVM models to obtain more compact pattern-based classification models. As stated in the chapter, there are at least two possible ways of doing so: by analyzing a trained SVM model together with the training set, or by using the models as oracles to label instances [62]. In this thesis, only the former of the two approaches was studied. The second approach, however, may be a promising future direction. More precisely, trained SVM models may be used as an oracle to label or classify a set of unlabeled examples which are then added to the labeled data. The black box SVM model is thus used as an oracle to answer class membership queries about unlabeled data points. A new model is then learned and the process iterated. The idea behind this technique is the assumption that the trained model can better represent the data than the original data set. That is, the data is cleaner and free of apparent conflicts. It is expected that this process improves the SVM classifier's performance. Further, the newly created data set may provide more useful information for pattern-based classification. The problem is also referred to as *active learning* in the literature [53]. Active learning is a technique originally designed for learning tasks in which training data is scarcely available. The approach can be further extended by selecting the most informative samples for labeling by the oracle, so as to reduce the classification error. Different heuristics may be considered to approximate the "informativeness" measure, such as uncertainty and diversity [220, 146].

Given the research presented in the previous chapters and the manifold directions for future work, I believe that graph clustering approaches as presented in this thesis have a role to play in future and upcoming research challenges in many application areas such as bio- and cheminformatics.



---

## List of Figures

---

1.1	Overview of the contributions of the thesis. . . . .	7
1.2	A 2D graph representation of a molecular compound (1-Phenylethanol). . . . .	8
2.1	Examples for directed, undirected and labeled graphs. Left: Undirected graph. Center: Directed graph. Right: Labeled undirected graph. . . . .	12
2.2	Examples for graph isomorphism and subgraph isomorphism. The centered graph is isomorphic to the left graph, and the right graph is isomorphic to a subgraph of the left graph. The node labels are indicated by different letters. All edge are assumed to have identical edge labels. . . . .	13
2.3	Maximum common subgraph example. The right graph is a maximum common subgraph of the left graph and centered graph. . . . .	14
2.4	Example of a graph represented by a binary feature vector indicating the presence or absence of subgraph occurrences. . . . .	16
2.5	Fragment spectrum of a graph adapted from Yoshida <i>et al.</i> [250]. . . . .	18
2.6	Comparing the explicit mapping of patterns $x$ and $x'$ in a feature space $\mathcal{H}$ via $\varphi$ and subsequent dot product computation with the shortcut kernel-trick. Note that the pattern space $X$ can be any domain (e.g. the domain of graphs $G$ , or a vector space $\mathcal{H}$ ). . . . .	26
2.7	Example of selector (light blue vertex) and context (dark blue vertices and light blue vertex) for a graph. Adapted from Menchetti [165]. . . . .	29
2.8	Example of a linear classifier separating two classes (filled dots and unfilled dots). The decision surface (in dark blue) is a hyperplane defined by $\langle w, x_i \rangle + b = 0$ . The margin (dashed line) is defined by the distance of the closest points ( $x_1$ and $x_2$ ). Data points located on the margin are support vectors. . . . .	32
2.9	Illustration of the slack variables $\xi_i$ , for $i = 1 \dots n$ . Note that only the values $\xi_i \neq 0$ are shown, corresponding to points on the wrong side of the margin. All the other points lying either on the margin or on the correct side have $\xi_i = 0$ . . . . .	33
2.10	Toy example illustrating the kernel trick. Mapping a circle into feature space: data distribution in input space (left) and feature space (right). By transformation from input space $X$ to feature space $\mathcal{H}$ by function $\varphi$ , the light blue and dark blue dots become linearly separable. . . . .	36

---

2.11	The translucency criterion for categorizing techniques for extracting knowledge from trained SVMs. . . . .	40
2.12	Example illustrating the approach by Núñez <i>et al.</i> [175]. . . . .	41
2.13	Example illustrating the approach by Fung <i>et al.</i> [84]. The non-overlapping rules covering the half-space are represented as rectangles. . . . .	42
2.14	Pedagogical rule extraction approach by Barakat and Diederich [16]. Adapted from [18]. . . . .	44
2.15	Illustration of the pre-image problem in kernel-based machines. Adapted from Honeine and Richard [105]. . . . .	47
3.1	(Above) Results of gSpan: Runtime behavior (left) and number of subgraphs (right) on COX2 and CPDB. (Below) Results of structural clustering on COX2 (left) and CPDB (right). . . . .	51
3.2	Sample output from structural clustering. Atoms correspond to labeled vertices, bonds to edges. Vertices without atom labels represent carbon (C) atoms. Only heavy atoms are considered, i.e., hydrogen atoms (H) are ignored. The figure distinguishes two bond types: Single bonds and double bonds. . . . .	52
3.3	Schematic overview of the cluster membership assignment for instance $x_i$ . Graph instances are represented by $x_1, \dots, x_n$ , clusters by $C_1, \dots, C_k$ . . . . .	53
3.4	Example illustrating the cluster assignment step of the proposed structural clustering approach for $\theta = 0.6$ . The figure shows the clustering state at different time steps. To be assigned to the cluster, query instance $x_3$ needs to share at least one common subgraph with the cluster members $x_1$ and $x_2$ that meets the <i>minSize</i> threshold defined in Equation 3.2. As $x_3$ shares such a subgraph with $x_1$ and $x_2$ that meets the <i>minSize</i> threshold of 5.4, $x_3$ is assigned to the cluster. . . . .	54
3.5	Results of structural clustering (a), (c), (e) vs. fingerprint clustering (b), (d), (f) on CPD MOUSE, CPD RAT and EPAFHM. . . . .	58
3.6	Histogram of the share of the MCS of the largest cluster instance for fingerprint clustering on (a) CPD MOUSE, (b) CPD RAT and (c) EPAFHM using a Tanimoto coefficient value of 0.6. . . . .	59
3.7	Results of (a) structural clustering for $\theta = 0.6$ and (b) DP Clustering for $\alpha = 0.1$ and $m = 1000$ on the SACA data set. The different symbols for the cluster instances represent the six SACA classes. . . . .	60
3.8	Runtime performance of the structure-based clustering approach with and without clustering exclusion criteria on CPD MOUSE (left), CPD RAT (middle) and EPAFHM (right). . . . .	62
3.9	Runtime performance of the structure-based clustering approach on ten data sets from the NCI anti-HIV database consisting of $x$ graphs ( $x \in [1000, 10000]$ ). . . . .	63
3.10	Flowchart of the master-worker paradigm employed by PSCG. . . . .	69
3.11	Flowchart of the worker computation. . . . .	69



3.12	Example sequence of steps of PSCG. . . . .	71
3.13	Example illustrating the use of the size-based cluster exclusion criterion on a data set of chemical compounds containing eight graphs ( $\theta = 0.5$ ). Left: Clustering at time $t_1$ . Only graphs of size $\leq \frac{1}{\theta} \cdot x_{min} = \frac{4}{0.5} = 8$ , i.e., graph $x_2$ , need to be considered for comparison against cluster 1. Right: Clustering at time $t_2$ . Only graphs of size $\leq \frac{1}{\theta} \cdot x_{min} = \frac{7}{0.5} = 14$ , i.e., graphs $x_3$ to $x_7$ , need to be considered for clustering against cluster 2. . . . .	73
3.14	Example use of the feature vector-based cluster exclusion criterion ( $\theta = 0.6$ ). Since the similarity between the feature vectors of query graph $x_3$ and cluster 1 is smaller than the minimum required size of the common subgraph, the common subgraph computation step can be omitted. . . . .	75
3.15	Example illustrating the definition of a cluster representative. At time $t_1$ , the cluster members $x_1$ and $x_2$ share two common subgraphs that meet the <i>minSize</i> threshold. Hence, there does not exist a unique cluster scaffold. At time $t_2$ , there exists a unique cluster scaffold which is used as cluster representative for further cluster assignments. . . . .	76
3.16	Execution time (left) and speedup (right) of PSCG on the first 10,000 graphs of the NCI anti-HIV data set. . . . .	77
3.17	Runtime reduction due to algorithm improvements. . . . .	78
3.18	Relative frequency of size-based and feature vector-based exclusion criterion and number of gSpan calls. . . . .	79
3.19	Example of the pending cluster conversion process using two neighboring clusters reaching the threshold size <i>max_pending</i> = 5 (left). For both clusters, the fixed seeds and the fixed cluster members are determined (center and right). After the conversion process, the former pending seed of the cluster can be shifted to another member of the original pending cluster (see arrows). . . . .	85
3.20	Example of the pending cluster conversion process using two neighboring clusters. The pending cluster $C_{p_1}$ reaches the threshold size <i>max_pending</i> = 3 (left). Hence, for this cluster, the fixed seed and the fixed cluster members are determined (center). If a graph that is assigned to the new fixed cluster is also member of a pending cluster, it has to be removed from the pending cluster (right). . . . .	85
3.21	APreClus workflow . . . . .	87
3.22	Example illustrating the use of the abstraction-based similarity measure employed by APreClu ( $\theta = 0.6$ ). Since the similarity between the feature vectors of query graph $x_3$ and the cluster representative <i>seed</i> $_{C_p}$ is smaller than the minimum required size of the common subgraph, $x_3$ won't be assigned to cluster $C_p$ . . . . .	88

---

3.23	Example illustrating the definition of a cluster representative (for $\theta = 0.7$ ). Left: Cluster at time $t_1$ . To be assigned to the cluster, the query graph $x_2$ needs to share at least one common subgraph with the cluster member $x_1$ that meets the <i>minSize</i> threshold defined in Equation 3.2. As $x_2$ shares such a subgraph with $x_1$ that meets the <i>minSize</i> threshold of 8, $x_2$ is assigned to the cluster. Right: Cluster at time $t_2$ . The minimum common subgraph is taken as cluster representative. In the following, all subsequent graphs are compared only against the cluster representative. . . . .	90
3.24	Runtime performance of the clustering approach SCAP on the data sets sampled from the ChemDB database comprising 100k to 3M graphs. . . . .	96
3.25	Runtime performance of BIRCH on the data sets sampled from ChemDB. . . . .	97
4.1	Sample similarity matrix calculation. The MCS between the graph structures $g_1$ and $g_2$ is marked blue. . . . .	104
4.2	Example of determining the $k$ nearest neighbors (here: $k = 6$ ) of a given test instance $x_q$ from a training set of graph instances. . . . .	106
4.3	Mean absolute errors and 95% confidence intervals for the comparison methods using 66% of the data for training. . . . .	114
4.4	Mean absolute errors and 95% confidence intervals for the comparison methods using 90% of the data for training. . . . .	114
4.5	Relation between the Tanimoto similarity and the difference in MAE between LWL-MCS and CS ( $f_D = 66\%$ ). . . . .	115
5.1	Illustration of the cluster kernel concept. The cluster-based similarity $K_{C_i}(x_1, x_2)$ between the highlighted structures $x_1$ and $x_2$ is computed based on the averaged pairwise similarities between the clusters they belong to. $x_1$ belongs to $C_1$ and $C_2$ , $x_2$ to $C_2$ and $C_3$ . Thus, the pairwise similarities between the cluster instances of cluster $C_1C_2$ , $C_1C_3$ , $C_2C_2$ (which equals 1) and $C_2C_3$ need to be computed. . . . .	121
5.2	Flowchart illustrating the construction of the structural cluster kernel in the semi-supervised setting. . . . .	122
5.3	Mean absolute errors with 95% confidence intervals on the different comparison methods for the regression data sets in Table 5.1. . . . .	129
5.4	Classification accuracies with 95% confidence intervals on the different comparison methods for the classification data sets in Table 5.1. . . . .	130
5.5	Mean absolute errors with 95% confidence intervals on the different comparison methods for the first ten regression data sets in Table 5.1. . . . .	131
6.1	Three example trees with the same backbone (bold). Its sequence is 'c:c-c-C=C-O-C' (reflecting that the fragments include part of an aromatic ring). It also holds that $q_1 \preceq_b q_2$ and $q_3 \preceq_b q_2$ , but neither $q_1 \preceq_b q_3$ nor $q_3 \preceq_b q_1$ . Therefore, $q_1$ and $q_3$ are not in the same Backbone Refinement Class. . . . .	137

---

6.2	Flowchart of the proposed approach for extracting information from support vector machines for pattern-based classification. . . . .	141
6.3	Relationship between the number of patterns (features) generated by BBRC and the predictive performance of the compared methods on all benchmark data sets. In each figure "train" denotes the method using the full data set as input for pattern mining, whereas "60% SVs" represents the approach incorporating only the support vectors and their corresponding weights into the mining process.	147
6.4	Relationship between the runtime and the predictive performance of the compared methods on all benchmark data sets. In each figure "train" denotes the method using the full data set as input for pattern mining, whereas "60% SVs" represents the approach incorporating only the support vectors and their corresponding weights into the mining process. . . . .	148
7.1	Overview of the contributions of the thesis. . . . .	151



---

## List of Tables

---

2.1	List of well-known positive definite kernels for vectorial data. . . . .	25
3.1	Overview of the data sets used for assessing the structural clustering method. .	56
3.2	Number of clusters and Rand index values for structural clustering on SACA. .	60
3.3	Number of clusters and size of the DFS code tree for DP clustering on the SACA data set with $\alpha = 0.1$ . . . . .	61
3.4	Runtime (in sec) of the sequential clustering version vs. PSCG on the first 10,000 graphs of the NCI anti-HIV data set for different values of $\theta$ . . . . .	79
3.5	Runtime (in sec) for the sampled data sets. . . . .	79
3.6	Number of clusters for the sampled data sets. . . . .	80
3.7	Clustering results for SCAP and PSCG. . . . .	93
3.8	Clustering results for the incremental FP clustering ( $\tau$ : Tanimoto similarity threshold). . . . .	95
3.9	Clustering results for BIRCH (t: distance threshold). . . . .	95
3.10	Clustering results for SCAP using $\theta = 0.7$ . . . . .	95
3.11	Clustering results for DP Clustering. . . . .	96
3.12	Runtime (in sec), number of #partitions produced by APreClus and number of clusters generated by SCAP on the three large data sets sampled from ChemDB. . . . .	98
4.1	Percentage of MCS computations that do not finish within either ten milliseconds or 1.28 seconds for each of the data sets used in the experiments reported later. . . . .	103
4.2	Overview of the data sets used for assessing the LWL-MCS method. $n$ denotes the number of molecules in the respective data set. . . . .	110
4.3	Mean absolute errors (MAE) with standard errors and the results of the Wilcoxon signed-rank test with a 95% confidence level between LWL-MCS and LWL-EUC, EKM-MCS, EKM-COMB, CS, EQ and GL respectively. Abbreviations: $f_D$ = fraction of data set used for training; Wil = Wilcoxon signed-rank test, W/L = wins/losses. . . . .	111
4.4	Mean runtime (training and testing time in sec) of one time hold-out validation with standard deviations for LWL-MCS and LWL-EUC, EKM-MCS, EKM-COMB, CS, EQ and GL. The abbreviations correspond to Table 4.3. . . . .	112

---

5.1	Overview of the data sets used for assessing the structural cluster kernel. $n$ denotes the number of molecular graphs in the respective data set. . . . .	122
5.2	Mean absolute errors with standard deviations of SCK NSPDK, NSPDK, LoMoGraph NSPDK and LoMoGraph on the regression data sets. Statistically significant results are reported using both the Wilcoxon signed-rank test and the corrected resampled t-test (separated by a ' '). . . . .	125
5.3	Mean absolute errors with standard deviations of SCK WDK, WDK, LoMoGraph WDK and LoMoGraph on the regression data sets. Statistically significant results are reported using both the Wilcoxon signed-rank test and the corrected resampled t-test (separated by a ' '). . . . .	125
5.4	Classification accuracies with standard deviations of SCK NSPDK, NSPDK, LoMoGraph NSPDK and LoMoGraph on the classification data sets. Statistically significant results are reported using both the Wilcoxon signed-rank test and the corrected resampled t-test (separated by a ' '). . . . .	126
5.5	Classification accuracies with standard deviations of SCK WDK, WDK, LoMoGraph WDK and LoMoGraph on the classification data sets. Statistically significant results are reported using both the Wilcoxon signed-rank test and the corrected resampled t-test (separated by a ' '). . . . .	126
5.6	Mean absolute errors with standard deviations of SCK NSPDK in both the semi-supervised and supervised setting and NSPDK on the regression data sets. Statistically significant results are reported using the Wilcoxon signed-rank test and the corrected resampled t-test (separated by a ' '). . . . .	128
5.7	Classification accuracies with standard deviations of SCK NSPDK in both the semi-supervised and supervised setting and NSPDK on the classification data sets. Statistically significant results are reported using the Wilcoxon signed-rank test and the corrected resampled t-test (separated by a ' '). . . . .	128
5.8	Mean absolute errors with standard deviations of SCK NSPDK in both the semi-supervised and supervised setting and LWL-MCS on the regression data sets. Statistically significant results are reported using the Wilcoxon signed-rank test and the corrected resampled t-test (separated by a ' '). . . . .	131
6.2	Overview of the data sets used for assessing the method. $n$ denotes the number of graph instances in the respective data set and <i>Tanimoto Sim.</i> describes the mean pair-wise Tanimoto similarity using ChemAxon's chemical fingerprint with default parameter setting. . . . .	142
6.3	Classification accuracies (ACC), number of attributes (#Feats) and runtime (in sec). MF denotes the minimum frequency parameter of BBRC. . . . .	144

---

## List of Algorithms

---

1	Structural Clustering . . . . .	55
2	PSCG: Master . . . . .	66
3	PSCG: Worker . . . . .	66
4	PSCG: Structural Clustering . . . . .	67
5	PSCG: Maintenance of cluster membership information . . . . .	68
6	SCAP . . . . .	82
7	APreClus . . . . .	83
8	APreClus: Reassign step . . . . .	84
9	PSCG' . . . . .	91
10	LWL-MCS . . . . .	107
11	Structural Cluster Kernel . . . . .	118





---

## Bibliography

---

- [1] ABOU-RJEILI, A., AND KARYPIS, G. Multilevel algorithms for partitioning power-law graphs. In *Proceedings of the 20th International Conference on Parallel and Distributed Processing* (2006), IPDPS 2006, IEEE Computer Society, pp. 124–124.
- [2] AGGARWAL, C. C., AND REDDY, C. K. *Data Clustering: Algorithms and Applications*. CRC Press, 2014.
- [3] AGGARWAL, C. C., TA, N., WANG, J., FENG, J., AND ZAKI, M. XProj: a framework for projected structural clustering of XML documents. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2007), KDD 2007, ACM, pp. 46–55.
- [4] AGGARWAL, C. C., AND WANG, H. *Managing and Mining Graph Data*, vol. 40 of *Advances in Database Systems*. Springer, 2010.
- [5] AGRAWAL, R., IMIELINSKI, T., AND SWAMI, A. Database mining: A performance perspective. *IEEE Transactions on Knowledge and Data Engineering* 5, 6 (Dec. 1993), 914–925.
- [6] AGRAWAL, R., AND SRIKANT, R. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases* (1994), VLDB 1994, Morgan Kaufmann Publishers Inc., pp. 487–499.
- [7] ALPHONSE, E., GIRSCHICK, T., BUCHWALD, F., AND KRAMER, S. A numerical refinement operator based on multi-instance learning. In *Proceedings of the 20th International Conference on Inductive Logic Programming* (2011), ILP 2010, Springer-Verlag, pp. 14–21.
- [8] AMBAUEN, R., FISCHER, S., AND BUNKE, H. Graph edit distance with node splitting and merging, and its application to diatom identification. In *Proceedings of the 4th IAPR International Conference on Graph Based Representations in Pattern Recognition* (2003), GbRPR 2003, Springer-Verlag, pp. 95–106.
- [9] ANDREWS, R., DIEDERICH, J., AND TICKLE, A. B. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems* 8, 6 (1995), 373–389.

- [10] ATKESON, C., MOORE, A., AND SCHAAL, S. Locally weighted learning. *Artificial Intelligence Review* 11, 1-5 (1997), 11–73.
- [11] BADER, D. A., AND MADDURI, K. A graph-theoretic analysis of the human protein-interaction network using multicore parallel algorithms. *Parallel Computing* 34, 11 (Nov. 2008), 627–639.
- [12] BAHLMANN, C., HAASDONK, B., AND BURKHARDT, H. On-line handwriting recognition with support vector machines - a kernel approach. In *Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (2002)*, IWFHR '02, IEEE Computer Society, pp. 49–.
- [13] BAKIR, G. H., WESTON, J., AND SCHÖLKOPF, B. Learning to find pre-images. In *Advances in Neural Information Processing Systems 16 (2003)*, NIPS 2003, MIT Press, pp. 449–456.
- [14] BAKIR, G. H., ZIEN, A., AND TSUDA, K. Learning to find graph pre-images. In *Pattern Recognition: Proceedings of the 26th DAGM Symposium (2004)*, DAGM 2004, Springer-Verlag, pp. 253–261.
- [15] BARAKAT, N., AND BRADLEY, A. P. Rule extraction from support vector machines: Measuring the explanation capability using the area under the ROC curve. In *Proceedings of the 18th International Conference on Pattern Recognition - Volume 02 (2006)*, ICPR 2006, IEEE Computer Society, pp. 812–815.
- [16] BARAKAT, N., AND DIEDERICH, J. Learning-based rule-extraction from support vector machines: Performance on benchmark data sets. In *Proceedings of the Conference on Neuro-Computing and Evolving Intelligence (2004)*, N. Kasabov and Z. S. H. Chan, Eds.
- [17] BARAKAT, N., AND DIEDERICH, J. Eclectic rule extraction from support vector machines. *International Journal of Computational Intelligence* 2 (2005), 59–62.
- [18] BARAKAT, N. H., AND BRADLEY, A. P. Rule extraction from support vector machines: A sequential covering approach. *IEEE Transactions on Knowledge and Data Engineering* 19, 6 (June 2007), 729–741.
- [19] BARBELLA, D., BENZAID, S., CHRISTENSEN, J., JACKSON, B., QIN, X. V., AND MUSICANT, D. R. Understanding support vector machine classifications via a recommender system-like approach. In *Proceedings of the 5th International Conference on Data Mining 2009 (2009)*, R. Stahlbock, S. F. Crone, and S. Lessmann, Eds., DMIN 2009, CSREA Press, pp. 305–311.
- [20] BEN-HUR, A., HORN, D., SIEGELMANN, H. T., AND VAPNIK, V. Support vector clustering. *Journal of Machine Learning Research* 2 (Mar. 2002), 125–137.

- 
- [21] BENIGNI, R., BOSSA, C., AND VARI, M. Chemical Carcinogens: Structures and Experimental Data, <http://www.iss.it/binary/ampp/cont/ISSCANv2aEn.1134647480.pdf>. Last accessed on 2014-04-27.
- [22] BENNETT, K. P., AND MANGASARIAN, O. L. Multicategory separation via linear programming. *Optimization Methods and Software* 3 (1993), 27–39.
- [23] BICKEL, S., AND SCHEFFER, T. Multi-view clustering. In *Proceedings of the Fourth IEEE International Conference on Data Mining* (2004), ICDM 2004, IEEE Computer Society, pp. 19–26.
- [24] BISHOP, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
- [25] BITSCH, A., JACOBI, S., MELBER, C., WAHNSCHAFFE, U., SIMETSKA, N., AND MANGELSDORF, I. REPDOSE: A database on repeated dose toxicity studies of commercial chemicals - a multifunctional tool. *Regulatory Toxicology and Pharmacology* 46, 3 (2006), 202–210.
- [26] BLEI, D. M., AND JORDAN, M. I. Variational inference for dirichlet process mixtures. *Bayesian Analysis* 1 (2005), 121–144.
- [27] BODO, Z. Hierarchical cluster kernels for supervised and semi-supervised learning. In *Proceedings of the 4th International Conference on Intelligent Computer Communication and Processing* (2008), ICCP 2008, IEEE, pp. 9–16.
- [28] BODO, Z., AND CSATO, L. Hierarchical and reweighting cluster kernels for semi-supervised learning. *International Journal of Computers Communications and Control* 5, 4 (2010), 469–476.
- [29] BOEHM, M. *Virtual Screening of Chemical Space: From Generic Compound Collections to Tailored Screening Libraries*. Wiley-VCH Verlag GmbH & Co. KGaA, 2011, pp. 1–33.
- [30] BORGELT, C., AND BERTHOLD, M. R. Mining molecular fragments: Finding relevant substructures of molecules. In *Proceedings of the 2002 IEEE International Conference on Data Mining* (2002), ICDM 2002, IEEE Computer Society, pp. 51–.
- [31] BORGWARDT, K. M. *Graph kernels*. PhD thesis, 2007.
- [32] BORGWARDT, K. M., AND KRIEGEL, H.-P. Shortest-path kernels on graphs. In *Proceedings of the Fifth IEEE International Conference on Data Mining* (2005), ICDM 2005, IEEE Computer Society, pp. 74–81.
- [33] BORGWARDT, K. M., ONG, C. S., SCHÖNAUER, S., VISHWANATHAN, S. V. N., SMOLA, A. J., AND KRIEGEL, H.-P. Protein function prediction via graph kernels. *Bioinformatics* 21, 1 (Jan. 2005), 47–56.

- [34] BOSER, B. E., GUYON, I. M., AND VAPNIK, V. N. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory* (1992), COLT 1992, ACM, pp. 144–152.
- [35] BOYD, M. R. *Cancer: Principles and Practice of Oncology*, vol. 3. DeVita, V. T., Jr. and Hellman, S. and Rosenberg, S. A., Eds, Lippincott: Philadelphia, PA, 1989, pp. 1–12.
- [36] BRADLEY, J. K., KYROLA, A., BICKSON, D., AND GUESTRIN, C. Parallel coordinate descent for  $l_1$ -regularized loss minimization. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011* (2011), ICML 2011, Omnipress, pp. 321–328.
- [37] BRODER, A., KUMAR, R., MAGHOUL, F., RAGHAVAN, P., RAJAGOPALAN, S., STATA, R., TOMKINS, A., AND WIENER, J. Graph structure in the web. In *Proceedings of the 9th International World Wide Web Conference on Computer Networks : The International Journal of Computer and Telecommunications Netowrking* (2000), North-Holland Publishing Co., pp. 309–320.
- [38] BUCHWALD, F., GIRSCHICK, T., SEELAND, M., AND KRAMER, S. Using local models to improve (Q)SAR predictivity. *Molecular Informatics* 30, 2-3 (2011), 205–218.
- [39] BUNKE, H., AND ALLERMANN, G. Inexact graph matching for structural pattern recognition. *Pattern Recognition Letters* 1, 4 (May 1983), 245–253.
- [40] BUNKE, H., FOGGIA, P., GUIDOBALDI, C., AND VENTO, M. Graph clustering using the weighted minimum common supergraph. In *Proceedings of the Fourth IAPR International Conference on Graph Based Representations in Pattern Recognition* (2003), GbRPR 2003, Springer-Verlag, pp. 235–246.
- [41] BUNKE, H., AND GÜNTER, S. Weighted mean of a pair of graphs. *Computing* 67, 3 (2001), 209–224.
- [42] BUNKE, H., MÜNGER, A., AND JIANG, X. Combinatorial search versus genetic algorithms: a case study based on the generalized median graph problem. *Pattern Recognition Letters* 20, 11-13 (1999), 1271–1277.
- [43] BURGES, C. J. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery* 2 (1998), 121–167.
- [44] BURGES, C. J. C. Simplified support vector decision rules. In *Proceedings of the 13th International Conference on Machine Learning* (1996), ICML 1996, Morgan Kaufmann, pp. 71–77.
- [45] CADASTER. Environmental Toxicity Prediction Challenge, <http://www.cadaster.eu/node/65>. Last accessed on 2014-04-27.

- 
- [46] CAO, Y., JIANG, T., AND GIRKE, T. A maximum common substructure-based algorithm for searching and predicting drug-like compounds. *Bioinformatics* 24, 13 (2008), i366–i374.
- [47] CHAPELLE, O., WESTON, J., AND SCHÖLKOPF, B. Cluster kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems* (2003), vol. 15 of *NIPS 2002*, MIT Press, pp. 585–592.
- [48] CHEN, J., SWAMIDASS, S. J., DOU, Y., AND BALDI, P. ChemDB: a public database of small molecules and related chemoinformatics resources. *Bioinformatics* 21 (2005), 4133–4139.
- [49] CHEN, J. H., LINSTED, E., SWAMIDASS, S. J., WANG, D., AND BALDI, P. ChemDB update – full-text search and virtual chemical space. *Bioinformatics* 23 (2007), 2348–2351.
- [50] CHEN, M.-S., HAN, J., AND YU, P. S. Data mining: An overview from a database perspective. *IEEE Transactions on Knowledge and Data Engineering* 8, 6 (Dec. 1996), 866–883.
- [51] CHEN, Z., LI, J., AND WEI, L. A multiple kernel support vector machine scheme for feature selection and rule extraction from gene expression data of cancer tissue. *Artificial Intelligence in Medicine* 41, 2 (2007), 161–175.
- [52] CLEVELAND, W. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association* 74 (1979), 829–836.
- [53] COHN, D., ATLAS, L., AND LADNER, R. Improving generalization with active learning. *Machine Learning* 15, 2 (1994), 201–221.
- [54] COLLINS, J. M. The DTP AIDS Antiviral Screen Program 1999, <http://dtp.nci.nih.gov/docs/aids/aidsdata.html>. Last accessed on 2014-04-27.
- [55] COLLINS, L. M., AND DENT, C. W. Omega: A general formulation of the rand index of cluster recovery suitable for non-disjoint solutions. *Multivariate Behavioral Research* 23 (1988), 231–242.
- [56] COMANICIU, D., AND MEER, P. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24, 5 (May 2002), 603–619.
- [57] CONTE, D., FOGGIA, P., SANSONE, C., AND VENTO, M. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence* 18, 3 (2004), 265–298.

- [58] COOK, D. J., AND HOLDER, L. B. Substructure discovery using minimum description length and background knowledge. *Journal of Artificial Intelligence Research* 1, 1 (Feb. 1994), 231–255.
- [59] COOK, D. J., AND HOLDER, L. B. *Mining Graph Data*. John Wiley & Sons, 2006.
- [60] CORTES, C., AND VAPNIK, V. Support vector networks. In *Machine Learning* (1995), vol. 20, pp. 273–297.
- [61] COSTA, F., AND DE GRAVE, K. Fast neighborhood subgraph pairwise distance kernel. In *Proceedings of the 26th International Conference on Machine Learning* (2010), ICML 2010, Omnipress, pp. 255–262.
- [62] CRAVEN, M., AND SHAVLIK, J. W. Using sampling and queries to extract rules from trained neural networks. In *Proceedings of the Eleventh International Conference on Machine Learning* (1994), ICML 1994, Morgan Kaufmann, pp. 37–45.
- [63] CRAVEN, M., AND SHAVLIK, J. W. Extracting tree-structured representations of trained networks. In *Advances in Neural Information Processing Systems 8* (1995), NIPS 1995, pp. 24–30.
- [64] CRAVEN, M. W. *Extracting comprehensible models from trained neural networks*. PhD thesis, 1996.
- [65] CRISTIANINI, N., AND SHAWE-TAYLOR, J. *An introduction to support Vector Machines and other kernel-based learning methods*. Cambridge University Press, New York, NY, USA, 2000.
- [66] DA COSTA F. CHAVES, A., VELLASCO, M. B. R., AND TANSCHKEIT, R. Fuzzy rule extraction from support vector machines. In *5th International Conference on Hybrid Intelligent Systems* (2005), HIS 2005, IEEE Computer Society, pp. 335–340.
- [67] DALAMAGAS, T., CHENG, T., WINKEL, K.-J., AND SELLIS, T. Clustering XML documents using structural summaries. In *Proceedings of the 2004 International Conference on Current Trends in Database Technology* (2004), EDBT 2004, Springer-Verlag, pp. 547–556.
- [68] DE MAURO, C., DILIGENTI, M., GORI, M., AND MAGGINI, M. Similarity learning for graph-based image representations. *Pattern Recognition Letters* 24, 8 (May 2003), 1115–1122.
- [69] DE RAEDT, L. *Logical and Relational Learning*. Springer, 2008.
- [70] DICKINSON, P. J., BUNKE, H., DADEJ, A., AND KRAETZL, M. On graphs with unique node labels. In *Proceedings of the 4th IAPR International Conference on Graph Based Representations in Pattern Recognition* (2003), GbRPR 2003, Springer-Verlag, pp. 13–23.

- 
- [71] DICKINSON, P. J., KRAETZL, M., BUNKE, H., NEUHAUS, M., AND DADEJ, A. Similarity measures for hierarchical representations of graphs with unique node labels. *International Journal of Pattern Recognition and Artificial Intelligence* 18, 3 (2004), 425–442.
- [72] DUDA, R. O., HART, P. E., AND STORK, D. G. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.
- [73] DZEROSKI, S., BLOCKEEL, H., KOMPARE, B., KRAMER, S., PFAHRINGER, B., AND VAN LAER, W. Experiments in predicting biodegradability. In *Applied Artificial Intelligence* (1999), Springer, pp. 80–91.
- [74] ELLISON, N. B., STEINFELD, C., AND LAMPE, C. The benefits of facebook "friends:" social capital and college students' use of online social network sites. *Journal of Computer-Mediated Communication* 12, 4 (2007), 1143–1168.
- [75] EMMS, D., WILSON, R. C., AND HANCOCK, E. Graph embedding using quantum commute times. In *Proceedings of the 6th IAPR-TC-15 International Conference on Graph-based Representations in Pattern Recognition* (2007), GbRPR 2007, Springer-Verlag, pp. 371–382.
- [76] FALOUTSOS, M., FALOUTSOS, P., AND FALOUTSOS, C. On power-law relationships of the internet topology. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (1999), SIGCOMM 1999, ACM, pp. 251–262.
- [77] FERRER, M., SERRATOSA, F., AND SANFELIU, A. Synthesis of median spectral graph. In *Proceedings of the Second Iberian conference on Pattern Recognition and Image Analysis - Volume Part II* (2005), IbPRIA 2005, Springer-Verlag, pp. 139–146.
- [78] FERRER, M., VALVENY, E., SERRATOSA, F., BARDAJÍ, I., AND BUNKE, H. Graph-based k-means clustering: A comparison of the set median versus the generalized median graph. *Computer Analysis of Images and Patterns* (2009), 342–350.
- [79] FLAKE, G. W., TARJAN, R. E., AND TSIOUTSIOLIKLIS, K. Graph clustering and minimum cut trees. *Internet Mathematics* 1, 4 (2003), 385–408.
- [80] FONTAINE, F., PASTOR, M., ZAMORA, I., AND SANZ, F. Anchor–grind: Filling the gap between standard 3d qsar and the grid-independent descriptors. *Journal of Medicinal Chemistry* 48 (2005), 2687–2694.
- [81] FORTUNATO, S. Community detection in graphs. *Physics Reports* 486, 3-5 (2010), 75 – 174.

- [82] FRÖHLICH, H., WEGNER, J. K., SIEKER, F., AND ZELL, A. Optimal assignment kernels for attributed molecular graphs. In *Proceedings of the 22nd International Conference on Machine learning* (2005), ICML 2005, ACM, pp. 225–232.
- [83] FU, X., ONG, C., KEERTHI, S., HUNG, G. G., AND GOH, L. Extracting the knowledge embedded in support vector machines. In *Proceedings of the 2004 IEEE International Joint Conference on Neural Networks* (2004), vol. 1 of *IJCNN 2004*, pp. 291–296.
- [84] FUNG, G., SANDILYA, S., AND RAO, R. B. Rule extraction from linear support vector machines. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2005), KDD 2005, ACM, pp. 32–40.
- [85] FUREY, T. S., CRISTIANINI, N., DUFFY, N., BEDNARSKI, D. W., SCHUMMER, M., AND HAUSSLER, D. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics* 16, 10 (2000), 906–914.
- [86] GARCEZ, A. S. D., BRODA, K., AND GABBAY, D. M. Symbolic knowledge extraction from trained neural networks: A sound approach. *Artificial Intelligence* 125, 1-2 (2001), 155–207.
- [87] GÄRTNER, T. *Kernels for Structured Data*. PhD thesis, Universität Bonn, 2005.
- [88] GÄRTNER, T., FLACH, P. A., AND WROBEL, S. On graph kernels: Hardness results and efficient alternatives. In *Proceedings of the 16th Annual Conference on Computational Learning Theory* (2003), COLT 2003, Springer, pp. 129–143.
- [89] GASTEIGER, J., AND ENGEL, T., Eds. *Chemoinformatics: A Textbook*, 1 ed. Wiley-VCH, 2003.
- [90] GEAMSAKUL, W., MATSUDA, T., YOSHIDA, T., MOTODA, H., AND WASHIO, T. Performance evaluation of decision tree graph-based induction. In *Proceedings of the 6th International Conference on Discovery Science* (2003), DS 2003, Springer, pp. 128–140.
- [91] GIBSON, D., KUMAR, R., AND TOMKINS, A. Discovering large dense subgraphs in massive graphs. In *Proceedings of the 31st International Conference on Very large Data Bases* (2005), VLDB 2005, VLDB Endowment, pp. 721–732.
- [92] GOLD, L. S. The Carcinogenic Potency Database (CPDB): <http://potency.berkeley.edu/>. Last accessed on 2014-04-27.
- [93] GOLD, L. S., SLONE, T., AMES, B. N., MANLEY, N. B., GARFINKEL, G. B., AND ROHRBACH, L. *Handbook of Carcinogenic Potency and Genotoxicity Databases*. CRC Press, 1997, ch. Carcinogenic Potency Database, pp. 1–605.



- 
- [94] GÜNTER, S. *Graph clustering using Kohonen's method, Master's thesis*. PhD thesis, University of Bern, 2000.
- [95] GÜNTER, S., AND BUNKE, H. Self-organizing map for clustering in the graph domain. *Pattern Recognition Letters* 23, 4 (2002), 405–417.
- [96] GÜNTER, S., AND BUNKE, H. Validation indices for graph clustering. *Pattern Recognition Letters* 24, 8 (2003), 1107–1113.
- [97] HALL, M., FRANK, E., HOLMES, G., PFAHRINGER, B., REUTEMANN, P., AND WITTEN, I. H. The WEKA data mining software: an update. *SIGKDD Explorations* 11, 1 (2009), 10–18.
- [98] HAN, J. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.
- [99] HAN, J.-D. J., BERTIN, N., HAO, T., GOLDBERG, D. S., BERRIZ, G. F., ZHANG, L. V., DUPUY, D., WALHOUT, A. J. M., CUSICK, M. E., ROTH, F. P., AND VIDAL, M. Evidence for dynamically organized modularity in the yeast protein-protein interaction network. *Nature* 430, 6995 (2004), 88–93.
- [100] HARCHAOUI, Z., AND BACH, F. Image classification with segmentation graph kernels. In *IEEE Conference on Computer Vision and Pattern Recognition* (2007), CVPR 2007, IEEE Computer Society, pp. 1–8.
- [101] HAUSSLER, D. Convolution kernels on discrete structures. Technical report, University of California at Santa Cruz, Santa Cruz, CA, USA, 1999.
- [102] HELMA, C., CRAMER, T., KRAMER, S., AND DE RAEDT, L. Data mining and machine learning techniques for the identification of mutagenicity inducing substructures and structure activity relationships of noncongeneric compounds. *Journal of Chemical Information and Computer Sciences* 44, 4 (2004), 1402–1411.
- [103] HLAOUI, A., AND WANG, S. Median graph computation for graph clustering. *Soft Computing* 10, 1 (2006), 47–53.
- [104] HOERL, A. E., AND KENNARD, R. W. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics* 12 (1970), 55–67.
- [105] HONEINE, P., AND RICHARD, C. A closed-form solution for the pre-image problem in kernel-based machines. *Journal of Signal Processing Systems* 65, 3 (2011), 289–299.
- [106] HORVÁTH, T., GÄRTNER, T., AND WROBEL, S. Cyclic pattern kernels for predictive graph mining. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2004), KDD 2004, ACM, pp. 158–167.

- [107] HOSSAIN, M. S., AND ANGRYK, R. A. GDClust: A graph-based document clustering technique. In *Proceedings of the Seventh IEEE International Conference on Data Mining Workshops* (2007), ICDMW 2007, IEEE Computer Society, pp. 417–422.
- [108] HOU, T., AND XU, X. Recent development and application of virtual screening in drug discovery: an overview. *Current Pharmaceutical Design* 10, 9 (2004), 1011–1033.
- [109] HUAN, J., WANG, W., AND PRINS, J. Efficient mining of frequent subgraphs in the presence of isomorphism. In *Proceedings of the Third IEEE International Conference on Data Mining* (2003), ICDM 2003, IEEE Computer Society, pp. 549–552.
- [110] HUAN, J., WANG, W., PRINS, J., AND YANG, J. SPIN: Mining maximal frequent subgraphs from graph databases. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2004), KDD 2004, ACM, pp. 581–586.
- [111] HUYSMANS, J., BAESSENS, B., AND VANTHIENEN, J. ITER: An algorithm for predictive regression rule extraction. In *Proceedings of the Eighth International Conference on Data Warehousing and Knowledge Discovery* (2006), A. M. Tjoa and J. Trujillo, Eds., DaWaK 2006, Springer, pp. 270–279.
- [112] INOKUCHI, A., WASHIO, T., AND MOTODA, H. 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* (2000), PKDD 2000, Springer-Verlag, pp. 13–23.
- [113] INOKUCHI, A., WASHIO, T., AND MOTODA, H. Complete mining of frequent patterns from graphs: Mining graph data. *Machine Learning* 50, 3 (2003), 321–354.
- [114] JACCARD, P. Nouvelles recherches sur la distribution florale. *Bulletin de la Société Vaudoise des Sciences Naturelles* 44 (1908), 223–270.
- [115] JAHN, K., AND KRAMER, S. Optimizing gSpan for molecular datasets. In *Proceedings of the Third International Workshop on Mining Graphs, Trees and Sequences* (2005), MGTS 2005.
- [116] JAIN, A. K., AND DUBES, R. C. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.
- [117] JAIN, A. K., MURTY, M. N., AND FLYNN, P. J. Data clustering: A review. *ACM Computing Surveys* 31, 3 (Sept. 1999), 264–323.
- [118] JCHEM JAVA PACKAGE. Version 5.1.3 2, ChemAxon, 2008: <http://www.chemaxon.com>. Last accessed on 2014-04-27.

- 
- [119] JIANG, X., MÜUNGER, A., AND BUNKE, H. On median graphs: Properties, algorithms, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23, 10 (2001), 1144–1151.
- [120] JOACHIMS, T. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning* (1998), ECML 1998, Springer-Verlag, pp. 137–142.
- [121] JOHANSSON, U., KÖNIG, R., AND NIKLASSON, L. The truth is in there - rule extraction from opaque models using genetic programming. In *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference* (2004), V. Barr and Z. Markov, Eds., FLAIRS Conference 2004, AAAI Press, pp. 658–663.
- [122] JORISSEN, R. N., AND GILSON, M. K. Virtual screening of molecular databases using a support vector machine. *Journal of Chemical Information and Modeling* 45, 3 (2005), 549–561.
- [123] JOUILI, S., AND TABBONE, S. A hypergraph-based model for graph clustering: Application to image indexing. In *Computer Analysis of Images and Patterns*, X. Jiang and N. Petkov, Eds., vol. 5702 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2009, pp. 360–368.
- [124] JOUILI, S., TABBONE, S., AND LACROIX, V. Median graph shift: A new clustering algorithm for graph domain. In *20th International Conference on Pattern Recognition* (2010), ICPR 2010, IEEE, pp. 950–953.
- [125] KARGER, D. R. Random sampling in cut, flow, and network design problems. In *Proceedings of the 26th Annual ACM symposium on Theory of Computing* (1994), STOC 1994, ACM, pp. 648–657.
- [126] KARYPIS, G., AND KUMAR, V. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing* 20, 1 (Dec. 1998), 359–392.
- [127] KASHIMA, H., TSUDA, K., AND INOKUCHI, A. Marginalized kernels between labeled graphs. In *Proceedings of the Twentieth International Conference on Machine Learning* (2003), ICML 2003, AAAI Press, pp. 321–328.
- [128] KAUFMAN, L., AND ROUSSEEUW, P. J. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley, 1990.
- [129] KAZIUS, J., MCGUIRE, R., AND BURSI, R. Derivation and validation of toxicophores for mutagenicity prediction. *Journal of Medicinal Chemistry* 48, 1 (2005), 312–320.

- [130] KECMAN, V. *Learning and Soft Computing: Support Vector Machines, Neural Networks, and Fuzzy Logic Models*. MIT Press, Cambridge, MA, USA, 2001.
- [131] KERNIGHAN, B. W., AND LIN, S. An Efficient Heuristic Procedure for Partitioning Graphs. *The Bell system technical journal* 49, 1 (1970), 291–307.
- [132] KILPELÄINEN, P. Tree matching problems with applications to structured text databases. Tech. rep., 1992.
- [133] KLOFT, M., RÜCKERT, U., AND BARTLETT, P. L. A unifying view of multiple kernel learning. In *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part II* (2010), ECML PKDD 2010, Springer-Verlag, pp. 66–81.
- [134] KOHONEN, T. *Self-Organizing maps*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.
- [135] KOUTSOUKOS, A. D., RUBINSTEIN, L. V., FARAGGI, D., SIMON, R. M., KALYANDRUG, S., WEINSTEIN, J. N., KOHN, K. W., AND PAULL, K. D. Discrimination techniques applied to the NCI in vitro anti-tumour drug screen: Predicting biochemical mechanism of action. *Statistics in Medicine* 13 (1994), 719–730.
- [136] KOZA, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.
- [137] KRAMER, S., DE RAEDT, L., AND HELMA, C. Molecular feature mining in hiv data. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2001), KDD 2001, ACM, pp. 136–143.
- [138] KRIEGE, N., AND MUTZEL, P. Subgraph matching kernels for attributed graphs. In *Proceedings of the 29th International Conference on Machine Learning* (2012), ICML 2012, icml.cc / Omnipress, pp. 1015–1022.
- [139] KUDO, T., EISAKU, M., AND MATSUMOTO, Y. An application of boosting to graph classification. In *Advances in Neural Information Processing Systems 17* (2004), NIPS 2004.
- [140] KUMAR, R., RAGHAVAN, P., RAJAGOPALAN, S., SIVAKUMAR, D., TOMPKINS, A., AND UPFAL, E. The web as a graph. In *Proceedings of the 19th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (2000), PODS 2000, ACM, pp. 1–10.
- [141] KURAMOCHI, M., AND KARYPIS, G. An efficient algorithm for discovering frequent subgraphs. *IEEE Transactions on Knowledge and Data Engineering* 16, 9 (Sept. 2004), 1038–1051.

- 
- [142] KURIHARA, K., WELLING, M., AND VLASSIS, N. Accelerated variational dirichlet process mixtures. In *Advances in Neural Information Processing Systems* (2006), NIPS 2006, pp. 761–768.
- [143] LE SAUX, B., AND BUNKE, H. Feature selection for graph-based image classifiers. In *Proceedings of the Second Iberian Conference on Pattern Recognition and Image Analysis - Volume Part II* (2005), IbPRIA 2005, Springer-Verlag, pp. 147–154.
- [144] LESLIE, C. S., ESKIN, E., COHEN, A., WESTON, J., AND NOBLE, W. S. Mismatch string kernels for discriminative protein classification. *Bioinformatics* 20, 4 (2004), 467–476.
- [145] LI, H., YAP, C. W., UNG, C. Y., XUE, Y., CAO, Z. W., AND CHEN, Y. Z. Effect of selection of molecular descriptors on the prediction of blood-brain barrier penetrating and nonpenetrating agents by statistical learning methods. *Journal of Chemical Information and Modeling* 45, 5 (2005), 1376–1384.
- [146] LIU, D., HUA, X.-S., YANG, L., AND ZHANG, H.-J. Multiple-instance active learning for image categorization. In *Proceedings of the 15th International Multimedia Modeling Conference on Advances in Multimedia Modeling* (2008), MMM 2009, Springer-Verlag, pp. 239–249.
- [147] LUO, B., WILSON, R. C., AND HANCOCK, E. R. The independent and principal component of graph spectra. In *Proceedings of the 16th International Conference on Pattern Recognition* (2002), ICPR 2002, pp. 164–167.
- [148] LUO, B., WILSON, R. C., AND HANCOCK, E. R. Spectral feature vectors for graph clustering. In *Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition* (2002), Springer-Verlag, pp. 83–93.
- [149] LUO, B., WILSON, R. C., AND HANCOCK, E. R. Spectral embedding of graphs. *Pattern Recognition* 36, 10 (2003), 2213–2223.
- [150] LYNE, P. D. Structure-based virtual screening: An overview. *Drug Discovery Today* 7, 20 (2002), 1047 – 1055.
- [151] MACQUEEN, J. B. Some methods for classification and analysis of multivariate observations. In *5th Berkeley Symposium on Mathematical Statistics and Probability* (1967), vol. 1, University of California Press, pp. 281–297.
- [152] MAHÉ, P., UEDA, N., AKUTSU, T., PERRET, J.-L., AND VERT, J.-P. Graph kernels for molecular structure–activity relationship analysis with support vector machines. *Journal of Chemical Information and Modeling* 45, 4 (2005), 939–951.
- [153] MAHÉ, P., AND VERT, J.-P. Graph kernels based on tree patterns for molecules. *Machine Learning* 75, 1 (Apr. 2009), 3–35.

- [154] MAIO, D., AND MALTONI, D. A structural approach to fingerprint classification. In *Proceedings of the 13th International Conference on Pattern Recognition* (1996), vol. 3, pp. 578–585.
- [155] MANGAN, S., AND ALON, U. Structure and function of the feed-forward loop network motif. *Proceedings of the National Academy of Sciences* 100, 21 (2003), 11980–11985.
- [156] MANNINO, M. V., AND KOUSHIK, M. V. The cost-minimizing inverse classification problem: a genetic algorithm approach. *Decision Support Systems* 29, 3 (2000), 283–300.
- [157] MARTENS, D., BAESENS, B., AND VAN GESTEL, T. Decompositional rule extraction from support vector machines by active learning. *IEEE Transactions on Knowledge and Data Engineering* 21, 2 (Feb. 2009), 178–191.
- [158] MARTENS, D., BAESENS, B., VAN GESTEL, T., AND VANTHIENEN, J. Comprehensive credit scoring models using rule extraction from support vector machines. *European Journal of Operational Research* 183, 3 (2007), 1466–1476.
- [159] MARTENS, D., HUYSMANS, J., SETIONO, R., VANTHIENEN, J., AND BAESENS, B. Rule extraction from support vector machines: An overview of issues and application in credit scoring. In *Rule Extraction from Support Vector Machines*, J. Diederich, Ed., vol. 80 of *Studies in Computational Intelligence*. Springer, 2008, pp. 33–63.
- [160] MARTIN, Y. C., KOFRON, J. L., AND TRAPHAGEN, L. M. Do structurally similar molecules have similar biological activity? *Journal of Medicinal Chemistry* 45 (2002), 4350–4358.
- [161] MATSUDA, T., MOTODA, H., YOSHIDA, T., AND WASHIO, T. Mining patterns from structured data by beam-wise graph-based induction. In *Proceedings of the 5th International Conference on Discovery Science* (2002), DS 2002, Springer-Verlag, pp. 422–429.
- [162] MATTHEWS, E., KRULAK, N., BENZ, R., AND CONTRERA, J. Assessment of the health effects of chemicals in humans: I. QSAR estimation of the maximum recommended therapeutic dose (MRTD) and no effect level (NOEL) of organic chemicals based on clinical trial data. *Current Drug Discovery Technologies* 1, 1 (2004), 61–76.
- [163] MAUNZ, A., HELMA, C., AND KRAMER, S. Efficient mining for structurally diverse subgraph patterns in large molecular databases. *Machine Learning* 83, 2 (2011), 193–218.
- [164] MCGREGOR, M. J., AND PALLAI, P. V. Clustering of large databases of compounds: Using the MDL "keys" as structural descriptors. *Journal of Chemical Information and Computer Sciences* 37, 3 (1997), 443–448.

- 
- [165] MENCHETTI, S. *Learning Preference and Structured Data: Theory and Applications*. PhD thesis, Dipartimento di Sistemi e Informatica, DSI, Università di Firenze, Italy, 2005.
- [166] MENCHETTI, S., COSTA, F., AND FRASCONI, P. Weighted decomposition kernels. In *Proceedings of the 22nd International Conference on Machine Learning (2005)*, ICML 2005, ACM, pp. 585–592.
- [167] MESCHKOWSKI, H. *Hilbertsche Räume mit Kernfunktion*. Springer Verlag, 1962.
- [168] MESSMER, B. T., AND BUNKE, H. A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20, 5 (May 1998), 493–504.
- [169] MILO, R., SHEN-ORR, S., ITZKOVITZ, S., KASHTAN, N., CHKLOVSKII, D., AND ALON, U. Network motifs: simple building blocks of complex networks. *Science* 298, 5594 (2002), 824–827.
- [170] MITCHELL, T. M. *Machine Learning*, 1 ed. McGraw-Hill, Inc., New York, NY, USA, 1997.
- [171] MORISHITA, S., AND SESE, J. Transversing itemset lattices with statistical metric pruning. In *Proceedings of the 19th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (2000)*, PODS 2000, ACM, pp. 226–236.
- [172] NADEAU, C., AND BENGIO, Y. Inference for the generalization error. *Machine Learning* 52, 3 (2003), 239–281.
- [173] NIJSSEN, S., AND KOK, J. N. A quickstart in frequent structure mining can make a difference. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2004)*, KDD 2004, ACM, pp. 647–652.
- [174] NIJSSEN, S., AND KOK, J. N. The gaston tool for frequent subgraph mining. *Electronic Notes in Theoretical Computer Science* 127, 1 (2005), 77–87.
- [175] NÚÑEZ, H., ANGULO, C., AND CATALÀ, A. Rule extraction from support vector machines. In *10th Euroean Symposium on Artificial Neural Networks (2002)*, ESANN 2002, pp. 107–112.
- [176] PUBCHEM COMPOUND DATABASE. [http://www.ncbi.nlm.nih.gov/pccompound?term=all\[filt\]&cmd=search](http://www.ncbi.nlm.nih.gov/pccompound?term=all[filt]&cmd=search). Last accessed on 2014-04-27.
- [177] QUINLAN, J. R. *C4.5: Programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
- [178] RAGHAVAN, S., AND GARCIA-MOLINA, H. Representing web graphs. In *Proceedings of the 19th International Conference on Data Engineering (2003)*, ICDE 2003, IEEE Computer Society, pp. 405–416.

- [179] RALAIVOLA, L., SWAMIDASS, S. J., SAIGO, H., AND BALDI, P. Graph kernels for chemical informatics. *Neural Networks* 18, 8 (2005), 1093–1110.
- [180] RAMON, J., AND GÄRTNER, T. Expressivity versus efficiency of graph kernels. In *Proceedings of the First International Workshop on Mining Graphs, Trees and Sequences* (2003), pp. 65–74.
- [181] RATTIGAN, M. J., MAIER, M., AND JENSEN, D. Graph clustering with network structure indices. In *Proceedings of the 24th International Conference on Machine Learning* (2007), ICML 2007, ACM, pp. 783–790.
- [182] RAYMOND, J. W., BLANKLEY, C. J., AND WILLETT, P. Comparison of chemical clustering methods using graph-based and fingerprint-based similarity measures. *Journal of Molecular Graphics and Modelling* 21, 5 (2003), 421–433.
- [183] RAYMOND, J. W., GARDINER, E. J., AND WILLETT, P. Heuristics for rapid similarity searching of chemical graphs using a maximum common edge subgraph algorithm. *Journal of Chemical Information and Computer Sciences* 42 (2002), 305–316.
- [184] RAYMOND, J. W., GARDINER, E. J., AND WILLETT, P. RASCAL: Calculation of graph similarity using maximum common edge subgraphs. *The Computer Journal* 45 (2002), 2002.
- [185] RAYMOND, J. W., AND WILLETT, P. Effectiveness of graph-based and fingerprint-based similarity measures for virtual screening of 2D chemical structure databases. *Journal of Computer-Aided Molecular Design* 16, 1 (2002), 59–71.
- [186] REGULATION (EC). No 1907/2006 of the European Parliament and of the Council of 18 December 2006 concerning the Registration, Evaluation, Authorisation and Restriction of Chemicals (REACH), establishing a European Chemicals Agency, 2006.
- [187] REN, L., AND D’AVILA GARCEZ, A. S. Symbolic knowledge extraction from support vector machines: A geometric approach. In *Proceedings of the 15th International Conference on Advances in Neuro-Information Processing* (2009), ICONIP 2008, Springer, pp. 335–343.
- [188] RIESEN, K., NEUHAUS, M., AND BUNKE, H. Graph embedding in vector spaces by means of prototype selection. In *Proceedings of the 6th IAPR-TC-15 International Conference on Graph-based Representations in Pattern Recognition* (2007), GbRPR 2007, Springer-Verlag, pp. 383–393.
- [189] ROBLES-KELLY, A., AND HANCOCK, E. R. A riemannian approach to graph embedding. *Pattern Recognition* 40, 3 (Mar. 2007), 1042–1056.



- 
- [190] RÜCKERT, U., GIRSCHICK, T., BUCHWALD, F., AND KRAMER, S. Adapted transfer of distance measures for quantitative structure-activity relationships. In *Proceedings of the 13th International Conference on Discovery Science* (2010), DS 2010, Springer-Verlag, pp. 341–355.
- [191] RÜCKERT, U., GIRSCHICK, T., BUCHWALD, F., AND KRAMER, S. Adapted transfer of distance measures for quantitative structure-activity relationships. In *Proceedings of the 13th International Conference on Discovery Science* (2010), DS 2010, Springer-Verlag, pp. 341–355.
- [192] RÜPING, S. Globalization of local models with SVMs. In *LeGo-08 - From Local Patterns to Global Models, Workshop at ECML/PKDD* (2008).
- [193] RUSSOM, C., BRADBURY, S., BRODERIUS, D., HAMMERMEISTER, S., AND DRUMMOND, R. Predicting modes of action from chemical structure: Acute toxicity in the fathead minnow (*pimephales promelas*). *Environmental Toxicology and Chemistry* 5, 16 (1997), 948–967.
- [194] SANFELIU, A., ALQUÉZAR, R., AND SERRATOSA, F. Clustering of attributed graphs and unsupervised synthesis of function-described graphs. In *Proceedings of the 15th International Conference on Pattern Recognition* (2000), ICPR, pp. 6022–6025.
- [195] SCHAEFFER, S. E. Graph clustering. *Computer Science Review* 1, 1 (2007), 27–64.
- [196] SCHENKER, A., BUNKE, H., LAST, M., AND KANDEL, A. Graph-theoretic techniques for web content mining. *World Scientific Publishing* (2005).
- [197] SCHENKER, A., LAST, M., BUNKE, H., AND KANDEL, A. Classification of web documents using graph matching. *International Journal of Pattern Recognition and Artificial Intelligence* 18, 03 (2004), 475–496.
- [198] SCHIETGAT, L., COSTA, F., RAMON, J., AND DE RAEDT, L. Effective feature construction by maximum common subgraph sampling. *Machine Learning* 83, 2 (2011), 137–161.
- [199] SCHÖLKOPF, B., AND SMOLA, A. J. *Learning with Kernels*. MIT Press, 2002.
- [200] SCHÖLKOPF, B., SMOLA, A. J., WILLIAMSON, R. C., AND BARTLETT, P. L. New support vector algorithms. *Neural Computation* 12, 5 (2000), 1207–1245.
- [201] SCHÖLKOPF, B., TSUDA, K., AND VERT, J. P. *Kernel Methods in Computational Biology*. MIT Press, 2004.
- [202] SEELAND, M., BERGER, S. A., STAMATAKIS, A., AND KRAMER, S. Parallel structural graph clustering. In *Proceedings of the 2011 European Conference on Machine*

- Learning and Knowledge Discovery in Databases - Volume Part III* (2011), ECML PKDD 2011, Springer-Verlag, pp. 256–272.
- [203] SEELAND, M., BUCHWALD, F., KRAMER, S., AND PFAHRINGER, B. Maximum common subgraph based locally weighted regression. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing* (2012), SAC 2012, ACM, pp. 165–172.
- [204] SEELAND, M., GIRSCHICK, T., BUCHWALD, F., AND KRAMER, S. Online structural graph clustering using frequent subgraph mining. In *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part III* (2010), ECML PKDD 2010, Springer-Verlag, pp. 213–228.
- [205] SEELAND, M., KARWATH, A., AND KRAMER, S. A structural cluster kernel for learning on graphs. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2012), KDD 2012, ACM, pp. 516–524.
- [206] SEELAND, M., KARWATH, A., AND KRAMER, S. Structural clustering of millions of molecular graphs. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing* (2014), SAC 2014, ACM, pp. 121–128.
- [207] SEELAND, M., KRAMER, S., AND PFAHRINGER, B. Model selection based product kernel learning for regression on graphs. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing* (2013), SAC 2013, ACM, pp. 136–143.
- [208] SEELAND, M., MAUNZ, A., KARWATH, A., AND KRAMER, S. Extracting information from support vector machines for pattern-based classification. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing* (2014), SAC 2014, ACM, pp. 129–136.
- [209] SHASHA, D., WANG, J. T. L., AND GIUGNO, R. Algorithmics and applications of tree and graph searching. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (2002), PODS 2002, ACM, pp. 39–52.
- [210] SHAWE-TAYLOR, J., AND CRISTIANINI, N. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.
- [211] SHERVASHIDZE, N., AND BORGWARDT, K. M. Fast subtree kernels on graphs. In *Advances in Neural Information Processing Systems 22* (2009), NIPS 2009, Curran, pp. 1660–1668.
- [212] SHI, Q., PETTERSON, J., DROR, G., LANGFORD, J., SMOLA, A. J., STREHL, A. L., AND VISHWANATHAN, V. Hash kernels. In *Proceedings of the Twelfth Inter-*

- national Conference on Artificial Intelligence and Statistics* (2009), AISTATS 2009, pp. 496–503.
- [213] STAHL, M., AND MAUSER, H. Database clustering with a combination of fingerprint and maximum common substructure methods. *Journal of Chemical Information and Modeling* 45 (2005), 542–548.
- [214] STEINHAUS, H. Sur la division des corp materiels en parties. *Bulletin de l'Académie Polonaise des Sciences* 1 (1956), 801–804.
- [215] SUBIANTO, M., AND SIEBES, A. Understanding discrete classifiers with a case study in gene prediction. In *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining* (2007), ICDM 2007, IEEE Computer Society, pp. 661–666.
- [216] SUTHERLAND, J. J., O'BRIEN, L. A., AND WEAVER, D. F. Spline-fitting with a genetic algorithm: A method for developing classification structure-activity relationships. *Journal of Chemical Information and Computer Sciences* 43, 6 (2003), 1906–1915.
- [217] SUTHERLAND, J. J., O'BRIEN, L. A., AND WEAVER, D. F. A comparison of methods for modeling quantitative structure-activity relationships. *Journal of Medicinal Chemistry* 47, 22 (2004), 5541–5554.
- [218] THRUN, S. B. Extracting provably correct rules from artificial neural networks. Tech. rep., University of Bonn, 1993.
- [219] TONG, S., AND CHANG, E. Support vector machine active learning for image retrieval. In *Proceedings of the Ninth ACM International Conference on Multimedia* (2001), Multimedia 2001, ACM, pp. 107–118.
- [220] TONG, S., AND KOLLER, D. Active learning for parameter estimation in bayesian networks. In *Neural Information Processing Systems* (2000), NIPS 2000, pp. 647–653.
- [221] TORSELLO, A., AND HANCOCK, E. R. Graph embedding using tree edit-union. *Pattern Recognition* 40, 5 (May 2007), 1393–1405.
- [222] TSUDA, K. Support vector classifier with asymmetric kernel functions. In *Proceedings of the Seventh European Symposium on Artificial Neural Networks* (1999), ESANN 1999, pp. 183–188.
- [223] TSUDA, K., AND KUDO, T. Clustering graphs by weighted substructure mining. In *Proceedings of the 23rd International Conference on Machine Learning* (2006), ICML 2006, ACM, pp. 953–960.

- [224] TSUDA, K., AND KUDO, T. Clustering graphs by weighted substructure mining. In *Proceedings of the 23rd International Conference on Machine learning* (2006), ICML 2006, ACM, pp. 953–960.
- [225] TSUDA, K., AND KURIHARA, K. Graph mining with variational Dirichlet process mixture models. In *Proceedings of the Eighth SIAM International Conference on Data Mining* (2008), SDM 2008, pp. 432–442.
- [226] VAPNIK, V. *Statistical learning theory*. Wiley, 1998.
- [227] VAPNIK, V., AND LERNER, A. Pattern recognition using generalized portrait method. *Automation and Remote Control* 24 (1963), 774–780.
- [228] VAPNIK, V. N. *The nature of statistical learning theory*. Springer-Verlag, 1995.
- [229] VISHWANATHAN, S. V. N., BORGWARDT, K. M., AND SCHRAUDOLPH, N. N. Fast computation of graph kernels. In *Proceedings of the 20th Annual Conference on Neural Information Processing Systems* (2006), NIPS 2006, pp. 1449–1456.
- [230] VISHWANATHAN, S. V. N., SCHRAUDOLPH, N. N., KONDOR, R., AND BORGWARDT, K. M. Graph kernels. *Journal of Machine Learning Research* 11 (2010), 1201–1242.
- [231] VREEKEN, J., LEEUWEN, M., AND SIEBES, A. KRIMP: Mining itemsets that compress. *Data Mining and Knowledge Discovery* 23, 1 (2011), 169–214.
- [232] WALLIS, W. D., SHOUBRIDGE, P., KRAETZ, M., AND RAY, D. Graph distances using graph union. *Pattern Recognition Letters* 22, 6-7 (2001), 701–704.
- [233] WALTERS, W. P., STAHL, M. T., AND MURCKO, M. A. Virtual screening - An overview. *Drug Discovery Today* 3, 4 (1998), 160 – 178.
- [234] WANG, K., AND HAN, J. Bide: Efficient mining of frequent closed sequences. In *International Conference on Data Engineering* (2004).
- [235] WASZKOWYCZ, B., PERKINS, T. D. J., SYKES, R. A., AND LI, J. Large-scale virtual screening for discovering leads in the postgenomic era. *IBM Systems Journal* 40, 2 (Feb. 2001), 360–376.
- [236] WEGNER, J. Joelib2. <http://www-ra.informatik.uni-tuebingen.de/software/joelib/>.
- [237] WEINSTEIN, J., KOHN, K., GREVER, M., AND VISWANADHAN, V. Neural computing in cancer drug development: Predicting mechanism of action. *Science* 258 (1992), 447–451.
- [238] WEISLOW, O., KISER, R., FINE, D., BADER, J., SHOEMAKER, R., AND BOYD, M. New soluble formazan assay for HIV-1 cytopathic effects: Application to high

- flux screening of synthetic and natural products for AIDS antiviral activity. *Journal National Cancer Institute* 81 (1989), 577–586.
- [239] WESTON, J., CHAPELLE, O., ELISSEEFF, A., SCHÖLKOPF, B., AND VAPNIK, V. Kernel dependency estimation. In *Advances in Neural Information Processing Systems* (2002), NIPS 2002, pp. 873–880.
- [240] WESTON, J., LESLIE, C., IE, E., ZHOU, D., ELISSEEFF, A., AND NOBLE, W. S. Semi-supervised protein classification using cluster kernels. *Bioinformatics* 21, 15 (2005), 3241–3247.
- [241] WILLETT, P., BARNARD, J. M., AND DOWNS, G. M. Chemical similarity searching. *Journal of Chemical Information and Computer Sciences* 38, 6 (1998), 983–996.
- [242] WILTON, D., WILLETT, P., LAWSON, K., AND MULLIER, G. Comparison of ranking methods for virtual screening in lead-discovery programs. *Journal of Chemical Information and Computer Sciences* 43, 2 (2003), 469–474.
- [243] XU, R., AND WUNSCH, D., I. Survey of clustering algorithms. *IEEE Transactions on Neural Networks* 16, 3 (May 2005), 645–678.
- [244] YAN, X., CHENG, H., HAN, J., AND YU, P. S. Mining significant graph patterns by leap search. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data* (2008), SIGMOD 2008, ACM, pp. 433–444.
- [245] YAN, X., AND HAN, J. gSpan: Graph-based substructure pattern mining. In *Proceedings of the 2002 IEEE International Conference on Data Mining* (2002), ICDM 2002, IEEE Computer Society, pp. 721–724.
- [246] YAN, X., AND HAN, J. CloseGraph: Mining closed frequent graph patterns. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2003), KDD 2003, ACM, pp. 286–295.
- [247] YAO, J. T. Sensitivity analysis for data mining. In *Proceedings of the 22nd International Conference of the North American Fuzzy Information Processing Society* (2003), NAFIPS 2003, pp. 272–277.
- [248] YAP, C. W., AND CHEN, Y. Z. Prediction of cytochrome P450 3A4, 2D6, and 2C9 inhibitors and substrates by using support vector machines. *Journal of Chemical Information and Modeling* 45, 4 (2005), 982–992.
- [249] YOSHIDA, K., AND MOTODA, H. CLIP: Concept learning from inference patterns. *Artificial Intelligence* 75, 1 (May 1995), 63–92.
- [250] YOSHIDA, T., SHODA, R., AND MOTODA, H. Graph clustering based on structural similarity of fragments. In *Federation over the Web*, K. Jantke, A. Lunzer,

- N. Spyratos, and Y. Tanaka, Eds., vol. 3847 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006, pp. 97–114.
- [251] ZAKI, M. J., AND AGGARWAL, C. C. XRules: An effective structural classifier for XML data. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2003), KDD 2003, ACM, pp. 316–325.
- [252] ZENG, Z., WANG, J., ZHOU, L., AND KARYPIS, G. Out-of-core coherent closed quasi-clique mining from large dense graph databases. *ACM Transactions on Database Systems* 32, 2 (2007).
- [253] ZHANG, T., RAMAKRISHNAN, R., AND LIVNY, M. BIRCH: An efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data* (1996), SIGMOD 1996, ACM, pp. 103–114.
- [254] ZHANG, T., RAMAKRISHNAN, R., AND LIVNY, M. BIRCH: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery* 1, 2 (1997), 141–182.
- [255] ZHANG, Y., LI, Z., TANG, Y., AND CUI, K. DRC-BK: Mining classification rules with help of SVM. In *Advances in Knowledge Discovery and Data Mining, 8th Pacific-Asia Conference* (2004), PAKDD 2004, pp. 191–195.
- [256] ZHANG, Y., SU, H., JIA, T., AND CHU, J. Rule extraction from trained support vector machines. In *Proceedings of the 9th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining* (2005), PAKDD 2005, Springer-Verlag, pp. 61–70.
- [257] ZHENG, Z., KRAMER, S., AND SCHMIDT, B. DySC: Software for greedy clustering of 16S rRNA reads. *Bioinformatics* 28, 16 (2012).