TECHNISCHE UNIVERSITÄT MÜNCHEN

Fakultät für Informatik
Lehrstuhl für Bioinformatik

Graph Mining Methods for Predictive Toxicology

Andreas Maunz

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. M. Bichler

Prüfer der Dissertation:

1. Univ.-Prof. Dr. B. Rost
2. Univ.-Prof. Dr. St. Kramer
   Johannes Gutenberg-Universität Mainz

Die Dissertation wurde am 26.06.2013 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 05.12.2013 angenommen.

Acknowledgements

I particularly thank my advisor, Prof. Dr. Stefan Kramer, and my senior scientist, Dr. Christoph Helma.

Thanks to colleagues (in alphabetical order):
Dr. Björn Bringmann, Dipl.-Inf. Martin Gütlein, Dr. Alexandros Karatzoglou, Dr. Andreas Karwath, Siegfried Nijssen Ph.D., Ann M. Richard Ph.D., Dr. Ulrich Rückert, Leander Schietgat Ph.D., and all developers in the chemoinformatics community.

This work was supported by OpenTox – An Open Source Predictive Toxicology Framework, funded under the EU Seventh Framework Program: HEALTH-2007-1.3-3 Promotion, Development, Validation, Acceptance and Implementation of QSARs (Quantitative Structure-Activity Relationships) for Toxicology, Project Reference Number Health-F5-2008-200787 (2008-2011).

"The effort being made today to organize knowledge is also a way of participating in the evolution of knowledge itself. In fact, the significance of attempting such organization can be looked for in its ability not only to give information but also to create know-how: it provides not only a collection of facts – a store of information – but also a contribution to the evolution of knowledge. The fact is that splitting the organization of knowledge from its production is completely arbitrary: actually, knowledge organization is itself one way of doing research."

— *Handbook of Molecular Descriptors.* (taken from the introduction)

Abstract

The graph structures of molecules can be a rich source of information about their biological activity or chemical reactivity – however, very efficient methods are required for analyzing them. Due to its complexity, any representation of a chemical database can only convey some characteristics of the whole graph corpus. Additionally, the interesting patterns emerge only from the whole set of graphs that constitute the database, not from individual ones, which places a demand for time- and memory-efficient algorithms.

A primary goal of graph mining is to find subgraphs that occur with a certain frequency in a given dataset. The amount of such patterns is usually enormous for chemical structure graphs, even when additional filters are employed, such as restricting the result set to subgraphs that primarily occur in the toxic or non-toxic compounds. Therefore, the patterns can often not be used directly for predictive modeling, since they would overfit and/or place a high load on learning algorithms, while at the same time provide a much too fine-grained information to experts. More concise representations would have a significant value to the user, even if more time was needed to calculate them.

Concise representations may be obtained, for example, by compression of the pattern set, or lifted representations of molecular fragments. This work shows that such representations may be obtained efficiently in practice, and that they can be of considerable utility for predictive models. It presents a set of algorithmic tools for the extraction of interesting subgraphs and subgraph patterns from molecular databases, and reports on experiments that assess their utility in the context of predictive models. For discovering the most expressive patterns, a combination of structural and statistical constraints is employed. The structural constraints make use of the partial order, in which subgraphs can be put, and on which a refinement operator can be defined. The statistical constraints have the convexity property, allowing for efficient search in combination with the structural constraints. While the approaches are not restricted to chemical structures and toxicological databases, I find the problem of graph mining particularly compelling in this domain, because there has been a rapidly increasing need for efficient and precise computational models in chemical risk assessment during the last decade.

Contents

Chapter 1

Introduction

Chemicals influence biological systems in a huge variety of interactions, mostly on the cellular and molecular level. In toxicology, the aim is to understand the biochemical mechanisms involved, and the degree to which the chemicals induce toxic activity in living organisms, with respect to a well-defined endpoint, such as mutagenicity, or carcinogenicity. In predictive toxicology, toxicological knowledge about a set of chemical compounds (training compounds) is exploited in order to predict the degree of activity of other compounds (query compounds). The training compounds are stored in databases together with their toxicological activity values. Chemical fragments, and other descriptors are routinely used to describe the compounds. Computational methods have been applied to such databases for more than two decades [20].

This work investigates the utility of graph mining in the context of predictive toxicology and proposes two new algorithms for the calculation of subgraph descriptors from chemical structure graphs. The descriptors may serve to obtain a mathematical model, describing the relationships between them and the toxicological activity values. Such models are referred to as *Quantitative Structure Activity Relationships (QSAR)*. The most general mathematical form of a QSAR model is:

$$\text{Activity} = f(\text{structural descriptors and/or physico-chemical properties}) \quad (1.1)$$

QSAR models can be classified as either statistical, or expert/rule-based approaches [47, 48]. Statistical approaches use general toxic endpoints and activity values gathered for a wide range of structures and are primarily driven by information inherently present in the data, not from human expert knowledge. Expert/rule-based approaches build QSAR generalizations from individual chemicals to chemical classes based on prior knowledge, heuristics, expert judgement and

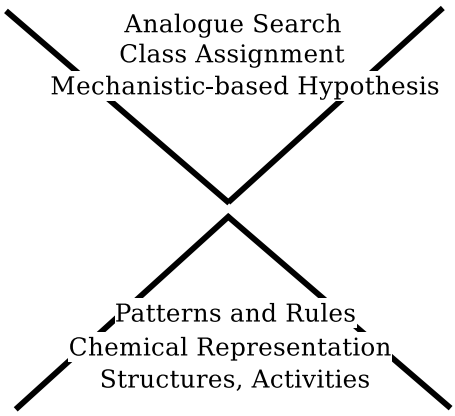


FIGURE 1.1: Top-Down *vs* Bottom-Up Approaches in Predictive Toxicology.

chemical and biological mechanism considerations (see Fig. 1.1). The techniques proposed here clearly fall in the statistical category.

1.1 Molecular Modelling/Systems Biology

The primary approach when trying to understand biochemical processes is to develop mathematical models of the involved mechanisms (e.g. receptor interactions, signalling pathways). Since there is a great variety of modes of action, this is only suitable for single toxicological processes and only for simpler endpoints. Current molecular modelling and systems biology approaches are limited to relatively simple effects (modelling from first principles). A complete modelling of complex toxicological effects is at present impossible. Fortunately, for most applications, including predictive toxicology, a complete model is seldom required. Instead, the current state of the art includes data-driven, statistical models, putting emphasis on evidence from the data.

1.2 Expert Systems

Expert knowledge is frequently used in predictive toxicology. Commercial systems provide predictions for specific toxicological endpoints, for instance carcinogenicity and mutagenicity. They build QSAR generalizations from individual chemicals to chemical classes based on prior knowledge, heuristics, expert judgement and chemical and biological mechanism considerations and are primarily geared towards detection of actives (toxic molecules). Therefore, positive test results are

often accepted as evidence for the toxic activity, whereas a negative result is not regarded valid to rule out risks.

Expert systems have a long-standing tradition and are still in frequent use by the industry and regulatory agencies. A prominent example is `DEREK`, a computer program sold by Lhasa Ltd [1]. Such systems make knowledge from their databases explicit, for example in the form of reports: A user who inputs a chemical structure is provided with a set of predictions about the chemical's toxic behaviour and the underlying rationale for each prediction, with references to literature. Thus, the system aggregates expert knowledge and rationally combines the evidence (logic of argumentation). Its databases are constantly updated and extended, and, in the case of `DEREK`, customers can participate in this process.

In systematic assessments of predictive power, such as the *Predictive Toxicology Evaluation* [58], however, expert systems have been performing rather badly, compared to statistical models. A reason for their remarkable spread despite this crucial deficiency may be that their logic closely mimics the line of argumentation of chemical experts, which may provide an intuitive familiarity and seeming plausibility.

1.3 Traditional QSAR models

Traditional QSAR methods use linear regression to identify a relationship between chemical descriptors and experimental activities. They rely on the idea that structural properties contribute in a linearly additive way to activity. Usually, critical molar concentrations C are modeled. The classical approaches are:

Hansch-analysis Physico-chemical properties are used as descriptor values. The formula is:

$$log(1/C) = a\ logP + b\ logP^2 + c\ E + d\ S + e,$$

where $logP$ is the octanol-water partition coefficient, describing the ability of the agent to reach the target site, and E and S are an electronic and a steric term, respectively. Electronic properties relate to binding ability and steric properties describe the bulk and shape of the compound. Descriptor values can be drawn from literature or calculated by computer programs. Relatively few descriptors are needed and they can be interpreted in biochemical terms.

[1] See `http://www.lhasalimited.org/`

Free-Wilson-analysis Structural descriptors are used in a group contribution approach (substituents). The formula is:

$$log(1/C) = \sum a_i x_i + \mu,$$

where x_i denotes the presence of group i (0 or 1) and μ the contribution of the unsubstituted compound. Predictions can only be made for substituents already included in the training set. Therefore, a large number of compounds is needed which yield a large number of descriptors.

The interpretation of linear QSAR models is done rather straightforward by inspecting the most important descriptors (i.e. the ones with high coefficients). The applicability of such models is restricted to *congeneric* series (i.e. compounds that exhibit a toxicological effect by a similar mechanism). Another problem with traditional QSAR techniques is the selection of descriptors for endpoints that are very complex and incorporate many different and potentially unknown biological mechanisms. In this case it is very likely to miss important descriptors or to suffer from the "curse of dimensionality", if too many descriptors are selected.

1.4 Data Mining Methods

The need to accelerate the toxicological assessment of chemicals and the prospect of using fewer lab animals and less expensive methods of analysis have spurred the development of predictive toxicology, and especially the structure-based approaches. Not surprisingly, by providing automated methods for the extraction of relevant knowledge from large and diverse databases, data mining methods have become important tools for predictive toxicology. An additional goal is to provide a certain degree of independence from human expert judgement by building on statistical criteria instead.

Data mining is often applied to derive representations of data which serve in a subsequent step as input to learning algorithms (feature mining). Feature mining derives descriptors from large sets of diverse, previously unseen data. Such techniques are the focus of this work.

The first feature mining method applied to chemical data was `WARMR` [34], which mined chemical fragments of size up to three atoms from a database of compounds, represented as 2-D structure graphs. Interest in feature mining started to increase, and several algorithms with extended capabilities were published. `MOLFEA` was able

to mine linear fragments comprising up to 20 atoms, and, for the first time, the resulting features were directly employed in QSAR modeling [36], employing the `WEKA` [16] implementation of support vector machines. Also, the first general graph mining algorithms – without a focus on chemical data – were published at that time, such as `AGM` [30], and `FSG` [37] (see also chapter 2).

Nowadays, there is a variety of implementations of well-known statistical learning algorithms for classification and regression available "off the shelf" [15]. Thus, through the combination of feature mining and statistical learning, creating a QSAR model has become a much more modular process compared to earlier times, with the steps for descriptor calculation, descriptor selection, and model learning being largely de-coupled [5]. Frequently, regularities and patterns are extracted from the data, represented in a unified format and fed into a machine learning algorithm (see chapter 3).

The process of building a QSAR model has also become much more flexible: Data mining methods allow for the combination of different types of descriptors. Whereas descriptors were traditionally delivered with the compounds in the form of tables, many can now be calculated from the molecular structure. Non-congeneric datasets are now useful as training data for learning algorithms. Data of various type can be combined and re-used, e.g. from different sources, or selected according to different criteria. Many machine learning methods are also able to deal with noisy or missing data. Moreover, much larger datasets may be processed with data mining methods, since they require no or little human intervention.

Of course, flexibility comes at a price: Data mining methods are designed to find non-trivial and previously unknown regularities within complex data. Complex data is inherently hard to represent, and the success of data mining methods crucially depend on adequate representations of the data. A molecule can, for example, be represented as 2-D graph, as 3-D graph, or by its physico-chemical parameters. Focusing on graph representations, again multiple representations are conceivable in a computer's memory: for example as adjacency list, or as a relational construct. This again restricts the ways for reasoning about the data.

Models can also fail for other reasons. Consider for example the traditional Free-Wilson model from the previous section: Since it can only model linear dependencies, it will fail if important relationships in the data are non-linear. On the other hand, non-linear models are able to fit more diverse data structures (in fact many can fit arbitrary data), but they are more likely to fit intricacies of the training data as well, due to their expressiveness, rather than the general regularities in the data. Learning such models usually takes a long time and they often show poor

generalization capability when applied to unseen data (*overfitting*). Of course, there are also other, more general, obstacles for building QSAR models, such as the data not being representative for the problem at hand.

These issues can be viewed from many different angles, accordingly the approaches to solve them vary greatly. This work addresses feature mining, as it provides the basis for data representations. Among the methods that derive symbolic representations of the data, it gives special attention to substructural features: In this setting, chemical compounds are represented as graphs, and relevant subgraphs are extracted from the set of graphs to form a description that conveys basically the presence or absence of any of the subgraphs (*chemical fingerprints* [23]). Since most physico-chemical properties, such as molecular weight or solubility, can be calculated from the molecular structure, it seems natural to directly use the structure for modeling. Structural representations also have intuitive and interpretable appeal, and may be useful for predictive, as well as explanatory purposes.

1.5 Thesis

In view of the adverse properties of redundant, high-dimensional representations for data mining in general, and in particular for learning algorithms, two graph mining algorithms that compress the search space of subgraphs are proposed in this work. Its hypothesis can be stated as follows:
Using the proposed graph mining algorithms, it is possible to efficiently extract a sparse selection of subgraph descriptors (molecular fragments) that compactly and diversely summarize possibly large databases of graphs (molecules). When used as input to learning algorithms, theses descriptors yield QSAR models with predictive power at least as good as the complete set of subgraphs from which they were selected or derived, and perform on par with or even better than highly optimized physico-chemical descriptors for complex biological endpoints.

1.6 Chapter Overview

The following summarizes the contents of the individual chapters of this work:

1.6.1 Chapters 2 and 3

Some basics of inductive databases, queries, and constraints are introduced. The special case of graph mining is explained, and the difference to item set mining is emphasized. This work considers connected subgraphs (fragments) of the 2-D structure graphs of chemical compounds. Search space representation, canonical enumeration of subgraphs and computational costs of graph mining are discussed. The stepwise approach to model building includes graph mining and feature selection as consecutive, distinct pre-processing steps. However, a more integrated approach to feature selection and data compression is proposed in this work, involving structural and statistical constraints to be integrated in the graph mining steps. A case study demonstrates the influence such constraints can have on model building.

1.6.2 Chapter 4

Common notations, datasets, and material relevant for subsequent chapters are presented. This avoids duplication, and allows for a uniform notation in the following sections about algorithms. This includes names and conventions for subgraph descriptors, molecular graphs, and datasets used in the experiments.

1.6.3 Chapter 5

Backbone Refinement Class Mining (BBRC), the first main algorithm of this work, is introduced. The chapter motivates the approach with the necessity for feature set compression and a review of related work. First, an intuitive account of the BBRC approach, combining frequency, structural, and statistical constraints, is given, then a formal representation is worked out. The compression potential of BBRC descriptors is theoretically derived and proven. Empirical results confirm the compression results in practice, while retaining good database coverage. Moreover, in the experiments, the structural constraints produce structurally diverse features with low co-occurrence rates, as shown with a dedicated visualization

method. BBRC descriptors compare favorably to other compressed representations in the context of classification models.

1.6.4 Chapter 6

Latent Structure Pattern Mining (LAST-PM), the second main algorithm, is introduced. LAST-PM combines related subgraphs into a weighted edge graph and mines elaborate patterns from this graph. More specifically, the process generates ambiguities (fragments containing optional parts). Heavy components are extracted from the weighted edge graph by spectral analysis, which yields a tightly condensed representation of the dataset. The results are expressed in a chemical fragment query language that preserves the ambiguities. Existing approaches for generalized subgraph mining are reviewed. The experiments involve classification models and comparison to other optimized representations, including BBRC. A concrete example shows how a chemical expert may interpret a pattern found by LAST-PM.

1.6.5 Chapter 7

This chapter illustrates how BBRC and LAST-PM may be applied in practice to a chemical dataset with a complex biological endpoint (human intestinal absorption). Besides descriptor type, pattern instantiation with binary or frequency information, as well as the type of learning algorithm (SVM or nearest-neighbor prediction), are validated.

1.6.6 Chapter 8

The results of all previous chapters are summarized. Consequences for feature mining algorithms regarding running time, compactness, and interpretability of descriptor sets, are described. Conclusions about strategies in graph mining applied to chemical databases are drawn and possible future work is outlined.

1.7 Contributions

The main contributions of this work are described as follows:

- BBRC and LAST-PM, two new algorithmic approaches to mining compact sets of descriptors in the search space of chemical structure graphs, are proposed. Conceptionally, the former creates a sparse selection from the search space of frequent and significant subtrees, based on structural and statistical constraints, while the latter "fuses" structurally and statistically related patterns. Therefore, they combine feature generation and feature selection into one step.

- Both approaches are scalable, since pattern set compression is achieved by combining frequency, statistical, *and* structural (graph-intrinsic) constraints, a novelty in graph mining. BBRC enables large-scale mining through a dynamic pruning criterion, such that tens and hundreds of thousands of chemical compounds can be processed in minutes. For LAST-PM, there is a positive tradeoff between compression and runtime. Expensive post-processing of the features is avoided.

- Detailed analysis shows that BBRC descriptors are structurally diverse, and thus cover the structural space well. A lot of descriptors per compound are produced, despite their absolute number being low, which is confirmed in low co-occurrence rates. The set of all frequent and statistically significant subtrees does much worse, despite containing much more descriptors. This is attributed to the joint effect of convex statistical and structural selection criteria employed in BBRC.

- LAST-PM descriptors are able to express ambiguities, i.e., any descriptor may contain optional parts of variable size. These ambiguities are found by structurally "aligning" subtrees and resolving conflicts by logical OR. This process obviously elicits latent (or hidden) motifs that are available from such a structurally organized view on several subtrees at once.

- In classification tasks with either nearest-neighbor or SVM (support vector machine) models, the accuracy of (models based on) BBRC descriptors is on par with the complete set of frequent and significant subtrees, but significantly better than that of other compressed representations. LAST-PM descriptors perform even significantly better than the complete set from which they were derived. They also outperform BBRC descriptors and highly

optimized physico-chemical descriptor models from the literature in the classification of compounds for complex biological endpoints.

The next chapter describes the graph mining problem, which is foundational to feature mining on chemical structure graphs.

Chapter 2

Tree and Graph Mining

2.1 Inductive Databases

An inductive database [28] is a database that can be queried for patterns within the data that fulfill complex queries. The queries are formulated in a domain-independent query language [11]. Thus, a general constraint-based mining problem on an inductive database comprises the following components:

- A database $\mathbf{r}$, constituted by a set of *instances*,
- An (usually infinite) hypothesis space of *patterns* $\mathcal{L}$, defined on a finite alphabet Σ
- A *constraint* $q(\theta, \mathbf{r})$ containing specific criteria that a pattern θ should meet with respect to the database.

The task is to find a *theory of the data*, i.e. the set

$$Th(\mathbf{r}, \mathcal{L}, q) = \{\theta \in \mathcal{L} \mid q(\theta, \mathbf{r}) = true\}, \tag{2.1}$$

referred to as the theory of q with regard to $\mathbf{r}$, where θ is called a *pattern* [40]. This learning setting can be generalized straightforwardly to more than one constraint by intersecting the associated individual theories. Although such a formulation is generic enough to be understood as a quite general learning problem, this work addresses the search for discrete patterns within finite sets of discrete instances. Therefore, also theories will be finite sets (often referred to as *solution sets* or *pattern sets*). With different choices of $\mathbf{r}$, $\mathcal{L}$ and q, different mining problems are obtained.

2.1.1 Types of Constraints

All patterns treated here obey a partial order, the so-called *specialization relation*, denoted by "$\preceq$." A pattern θ_1 is said to be more specific than (or a refinement of) pattern θ_2, if $\theta_2 \preceq \theta_1$. We also say that θ_2 is more general than θ_1.

The specialization relation "$\preceq$" gives rise to the notion of *monotonic* and *anti-monotonic* constraints as follows:

Monotonic Constraints: A constraint is called monotonic, if any specialization of a pattern that satisfies the constraint also satisfies the constraint, i.e. for all $\mathbf{r}$ and θ_2 it holds that if $q(\theta_2, \mathbf{r})$ and $\theta_2 \preceq \theta_1$, then $q(\theta_1, \mathbf{r})$.

Anti-Monotonic Constraints: A constraint is called anti-monotonic, if any generalization of a pattern that satisfies the constraint also satisfies the constraint, i.e. for all $\mathbf{r}$ and θ_1 it holds that if $q(\theta_1, \mathbf{r})$ and $\theta_2 \preceq \theta_1$, then $q(\theta_2, \mathbf{r})$.

We consider also:

Convex Constraints: A constraint is called convex, if there is a convex function f that maps any pattern p to a real value and there is a user-defined threshold $u \in \mathbb{R}$, such that if $f(p) > u$ then p fulfills the constraint. For convex functions, anti-monotonic bounds can be derived. Convex constraints are special cases of so-called *boundable constraints*.

Succinct Constraints: A constraint is called succinct, if it can be "pushed" into the dataset, i.e. patterns matching succinct constraints can be found by changing the way the dataset is represented.

2.1.2 Hypothesis Spaces

Monotonic and anti-monotonic constraints are transitive relations, thus they define absolute borders in the search space beyond which those constraints cannot be fulfilled. Conversely, they are guaranteed to be fulfilled on the other side of those borders. Therefore, each such constraint yields a half space where all patterns fulfill the constraint. This applies only to monotonic and anti-monotonic constraints, and results in the intersection of the respective half spaces as the combined theory.

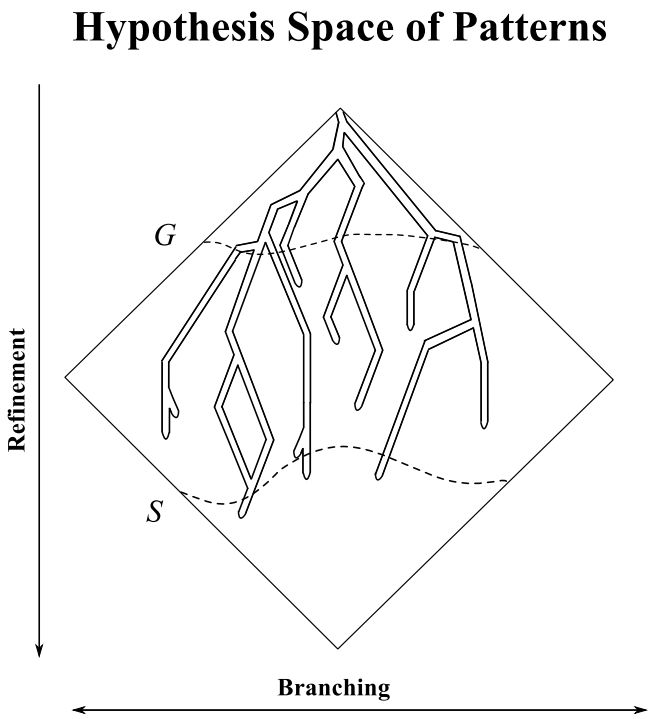


FIGURE 2.1: Schematic depiction of the hypothesis space with S- and G-border.

The border implied by a conjunction of anti-monotonic constraints is denoted by S. Specifically, it consists of all patterns θ, such that all generalizations of θ are in the theory and none of the specializations of θ is in the theory. Analogously, the border implied by a conjunction of monotonic constraints is denoted by G. It consists of all patterns θ, such that all specializations of θ are in the theory and none of the generalizations of θ is in the theory. Those elements of S that are in the theory are called *the positive border* of the theory, a concept introduced by Mannila and Toivonen [40]. Fig. 2.1 gives a schematic depiction.

2.2 Graph Mining

Databases that consist of graphs induce a very interesting and complex mining problem due to the structured nature of both patterns and instances. This section introduces graph mining as compared to itemset mining and presents some challenges in the domain of graphs, specifically on how to systematically and efficiently enumerate all patterns of interest, as well as some approaches to solve them. First, the domain of graphs is introduced and the necessary concepts are defined. Then, frequent pattern mining is exemplified for itemsets and graphs.

2.2.1 Basic Graph Theory

We assume a graph database $R = (\mathbf{r}, \Sigma, a)$, where $\mathbf{r}$ is a set of graphs, $\Sigma \neq \varnothing$ is a totally ordered set of labels and $a : \mathbf{r} \to \{0, 1\}$ is a function that assigns a truth value to every graph in the database (binary classification). Graphs with the same classification are collectively referred to as *target classes*. Every graph $r \in \mathbf{r}$ is a *labeled, undirected graph*, i.e. a tuple $r = (V, E, \Sigma, \lambda)$, where $V \neq \varnothing$ is a finite set of nodes and $E \subseteq V = \{\{v_1, v_2\} \in \{V \times V\}, v_1 \neq v_2\}$ is a set of edges and $\lambda : V \cup E \to \Sigma$ is a label function. The set of all labeled, undirected graphs is referred to as G[1].

An *alignment* of a graph r is a bijection $\phi_r : (V, E) \to P$, where P is a set of distinct, partially ordered, identifiers of size $n = |V| + |E|$, such as natural numbers. Thus, the alignment function applies to both nodes and edges.

An ordered set of nodes $\{v_1, \ldots, v_m\}$ is a *path* between v_1 and v_m, if $\{v_i, v_{i+1}\} \in E, i \in \{1, \ldots, m-1\}$ and $v_i \neq v_j$ for all $i, j \in \{1, \ldots, m\}$. The *length of a path* is defined as the number of edges it contains. We only consider *connected graphs* here, i.e. there is a path between each two nodes in the graph. The graph $r = (V, E, \Sigma, l)$ is *double connected* or *cyclic*, if there exist two distinct paths between a pair of nodes $\{v_i, v_j\} \in E$.

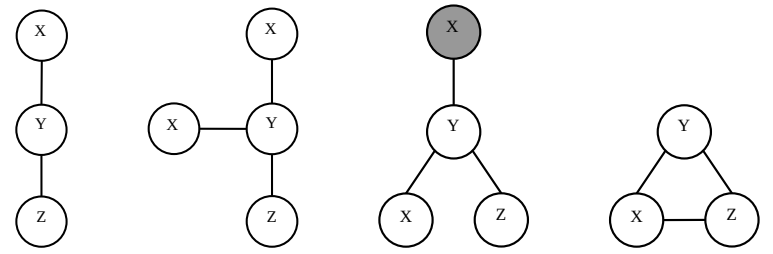
Assuming a graph with n nodes, then the graph is a *tree* if it has no more than $n-1$ edges (due to the requirement of connected graphs, it must also have at least $n-1$ edges). Being a tree is equivalent to not being double connected. Obviously, any path is a tree, but not *vice versa*. A *rooted tree* is an ordinary tree with a designated root node. The depth of a node in a rooted tree is the length of the path from it to the root (there can only be one such path). In an ordered tree, two nodes at the same depth are called *siblings*. If a tree has no root, it is sometimes also called *free tree* to emphasize the difference.

A *node refinement* is an addition of an edge and a node to a graph r. Given r has at least two edges, a *branch* is a node refinement that extends r at a node adjacent to at least two edges.

A *subgraph* $r' = (V', E', \Sigma', \lambda')$ of r is a graph such that

- $V' \subseteq V$ and $E' \subseteq E$ with $V' \neq \varnothing$ and $E' \neq \varnothing$,
- r' is connected, and

[1] In figures showing graphs, nodes and edges are sometimes identified with their labels.



(a) Path (b) (Free) Tree (c) Rooted Tree (d) Cyclic Graph

FIGURE 2.2: Some Graphs

- $\lambda'(v_1) = \lambda(v_2)$ whenever $v_1 = v_2$, and $\lambda'(e_1) = \lambda(e_2)$ whenever $e_1 = e_2$ for all $v_1 \in V'$, $v_2 \in V$, $e_2 \in E'$, $e_2 \in E$.

We say that r' is *subgraph-isomorphic* to r. This work considers free, connected, edge-induced subgraphs (and thus also subtrees) with those properties. Other notions of subtrees include *bottom-up* subtrees and *embedded* subtrees, see the overview by Chi *et al.* [7].

We denote the subgraph relation by $\sqsubseteq$. If, for any two graphs r, r', $r' \sqsubseteq r$, then r' is said to *cover* r. The subgraph relation induces a partial order on graphs, as required in section 2.1.1: For any graphs r, r', s,

$$r' \preceq r \quad \Leftrightarrow \quad r \sqsubseteq s \Rightarrow r' \sqsubseteq s. \tag{2.2}$$

Example 2.1. *Fig. 2.2 shows some graphs. Graphs 2.2 (b) and 2.2 (c) are subgraphs of each other, thus they are subgraph-isomorphic to each other. Graph 2.2 (a) is a subgraph of 2.2 (a), 2.2 (b), 2.2 (c), and 2.2 (d).*

The subset of $\mathbf{r}$ that a graph r covers is referred to as the *occurrences* of r, denoted by $occ(r, \mathbf{r})$, its size as *support* of r in $\mathbf{r}$, denoted by $supp(r, \mathbf{r})$. The graph r is called *open* (*closed*), if for any graph s with $occ(r, \mathbf{r}) = occ(s, \mathbf{r})$ it holds that $r \sqsubseteq s$ ($r \sqsupseteq s$).

Graphs are partially ordered as follows: Let $P \subseteq G$ be the set of paths and let $T \subseteq G$ be the set of trees. It holds that

$$P \subset T \subseteq G. \tag{2.3}$$

2.2.2 Frequent Item Set Mining

Frequent pattern mining is introduced with itemsets: Let the instances of $\mathbf{r}$ be sets, and let $\mathcal{L}$ be the hypothesis space of sets. Then, if q is the minimum frequency constraint:

$$q(\theta, \mathbf{r}) \Leftrightarrow |\{d \in \mathbf{r} \,|\, \theta \subseteq d\}| \geq minsup, \tag{2.4}$$

where *minsup* is the user-defined minimum frequency, then the frequent item set mining problem is obtained.

Example 2.2. *Let the alphabet be* $\Sigma = \{\bigcirc, \square, \triangle\}$.

- $D1 = \{\bigcirc, \square\}$
- $D2 = \{\bigcirc, \triangle\}$
- $D3 = \{\bigcirc, \square, \triangle\}$
- $\mathbf{r} = \{D1, D2, D3\}$

Let $minsup = 2$*, then* $Th(\mathbf{r}, \mathcal{L}, q) = \{\{\bigcirc\}, \{\square\}, \{\triangle\}, \{\bigcirc, \square\}, \{\bigcirc, \triangle\}\}$.
Let $minsup = 3$*, then* $Th(\mathbf{r}, \mathcal{L}, q) = \{\{\bigcirc\}\}$.

Due to the anti-monotonicity of the minimum frequency constraint, the theory for a given value of *minsup* can never be larger than the theory for any $minsup'$, such that $minsup' < minsup$. Moreover, the former is always a subset of the latter.

2.2.3 Frequent Subgraph Mining

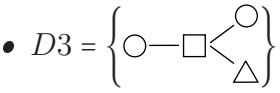
Now, frequent pattern mining is applied to graphs: Let the instances of $\mathbf{r}$ be (connected) graphs, and let $\mathcal{L}$ be the hypothesis space of (connected) graphs. Then, if q is a minimum frequency constraint:

$$q(\theta, \mathbf{r}) \Leftrightarrow |\{d \in \mathbf{r} \,|\, \theta \sqsubseteq d\}| \geq minsup, \tag{2.5}$$

then the frequent subgraph mining problem is obtained (note the subgraph relation "$\sqsubseteq$" instead of the subset relation in Equation 2.5).

Example 2.3. *Let the alphabet be again* $\Sigma = \{\bigcirc, \square, \triangle\}$.

- $D1 = \{\bigcirc\!\!-\!\!\square\}$

- $D2$ = {○—△}
- $D3$ = 
- $\mathbf{r}$ = $\{D1, D2, D3\}$

Let $minsup = 2$*, then* $Th(\mathbf{r}, \mathcal{L}, q)$ = {○, □, △, ○—□}.
Let $minsup = 3$*, then* $Th(\mathbf{r}, \mathcal{L}, q)$ = {○}.

Note the additional constraints that the subgraph mining problem imposes by requiring subgraphs to be connected, i.e. ○—△ is not in the theory for $minsup = 2$ (same of course for $minsup = 3$). Also note that ○—□ has a support of only 2, although it occurs 3 times in total. This is because it occurs two times in $D3$ but multiple occurrences (occurrences in the same instance) are ignored. However, the subset relation of theories for growing m carries over from frequent itemset mining.

2.2.4 Canonical Graph Representations

Itemset mining is essentially structureless: A set is simply a collection without inherent organization. For example, it is easy to test two given sets for equality (by testing if they are subsets of each other). The analogous situation for graphs is much more difficult: Here, the test for equivalence is known to involve an NP-hard problem, namely subgraph-isomorphism testing. This section discusses the importance of this test for frequent subgraph mining and reviews the closely related task of *canonical enumeration* of subgraphs, i.e. how to find all (interesting) subgraphs without considering any subgraph twice, which is an important concept in the implementation of efficient graph mining algorithms.

2.2.4.1 Pattern Matching Operators

Actually, there are well-known algorithms for subgraph-isomorphism testing, but all of them require exponential time. More specifically, it has been shown that subgraph-isomorphism testing is NP hard [10]. Thus, given a pair of arbitrary graphs in G, there has no algorithm yet been found that tests whether one is a subgraph of the other in polynomial time. If no such algorithm exists (which is unknown), an exact solution is not generally feasible.

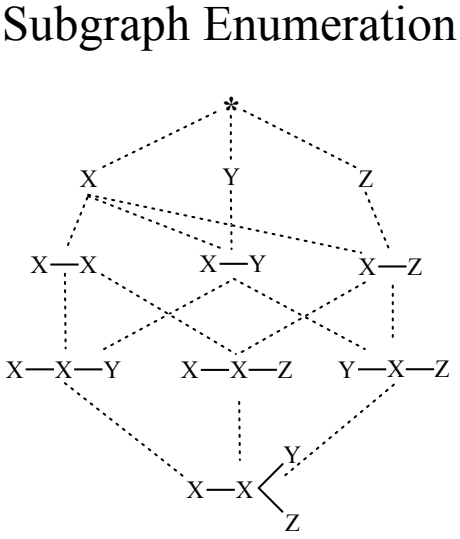


FIGURE 2.3: Example search space of subgraphs. Join operations are indicated by dashed edges.

The efficiency of subgraph isomorphism testing seems essential for graph mining, because such a *pattern matching operator* allows to check for redundant graphs in the result set and prune them[2]. However, databases composed of *graphs of bounded treewidth* (a proper subset of G containing cyclic graphs) allow for efficient (in polynomial time) frequent connected subgraph mining, by using a specialized matching operator, while the general pattern matching operator remains NP hard [24]. Similar results have been obtained for another cyclic subset of G, the class of *outerplanar graphs* [25]. For databases composed of non-cyclic graphs, such as T, efficient graph mining algorithms and pattern matching operators do also exist. For a comprehensive introduction to this setting see the review by Chi *et al.* [7].

Thus, there is no direct relationship between general pattern matching and graph mining. Nevertheless, no efficient algorithm for mining all frequent connected subgraphs from a database of arbitrary graphs has been identified. Moreover, this problem cannot be easier to solve than subgraph isomorphism: construct a graph database from the graph r and the subgraph candidate r'. Consider frequent subgraph mining with $minsup = 2$. The presence of a subgraph with the same number of nodes and edges as r' in the theory decides subgraph isomorphism.

2.2.4.2 Systematic Subgraph Enumeration

A naïve approach to frequent subgraph mining would be to (a) enumerate all possible refinements of a candidate subgraph, and (b) check whether the generated subgraphs satisfy the constraints, then iterate the process using each generated subgraph in turn as candidate until all subgraphs have been generated [65]. In a postprocessing step (or already during the search) filter out duplicate subgraphs by testing for subgraph isomorphism against the already generated subgraphs. Given the efficiency of subgraph isomorphism testing just discussed, it seems clear that the naïve approach must fail in practice.

As an example, Fig. 2.3 shows the search space of subgraphs for the graph given at the bottom. Obviously, the naïve approach can be enhanced in several ways (non-exhaustive list):

- Instead of generating all possible refinements from each graph, preferably *join* subgraphs already generated on the same level to form a new subgraph on the next level. If a subgraph does not occur (frequently), such as all graphs containing `Y-Z` in Fig. 2.3, then, apart from the most general one (which is `Y-Z` here), none of them will be generated further down in the search space. Conceptually, all subgraphs can be generated using joins, but this requires breadth-first search across the levels.
- Often, it is easy to avoid duplicate subgraphs by adopting a canonical representation for subgraphs. For example, consider the case of paths and specifically `X-Y` *vs* `Y-X`, which are subgraph-isomorphic to each other. By only creating lexicographically ordered sequences, `Y-X` would not have been generated, since it is not in order, as shown in Fig. 2.3.

The next section discusses canonical representations for trees, a tool that allows to implement both optimizations (and more) in a graph mining algorithm. In the context of systematic subgraph enumeration, the partial order that the subgraph relationship induces is often referred to as *refinement relation*.

In particular, the class of algorithms emerging from implementing the above optimizations recursively (a) refines new candidate patterns from existing patterns, and (b) checks if the candidates fulfill all constraints. Since the search process is guided by refining patterns and bound by anti-monotonic constraints, such as

[2]In the case of itemset mining this test is indeed efficiently performed by subset testing.

minimum frequency, this class of algorithms falls into the category of *branch-and-bound* algorithms.

2.2.4.3 Depth Sequences for Rooted Trees

In his study, Zaki [69] treats the enumeration of subtrees occurring in a database of rooted trees, but in a specialized setting compared to our definition (see section 2.2.1): the database trees and the subtrees are *ordered*, i.e. given two sibling nodes, the order that they appear in at the same depth matters. In the paper, he adopts an adjacency list representation for rooted trees, where the lists are encoded as strings. The adjacency list takes the form of a *depth-first search* (DFS) run through a rooted tree. The encoding maintains the invariant that, for any refinement of a rooted tree, the tree's *DFS code* is always a prefix of the refinement's DFS code[3].

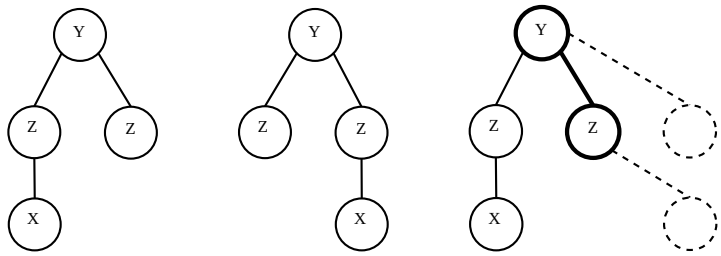
The paper by Nijssen and Kok [43] treats the enumeration of unordered rooted subtrees by generalizing Zaki's approach to databases of unordered rooted trees. They assign any unordered rooted tree to a specific ordered rooted tree, namely the equivalent one in so-called *ordered normal form*, thereby reducing the problem to the setting treated by Zaki before.

We will elaborate on this approach, starting with paths: Let $p = \{v_1, \ldots, v_m\} \in P$ be a path, then its sequence is defined as the string $\lambda(v_1) \ \ldots \ \lambda(v_m)$, obtained by concatenating node labels along the path (edge labels neglected here for brevity). Obviously, either the sequence or its reverse yields a lexicographically lower sequence. On the other hand, no other sequence can possibly represent the same path. Therefore, by convention, the lexicographically lower sequence uniquely identifies the path, inducing a total order on paths.

The concept of representing a structure canonically by adopting a linear string representation can be generalized from ordered to unordered rooted trees. In contrast to a path sequence, however, each node is augmented with depth information. The procedure works "bottom-up", starting at the nodes:

1. Assume a total order on the node labels. Together with node depth, this induces
2. A total order for so-called *depth tuples* (conjunction of node label and node depth), which in turn induces

[3]The term *DFS code* was coined by Yan and Han [66] for the `gSpan` algorithm. It is used here more generally to denote any canonical encodings of DFS runs.



(a) Tree #1 (b) Tree #2 (c) Tree #3

FIGURE 2.4: Ordered rooted trees and rightmost path refinements.

3. A total order on DFS codes, obtained by concatenating depth tuples in the order the nodes are visited during a depth first run through a rooted tree.

For any rooted tree, its lexicographically lowest DFS code (*Minimum DFS code*) is selected as its *ordered normal form*. The ordered normal form induces a total order on rooted trees and is sufficient (correct and complete) to enumerate all possible rooted trees.

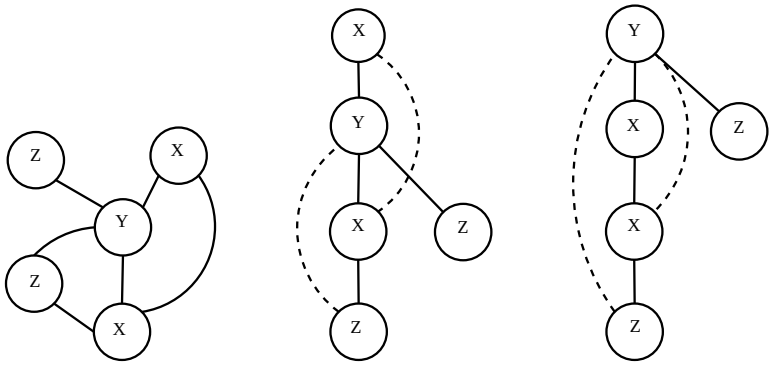
Example 2.4. *In Fig. 2.4, Tree #1 has DFS code* `(0,Y)(1,Z)(2,X)(1,Z)`*, while Tree #2 has DFS code* `(0,Y)(1,Z)(1,Z)(2,X)`*. Both rooted trees are subgraph-isomorphic to each other, but only Tree #1 is in ordered normal form, since depth tuple* `(2,X)<(1,Z)` *in the formalism by Nijssen and Kok.*

In order to obtain all possible refinements of a rooted tree, it is sufficient to enumerate only refinements growing from a node on the rightmost path in this rooted tree, as Nijssen and Kok show in their paper. Refining a rooted tree corresponds exactly to appending an appropriate suffix to the rooted tree's ordered normal form. Any prefix of a rooted tree's ordered normal form is the ordered normal form of some subtree of the tree. In Fig. 2.4, only the dashed refinements of Tree #1 are possible in Tree #3. All other configurations would contradict the minimality of the DFS code.

2.2.4.4 Algorithms for Mining General Graphs

The idea of using DFS codes was most generically captured in the theory underlying the `gSpan` algorithm, published in 2002 by Yan and Han [66], because it defines a code for the class of subgraphs in databases of general graphs in G.

Example 2.5. *Fig. 2.5 shows a cyclic graph, together with two possible representations, called* DFS trees *(slightly abridged from [66]). The edges found by the*



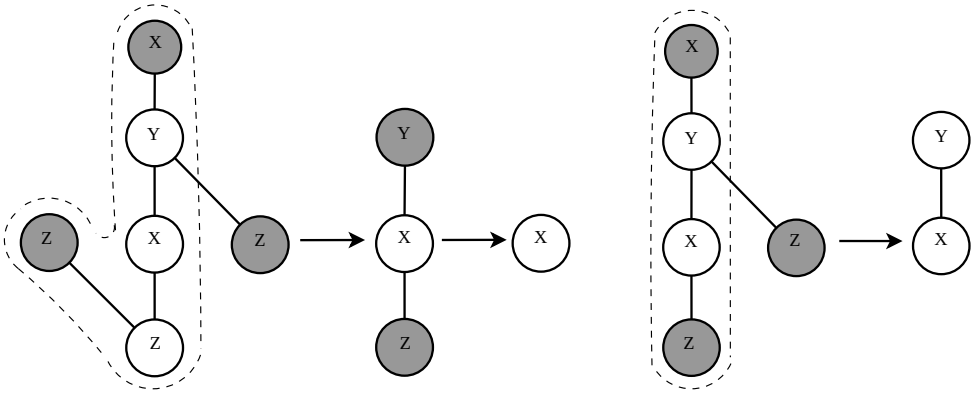
(a) Graph (b) DFS Tree #1 (c) DFS Tree #2

FIGURE 2.5: DFS Trees

DFS procedure are drawn normally (termed forward edges*), while the others appear dashed. They are always cycle closing edges (referred to as* backward edges*). As for rooted trees, DFS codes are totally ordered and subgraph-isomorphism is equivalent to prefix minimum DFS codes. The DFS code for the DFS tree in Fig. 2.5 (b) is lower than the DFS code for Fig. 2.5 (c), while both are equivalent to the graph in 2.5 (a).*

The `gSpan` algorithm does not fully exploit the subset relation $P \subset T \subseteq G$ of the different subgraph types (see section 2.2.1) while the `Gaston` algorithm, published by Nijssen and Kok in 2003 [44], uses a similar DFS code, but works in three different phases according to the subset relation. More specifically, the phases are kept strictly separated, which enables `Gaston` to make initially fast progress on the efficiently enumerable subgraph classes of paths and trees. Only after one class has been completely enumerated it starts to work on the next class.

A *spanning tree* of a cycle-closing graph g is a subgraph with the same number of nodes as g. Any such graph has a spanning tree. For example, consider the solidly drawn trees in Fig. 2.5 (b) and 2.5 (c) (the forward edges). They form spanning trees for their respective graphs. `Gaston` uses spanning trees to enumerate general subgraphs in the third phase. However, a similar concept is already applied to the second phase: `Gaston` employs longest paths (so-called *backbones*) to enumerate free trees. Therefore, backbones play a similar role for free trees as maximum spanning trees play for general subgraphs. Importantly, backbones partition the search space of free trees disjointly, since every free tree has exactly one backbone. A well-known property of free trees is that any free tree has center node or two nodes that form a bicenter. The center or bicenter is obtained by repeatedly removing leaf nodes (see Fig. 2.6, adopted from [7]). Whether a free tree is centered or bicentered is mediated by the length of the longest path within the



(a) Finding the Center (b) Finding the Bicenter

FIGURE 2.6: Finding Centers and Bicenters of free trees. Backbones are marked by dashes.

tree: being centered corresponds exactly to an even length of the longest path. The center (bicenter) is conceived as the root (the two roots) of a (two) rooted subtree(s), which enables the procedure described in section 2.2.4.3 for unordered rooted trees.

Among the longest paths in a free tree, Nijssen and Kok select the one with the lowest DFS code as the *backbone* of the free tree. To this end, they re-label nodes and edges to make the backbone appear always first in the canonical ordering for every tree. This allows `Gaston` to start with longest path enumeration in the first phase. The second phase uses each path (starting with the longest ones) as backbone and extends it to the maximum possible spanning tree.

2.2.5 Discussion

`Gaston` exploits specific properties of trees and graphs to structure the search for free trees in a more detailed fashion than `gSpan` does. Specifically, it distinguishes paths and real trees. For the latter, in case of a center, a free tree can be conceived simply as a rooted tree, with the root being the center. In case of a bicenter, the free tree can be conceived as two separate, rooted trees, each rooted at one of the bicenter's nodes, and connected by a single edge between the two root nodes. This effectively reduces the problem of enumerating free trees within databases of general graphs to enumerating rooted trees containing a specific path (the backbone).

Checking for canonical refinements in `Gaston`s representation has constant time complexity for the class of trees, as Nijssen and Kok show. Moreover, `Gaston`

uses a combination of joins and rightmost path extensions, in order to traverse the search space of subgraphs with a combination of breadth- and depth-first search. The procedure is more sophisticated than in `gSpan` in that it stores the embeddings of all refinements, which helps avoiding costly subgraph isomorphism checks [65] (see the `Gaston` article for details about the implementation [44]). The latter are computationally expensive, even for the canonically enumerable classes P and T, as explained in section 2.2.4.1.

2.2.6 Conclusions

Both frequent item set mining and frequent subgraph mining return sets. However, the former is concerned with sets of sets, whereas the latter returns sets of subgraphs. Subgraph mining implies stronger conditions through the requirement that subgraphs be connected. The more structured search space allows for partitioning the search into different phases, an important property which may serve as a basis for new perspectives on graph mining.

Since the search space in graph mining is more structured than for itemset mining, it implies higher demands to avoid redundant generation of patterns. To this end, canonical representations have been developed in the form of DFS codes. Although defined for general subgraphs, cyclic subgraphs cannot be efficiently enumerated using DFS code, in contrast to subtrees.

Chapter 3

From Patterns to Models

3.1 Introduction

Frequent subgraphs in graph databases encode patterns that are shared among many of the graphs, which makes them inherently informative. Frequent subgraphs have been used as patterns for a variety of learning tasks: for distance metrics [53] and clustering [57], as well as for classification or regression models. However, current methods for subgraph mining still suffer from scalability problems and problems with excessively large solution sets.

Most of the predominant approaches employ minimum frequency and possibly convex constraints such as χ^2 values [6, 18, 31, 36, 44, 66]. An increasing minimum frequency tends to favor a higher entropy with regard to target class distribution, whereas statistical constraints retrieve subgraphs that are correlated with the target classes. However, the thresholds used still lead to an explosion in the number of patterns found, which is a significant drawback of these approaches. As a result, the solution set size is usually multiple times larger than the original database, rendering it useless for many classification algorithms, and individual inspection by domain experts as well.

Whereas distance metrics and clustering methods mostly exploit the graph structure of the frequent subgraphs directly, many classification or regression algorithms take a more indirect approach, termed *propositionalization* [50]. In this setting, subgraphs (or general patterns) can be represented as bit vectors, indicating their presence or absence in the instances. In fact, this occurrence-based description makes any discrete patterns amenable to such machine learning algorithms [52].

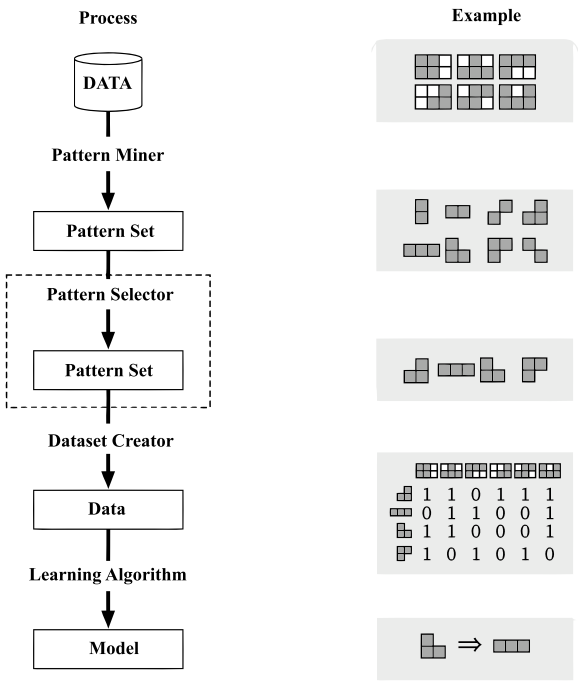


FIGURE 3.1: The Stepwise Approach to Model Building, adapted from Bringmann *et al.* [5].

To obtain a compressed propositionalization, researchers often constrain the solution set to open or closed frequent subgraphs (see section 2.2.1), yielding a significantly reduced number of patterns. This might however prune important structural information, since only the support is considered, which may not be sufficient to adequately summarize the diversity of subgraphs.

Fig. 3.1 shows a propositionalized workflow, where some patterns are generated by a mining algorithm and then represented in terms of occurrences in the database. Note that the patterns could be anything here: sets, graphs, or other descriptors. The important point of Fig. 3.1 is that patterns are merely represented by where they occur or not occur in the database once the pattern mining step is finished. To obtain a compressed representation, the workflow features a dedicated pattern selection step, where a pattern set most suitable for the learning problem at hand (more specifically, for the learning algorithm later on) is derived from the original pattern set. This means optimizing the bit vector representation such that patterns (columns) describe database instances (rows) as diverse (non-redundant) as possible. Importantly, no intrinsic information (graph structure) is used for compression directly here.

The following reviews techniques that have been proposed for pattern set compression[1]. Much effort has been dedicated to exploiting extrinsic properties (such as occurrences or correlation), but little work seems to have been done involving more domain dependent properties (such as graph structure) for that.

3.2 Pattern Set Compression

3.2.1 Motivation

The set of frequent subgraphs (denoted as the *result set* when referred to as result of a graph mining algorithm applied to a database of graphs) has to be compressed to be of actual use in most cases: after all, a graph with e edges may contain up to 2^e possibly disconnected subgraphs (*powerset* relation). Such large result sets may become intractable for all kinds of algorithms [8].

- The effort for mining an exponential amount of subgraphs scales unfavorably with database growth, no matter how efficient a graph mining algorithm works.
- It has been argued that an uncompressed result set would require post-processing due to redundancy of the patterns contained in it [7, 27, 52]. Experts would not be able to draw any conclusions from the vast amount of very similar subgraphs without such post-processing, since the useful patterns would be lost in the flood; similarly, the high-dimensional pattern space would prevent machine learning methods from obtaining meaningful models [2].
- Compressed representations save space, both on disk and in memory and allow for faster processing of the data [27]. On a positive note, the need for compression may turn into an advantage, since it can potentially prune parts of the search space and stop the search process early on [27].

[1]More examples are discussed in sections 5.5 and 5.7 in the context of experiments.

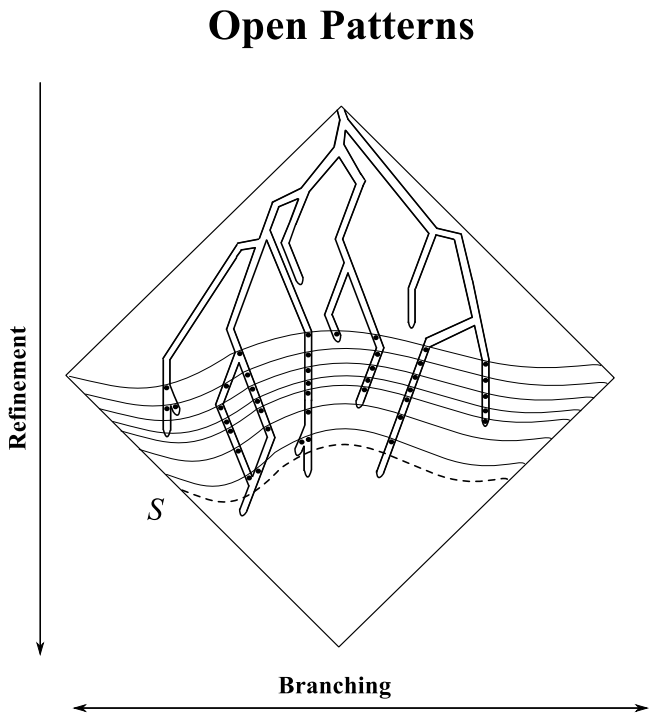


FIGURE 3.2: Open Patterns Search Space. Dashed lines indicates levels of frequency.

3.2.2 Related Work

Quite surprisingly, previously proposed compression methods for sets of subgraphs build mostly on the occurrences of subgraphs rather than directly on their structural composition. As such they are applicable to general pattern sets as well and graph-specific properties are left unexploited.

Among the "classical" occurrence-based methods are open [6] or closed [8, 67] frequent subgraphs or subtrees. They consist of the most general or most specific patterns of all the patterns with the same occurrences (see section 2.2.1). Note that, in the case of graphs, the notion of open/closed is not exactly the original meaning from algebra. The hypothesis space of open frequent patterns is depicted schematically in Fig. 3.2. A special case of closed frequent patterns are maximal frequent patterns, for which a lot of studies have been done [8, 27, 62].

Open and closed frequent patterns yield a lossless compression of the set of all frequent subgraphs [18], which is theoretically appealing. However, "lossless" is to be understood only in terms of distinct occurrences, i.e. no distinct pattern from the propositionalization is lost. In particular, the term does not apply in terms of structure when applied in the graph domain, i.e. a lot of very diverse frequent subgraphs are suppressed in the output. Subsequent treatment of the graph structures

will thus be greatly affected. Even in a purely propositional setting, many algorithms are affected by repeated patterns (for example k-nearest-neighbor methods employing a weighting based on common patterns). Thus they behave differently when receiving different amounts of equivalently propositionalized patterns. Moreover, the result set of open or closed patterns might still be enormous [18, 35] and do not guarantee a certain degree of compression.

To obtain representative subgraphs more efficiently, many approaches have traded completeness for smaller set sizes. Maximal frequent subgraphs encode frequent commonalities of maximal size between graphs [27], which is inherently interesting. Among the works employing maximal frequent subgraphs, sampling methods have dominated in recent years, imposing diversity [2] or top-k [39] constraints on the result set. However, they lose frequency information for the individual subgraphs, and thus this solution might be too coarse [18].

A very different approach, inspired by information theory, more specifically, by minimum description length, has been proposed by Cook and Holder [9]. They iteratively extract informative subgraphs from a graph database and replace the original occurrences, shrinking the database and creating a hierarchical representation from the extracted patterns. Another technique by Jeroen Knijf [35] tries to find patterns characteristic for a "master tree", assuming a forest (set of ordered trees) that represent embedded subtrees. It does so by overlaying the input trees according to minimum edit distance.

3.2.3 Discussion

It seems sensible to classify existing approaches to pattern set compression for subgraphs along two dimensions: systematic *vs* sampling methods and generic *vs* graph-specific methods.

Whereas sampling methods have been employing graph-specific properties as primary measures of interest and are especially targeted towards structural diversity, systematic (or even lossless) approaches (with the exception of Cook and Holder's and Knijf's work) have been leaning exclusively on generic properties, applicable to propositionalized and thus general pattern sets. One could argue that occurrences are determined by structure in the case when the patterns consist of subgraphs, which is true. However, it remains an indirect use of structure. Hence, the implementation of an explicitly structural summarization strategy (defined intensionally) should yield a higher compression ratio while still retaining all essential information.

3.3 Correlated Patterns

This section treats a special kind of pattern compression, namely the selection of patterns that are significantly correlated to the target values. Selecting correlated patterns from a set is a supervised process, and thus must be seen in direct context to model building. Thus, it is a more complex step than the compression strategies discussed so far, combining aspects of the *Pattern Selector* and the *Learning Algorithm* from Fig. 3.1.

Here, a classification function a is available, relating instances to target classes, but we will also allow the function a to assign numerical values to instances for the purposes of this chapter. We will first consider the special - but important - case where the target classes are binary (e.g. *active vs inactive*) and show how this setting integrates into branch-and-bound algorithms (see section 2.2). Generalizations to multinomial cases are straightforward, both in theory and implementation. Then, a case study using correlation to numerical target values is presented.

3.3.1 Statistical Metric Pruning

Since many correlation functions for categorical target values are convex [42], mining correlated patterns constitutes a convex constraint (see section 2.1.1). In practice, it is common to combine such convex constraints with at least one anti-monotonic constraint, such as minimum frequency. When combined with a branch-and-bound algorithm that generates patterns by extension of edges and nodes (as discussed in the last section), anti-monotonic constraints can be directly used to prune the search space and terminate the search process at the G-border. It is thus beneficial that convex constraints can be adapted to the branch-and-bound scenario. A possible integration is described in this section.

If each subgraph occurred in half of the graphs, a set of k subgraphs could in principle distinguish 2^k bins of graphs. Since a higher global minimum frequency raises the support of frequent patterns in at least one target class as well, it may leverage a pattern's discriminative potential for classification tasks.

Subgraphs that are correlated significantly with the target class can be filtered with statistical measures. Due to the absence of both monotonicity and anti-monotonicity, significance values cannot be used for anti-monotonic (nor monotonic) pruning directly, however, the convexity of the χ^2 function allows to derive a related measure for anti-monotonic pruning.

	θ	*all*
active	y	m
inactive	$x - y$	$n - m$
Σ	x	n

TABLE 3.1: Contingency table for pattern θ.

3.3.1.1 Target Class Correlation

For a given pattern θ, a 2×2 contingency table gives a joint distribution for two binary properties of interest. Each cell counts how many times the two properties are observed with the respective value. For example, consider a table with entries f_{ij}, where $i, j \in \{1, 2\}$. Correlation can now be measured with the χ^2 function:

$$\chi^2(x, y) = \sum_{i,j} \frac{(e_{ij} - f_{i,j})^2}{e_{i,j}}, \tag{3.1}$$

where $e_{i,j}$ is the expected value of cell i, j.

In our case, the χ^2 function takes into account both the ratio of class values of the occurrences as well as the support. For any pattern, the contingency table lists the class-internal support, i.e. depending on whether the instances r covered by θ are active or inactive, i.e. $a(r) = 1$, or $a(r) = 0$, respectively. More specifically, let $x = supp(\theta, \mathbf{r})$, and let $y \leq x$ be the number of *actives* in the support of θ, i.e., with label (say) '1'. Moreover, the contingency table contains the overall distribution of target classes. Specifically, $n = |\mathbf{r}|$, and $m \leq n$ is the total number of actives in $\mathbf{r}$. The result is depicted in Table 3.1.

It can now be checked whether the distribution of θ differs significantly from the distribution of all patterns. Given a contingency table, the χ^2_d function for distribution testing, defined as

$$\chi^2_d(x, y) = \frac{(y - \frac{xm}{n})^2}{\frac{xm}{n}} + \frac{(x - y - \frac{x(n-m)}{n})^2}{\frac{x(n-m)}{n}}, \tag{3.2}$$

calculates the value for the χ^2 distribution test as the sum of squares of deviation from the expected support for both classes. One can consider significance for each target class individually. Thus, a significant pattern θ is correlated to either

1. the positive class, denoted by $\theta_\oplus$, if $y > \frac{x}{n}m$, or
2. the negative class, denoted by $\theta_\ominus$, if $x - y > \frac{x}{n}(n - m)$.

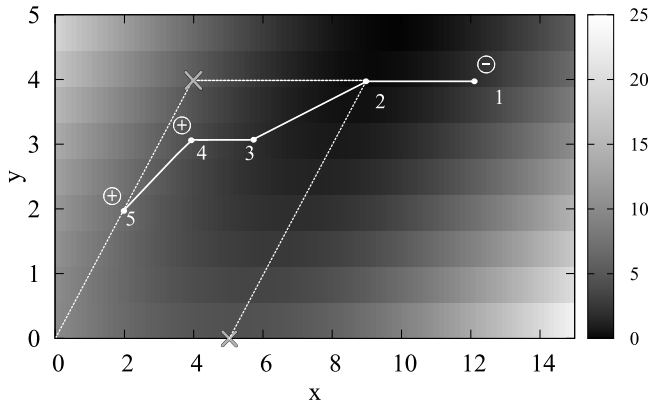


FIGURE 3.3: Visualized as stamp points, patterns 3, 4, 5 may be refinements of the current pattern (node 2), whereas pattern 1 cannot be a refinement. The χ^2_d values (indicated by the background gradient) of points 3, 4, 5 cannot exceed $\chi^2_d(4,4)$ and $\chi^2_d(5,0)$.

3.3.1.2 Stamp Points

Statistical Metric Pruning was initially suggested by Morishita and Sese [42], exploiting the convexity of the χ^2 function.

In the case described in the last section it holds that $0 \leq y \leq n$ and $0 \leq x - y \leq n$. Following Morishita and Sese's notation, we use $x(\theta)$ and $y(\theta)$ to emphasize the dependency of x and y on θ. Using the fact that convex functions take their extreme values at the points that form the convex hull of their domain, they showed that, for any subgraph θ, $x(\theta)$ and $y(\theta)$ allow to calculate an upper bound for the χ^2_d value of θ', for all refinements θ' with $\theta \preceq \theta'$:

$$\chi^2_d(x(\theta'), y(\theta')) \leq max\ \{\ \chi^2_d(y(\theta), y(\theta)),\ \chi^2_d(x(\theta) - y(\theta), 0)\ \} \tag{3.3}$$

Therefore, the upper bound measure is anti-monotonic and may thus be used for pruning the search space of methods that employ branch-and-bound, such as breadth- or depth-first algorithms.

Example 3.1. *Consider Fig. 3.3. Five different patterns are depicted as points in the space spanned by possible x and y values, forming a set of so-called* stamp points*. Moreover, it is assumed that they are in refinement relation to each other in ascending numbering, i.e. when identifying patterns with their stamp points, it holds that $1 \preceq 2 \preceq 3 \preceq 4 \preceq 5$. The depicted parallelogram determines the space where refinement stamp points of 2 with coordinates $(9,4)$ (i.e. 3, 4, 5) may lie. At the same time, no such pattern can possibly have a χ^2 value (indicated by the background gradient) larger than the maximum of $\chi^2_d(4,4)$ and $\chi^2_d(5,0)$, because*

convex functions (such as the χ^2 function) take their extreme values at the points that form the outer face of their domain.

Note that this example only considered the case of binary classification for illustrative purposes. However, generalization to the multinomial setting is straightforward using the χ^2 function.

3.3.2 Pattern Correlation and Statistical Learners

This section demonstrates (in a case study) the effectiveness of correlation measures for patterns used in a very common machine learning setting. In this study, we investigated the effects of correlated patterns on regression analysis, performed in a nearest-neighbor scenario [41]. The hypothesis was that correlated patterns would benefit a propositionalized setting in terms of predictive power of models, when no additional chemical expert knowledge was applied. Experiments with chemical datasets for various endpoints were conducted to support this claim.

3.3.2.1 Target Value Correlation

Analogous to patterns in a classification scenario, correlated patterns can be found for the case when the target information consists of continuous, numeric values. However, I am not aware of any possibility to derive an upper bound for correlation values analogous to the statistical metric pruning. The problem seems to be that convex functions are missing in this context.

The Kolmogorov-Smirnov (KS) test was used to identify features that correlate with the endpoint under consideration. It compares two cumulative probability distributions sampled from numerical data, and returns a p-value indicating the probability that the two sets were drawn from the same probability distribution (null hypothesis) [46]. Consider Fig. 3.4. Given a pattern θ, the set of target values of its support Y and the set of all target values X (whether in the support of θ or not) yield two different cumulative probability distributions.

3.3.2.2 Significance-Weighted Kernel

The learning algorithm consisted of a support vector machine (SVM), and the task was to learn a QSAR regression model that relates chemical structure to target activity in a propositionalized setting. SVMs are statistical learners that

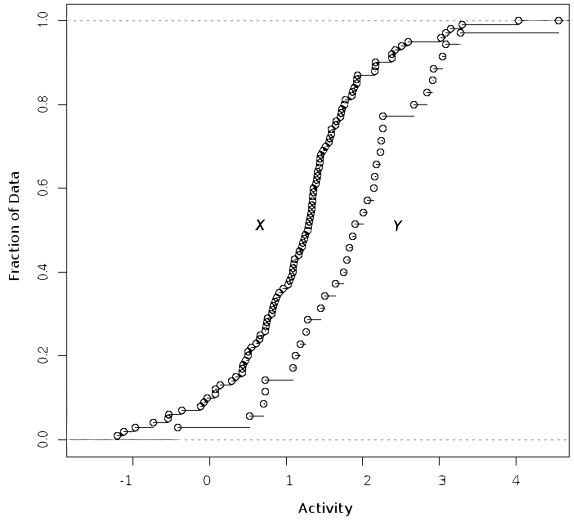


FIGURE 3.4: Comparison of the cumulative activity distributions of two (hypothetic) sets of activity values X and Y with sizes 100 and 35, respectively. The mean value of X is 1.0, the mean value of Y is 2.0. It is highly unlikely (KS test gives $p = 0.0001319$) that X and Y have been drawn from the same data source.

find class boundaries or functional relations by computing dot-products in high-dimensional feature spaces, employing so-called *kernel functions*. An introduction to kernel methods would be out of scope of this work, so the reader is referred to the literature [54] for more information.

Intuitively, a kernel defines a measure of similarity between two instances (here: molecules). In a comparative study, different spectral kernels based on chemical structure were defined and evaluated [60]. Among the kernels studied, the Tanimoto kernel [60] was evaluated favorably. It is essentially a set kernel [14]. The related Tanimoto index is one of the most useful chemical similarity indices, as shown in another famous study by Willet *et al.* [23]. The database molecules were instantiated with a set of very basic patterns, namely all linear fragments (paths) present in the database (*cf.* section 5.7). However, instead of denoting fragments covering a specific compound or not with 1 and 0, respectively (*cf.* data representation in Fig. 3.1), any fragment θ was represented by 1-p_θ, if it covered the compound, or by 0 else. Specifically, the significance weighted Tanimoto kernel is defined as:

$$k^w(r,s) = \frac{\sum_{\theta \preceq r \cap \theta \preceq s} 1 - p_\theta}{\sum_{\theta \preceq r \cup \theta \preceq s} 1 - p_\theta}, \tag{3.4}$$

i.e. the significance-weighted fraction of patterns that graphs r and s share. Note that setting all p_θ values to 0 returns the standard Tanimoto kernel.

3.3.2.3 Applicability Domain Estimation

Due to chemical diversity, every QSAR model has only a limited domain of applicability, namely "the physico-chemical, structural or biological space on which it has been trained" [32]. Consequently, it can only make valid predictions for this domain.

It has been shown that training compounds more similar to the query structure give better predictions [19]. In our approach, a neighborhood similar to the query compound with respect to structure and activity was automatically mined from the training data as follows: Only compounds $\mathbf{x_i}$ with $k^w(\mathbf{x_q}, \mathbf{x_i}) > 0.3$ were considered neighbors (referred to as the set N) to the query structure. If no neighbors for a query structure could be found according to this definition, no prediction was made. Preliminary trials showed that using a lower threshold yielded no significant information gain. Higher values, on the other hand, led to excessive amounts of empty neighborhoods and therefore missing predictions.

The so-described process of neighbor finding generates valuable information that can be incorporated into a *confidence index* indicating the reliability of the prediction, considering both dependent and independent variables [32]. It indicates the structural "density" and similarity in activity for the neighborhood, derived by statistical measures, which – of course – do not imply causality.

- The higher the median similarity $\tilde{s}$ of the neighbors, the higher the confidence in the prediction[5].
- Conversely, the higher the standard deviation σ_a of activity values of the neighbors, the lower the confidence.

A variety of kernel functions has been reviewed for smoothing similarities [3]. In the confidence index the Gaussian smoothed median similarity $\tilde{s}$ was assessed by smoothing the single neighbor similarities, and taking the median of these values:

$$\tilde{s} = median\{\ \varphi_{0.3}\ (\ 1 - k^w(r,s)) \mid \forall s \in N\ \}, \tag{3.5}$$

[5]The median is used rather than the mean to reduce the sensitivity to potential skew in the data.

where $\varphi_\sigma(x) = e^{\frac{-x^2}{2\sigma^2}}$ is the Gaussian squared exponential function with standard deviation σ. The confidence value was defined as

$$conf = \tilde{s}e^{-\sigma_a}. \tag{3.6}$$

The exponential rapidly "punishes" large σ_a, but stays close to 1 for small values. It was introduced due to the fact that the biggest errors occurred for the most active compounds, where closer investigation revealed greatly varying target values of the neighbors. This appeared plausible because those compounds require the smallest doses for a reaction, and measurement errors were more likely in the experiments that generated the data.

3.3.2.4 Significance-Weighted vs. Unweighted Kernel

The predictive performance of the significance-weighted kernel and the standard Tanimoto kernel were compared for the EPAFHM dataset (Fathead Minnow Acute Toxicity), specifically the LC_{50} values (`lc50_mmol`, 573 compounds) from *EPAFHM v4a*, dating from 15 June 2007 [51].

All patterns were used in the runs, i.e. there was no p-value threshold applied. A confidence value as described above was assigned to each prediction. Different measures of predictivity were assessed in a cumulative fashion in descending confidence order, meaning that, for every confidence level, the number describes the predictions with higher or equal confidence.

- nr is the number of predictions made.
- r^2 is the multiple correlation coefficient obtained by internal validation.
- q^2 is the cross-validated coefficient that indicates the explained variance.
- *Weighted accuracy* (wa) is the fraction of predictions within the 1 log unit error margin [4], weighted by their prediction confidence.
- *Mean error* (me) is the mean of the raw prediction errors, i.e. the errors are not standardized.
- *RMSE* ($rmse$) is the root mean-squared error of the predictions.

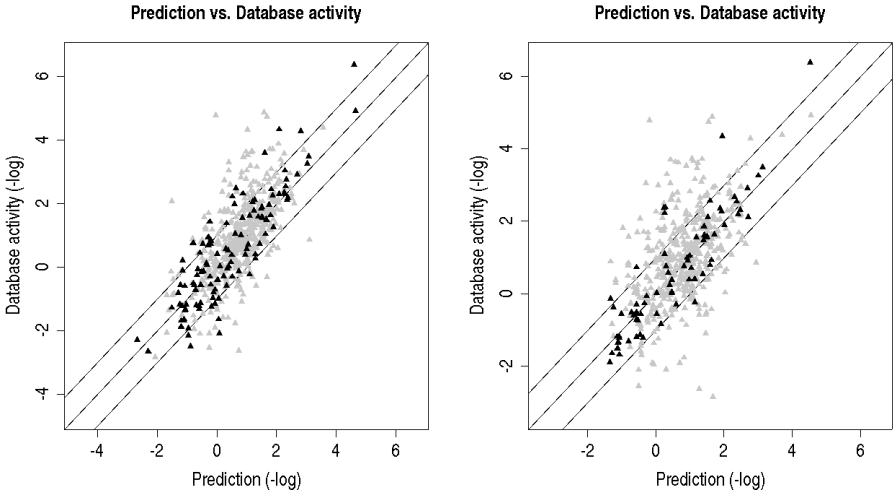
We expected for both kernels increasing performance with increasing confidence thresholds, with advantages for the significance-weighted kernel. The results (see

TABLE 3.2: Leave-one out crossvalidation comparing significance-weighted and standard Tanimoto kernel on the EPAFHM dataset using all patterns. For every confidence threshold, the table lists the number of predictions made, q^2, weighted accuracy, mean error, as well as root mean-squared error.

	SIGNIFICANCE-WEIGHTED					STANDARD TANIMOTO				
conf	***nr***	q^2	***wa***	***me***	***rmse***	***nr***	q^2	***wa***	***me***	***rmse***
0.150	207	0.68	0.79	0.61	0.82	144	0.66	0.86	0.6	0.88
0.163	195	0.68	0.78	0.62	0.83	133	0.65	0.86	0.63	0.91
0.175	178	0.69	0.78	0.62	0.83	116	0.63	0.84	0.67	0.96
0.188	165	0.69	0.77	0.63	0.85	100	0.68	0.87	0.62	0.88
0.200	147	0.74	0.77	0.62	0.8	87	0.67	0.86	0.64	0.91
0.213	127	0.75	0.76	0.65	0.83	64	0.65	0.84	0.72	1.01
0.225	116	0.76	0.77	0.65	0.83	56	0.63	0.82	0.73	1.04
0.238	100	0.76	0.75	0.68	0.86	45	0.62	0.84	0.72	1.07
0.250	90	0.75	0.75	0.67	0.85	41	0.58	0.83	0.77	1.12
0.263	81	0.72	0.74	0.68	0.87	37	0.74	0.83	0.68	0.87
0.275	70	0.73	0.72	0.71	0.9	31	0.75	0.86	0.66	0.85
0.288	65	0.73	0.76	0.64	0.83	27	0.68	0.87	0.62	0.8
0.300	60	0.73	0.77	0.62	0.8	26	0.7	0.89	0.6	0.79
median	116	0.73	0.77	0.64	0.83	56	0.66	0.86	0.66	0.91

Table 3.2) show that the significance-weighted kernel outperformed the unweighted Tanimoto kernel in nr, q^2, me and $rmse$, indicating that the p-values indeed can help to identify patterns that are significant for the exhibited activity. They also indicate a strong association between confidence and prediction quality (we suggest that the better wa values for the Tanimoto kernel may be an effect of the lower number of predictions made by this kernel).

Going one step further, it might be sensible to constrain the pattern set to patterns that are highly significant, i.e. simply to leave out the insignificant ones straight away. Plots of predictive vs. database activities for the datasets EPAFHM are presented in Fig. 3.5, assessing the difference between using all patterns and only those patterns with $p_\theta < 0.1$. Here, only the significance-weighted kernel was used. Clearly, the predictions made within the applicability domain (black dots) were much more precise in the latter scenario, although fewer predictions were made inside the applicability domain in total. Thus, using significant patterns only can



(a) (b)

FIGURE 3.5: EPAFHM scatterplots of predicted *vs.* database activity (-log(mmol/l)) with the significance-weighted kernel using all patterns (left) and significant patterns only (right). Predictions above a confidence threshold of 0.225 are drawn black, the rest gray.

benefit the model in principle. I omit a detailed discussion of results here and refer the reader to our study published elsewhere [41].

3.3.2.5 Discussion

Some aspects of the machine learning scenario detailed in the previous section deserve further attention:

Despite its members being sparsely distributed, the class of significant patterns (leaving out the other patterns) may yield predictive models when used as descriptors for machine learning. Here, we found that when the pattern set was constrained to significant patterns, then few predictions were actually made (for many query instances, no similar enough neighbors were found). In these cases, the proposed similarity was based on relatively few patterns, and such sparse selection may lead to low average similarity. Using all patterns led to more predictions, however the insignificant patterns seemed somewhat detrimental to model performance. This may be explained by the most significant being "lost" within the much larger set of insignificant patterns. The low total number of predictions may well have been an effect of the fact that only paths (linear fragments) were used in the model validation. Thus it is quite likely this could be changed once the set is extended to other substructural classes.

Confidence being associated with prediction quality is additional evidence for the importance of using significant patterns: By definition, confidence included similarity to the neighbor set as primary factor, being punished only by widely varying neighbor target values, which is in turn directly dependent on the selected patterns. As is clear from table 3.2, prediction accuracy could not be found to monotonically increase with confidence. This may be due to very small instances being (wrongly) predicted with high confidence, since the sketched nearest-neighbor approach has no correction for graph size.

3.4 Conclusions

Model building was framed as a mostly modular workflow in typical machine learning scenarios: patterns are abstractly represented as entries in a matrix (propositionalization). This representation can be readily fed into many machine learning algorithms that are implemented and publicly available.

In order to keep it as generic as possible, the workflow strictly maintains the propositionalized representation from step to step. Each step is dedicated to a specific task, the steps being mostly isolated from and ignorant of each other. At least one step in the workflow is concerned with optimizing the patterns (more specifically, the data represented by the patterns) for learning algorithms such as classification or regression algorithms.

Mining correlated patterns is somewhat orthogonal to this workflow, since, being itself a supervised process, it combines aspects of discrete pattern mining and model building. It was explained how correlated pattern mining may be integrated into frequent pattern mining and results were presented that highlight how a statistical learner can benefit from weighting patterns according to correlation, and how sparsely a dataset can be represented while still yielding high accuracy predictions.

The discussion highlighted a direction that will be followed throughout the rest of this work: The search for highly significant patterns to yield a solid representation of the dataset. It will be a crucial aspect to find those patterns in an efficient way and to omit post-processing of the pattern set, which is deemed inefficient.

Chapter 4

Common Materials and Methods

This chapter introduces some material that is common to both backbone refinement class mining and latent structure pattern mining, which will allow a better reading of the experimental sections. Accordingly, the focus is on experimental setups, which will share similar aspects, for example with regard to significance testing.

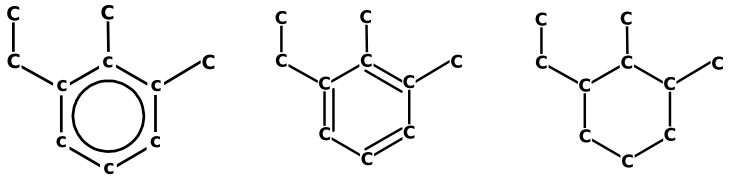
4.1 Types of Subgraph Patterns

In the experiments, several types of subgraph descriptors were evaluated and compared to backbone refinement class representatives and LAST-PM descriptors, respectively, which we list here collectively upfront:

1. *Linear Fragments*: The set of all frequent subpaths.
2. *Significant Trees*: The set of all frequent and significant (see section 3.3.1[1]) subtrees.
3. *Open Trees*: The most general trees of the subsets with the same occurrences from 2.
4. *Maximal Trees*: The most specific trees from 2.

The relation to the two types of subgraph descriptors presented in this work can be compactly described as follows:

[1]Values exceeding 3.84 ($\approx 95\%$ significance for $1df$) were considered significant.



(a) Aromatic (b) Kekulé (c) Reduced

FIGURE 4.1: Molecular Graph Representations.

- *Backbone Refinement Class Representatives*: The most significant representatives of the backbone refinement classes from 2.
- *LAST-PM Descriptors*: Latent patterns based on 2.

Note, that open trees, maximal trees, backbone refinement class representatives and LAST-PM descriptors form a summarization of the significant trees. Linear fragments will serve as a baseline comparison.

4.2 Molecular Graph Representation

We distinguish three types of representations for molecular graphs here, but from which hydrogens are deleted in any case (hydrogens attached to mined fragments can be inferred from matching the fragments back to the structures in a post-processing step):

- *Aromatic*: Most sophisticated representation, employs special node and bond labels for aromatic bindings, for example inside *Aromatic* rings. Furthermore, it allows single, double, or triple bonds anywhere else.
- *Kekulé*: In between *Aromatic* and *Reduced*, employs alternating single and double bonds within *Aromatic* rings but uses no special node labels. Allows single, double, or triple bonds anywhere else.
- *Reduced*: Most basic representation, employs only single (aliphatic) bonds throughout the structure and allows no special node labels.

Note that *Reduced* merely conveys the plain structural scaffold, whereas *Kekulé* and *Aromatic* make the special chemical role of aromatic bindings more or less explicit. *Kekulé* re-uses already available bond types to denote aromatic rings,

whereas *Aromatic* invents a dedicated bond and node type. An important property of *Kekulé* is that it is not possible to decide if a given non-cyclic fragment is part of an aromatic ring or not, in contrast to *Aromatic*. In other words, *Aromatic* is a very precise representation, whereas *Kekulé* is ambiguous. Fig. 4.1 gives examples of the three different representations, where all node labels have been made explicit. Note the special node label (lowercase "`c`") in the ring in *Aromatic*.

4.3 Datasets

4.3.1 Small and Medium-Sized Datasets

According to the formal introduction in section 2.2.1, the datasets feature binary target classification. This means every molecule is assigned to a target class (e.g. *active* vs. *inactive*). All datasets are publicly available, either from the original websites or from the author's website (see Appendix C).

OFS-Data: The three small-sized chemical datasets were originally used in "**O**ptimizing **F**eature **S**ets for Structured Data" by Rückert and Kramer [49].

- Estrogen Receptor Binding (`nctrer`, 139 active / 232 compounds) [12]
- Bioavailability (`yoshida`, 159 active / 265 compounds) [68]
- Blood-brain barrier (`bloodbarr`, 276 active / 413 compounds) [38]

The `nctrer` dataset deals with the binding activity of small molecules at the estrogen receptor, the `yoshida` dataset classifies molecules according to their bioavailability, and the `bloodbarr` dataset deals with the degree to which a molecule can cross the blood-brain barrier. These characteristics are important for drug design and drug treatment:

Bioavailability characterizes speed and quantity of a drug being available at the target organ of treatment and is thus a vital parameter of prospective new medicinal agents. Also, for treating disorders of the central nervous system, drugs must be able to penetrate the bloodbrain barrier to be able to reach the target organ (the brain). This ability is thus a vital aspect in designing such drugs. Finally, preconditions for estrogen receptor binding are applied in drug design for human estrogen replacement therapy, as well as to identify estrogenic endocrine disruptors.

CPDB-Data: Four medium-sized chemical datasets were obtained from the Carcinogenic Potency Database (CPDB) at `http://potency.berkeley.edu/cpdb.html`, version 08/04/29, were used in the experiments:

- *Salmonella* Mutagenicity (`SM`, 388 active / 810 compounds)
- Rat Carcinogenicity (`RC`, 459 active / 1145 compounds)
- Mouse Carcinogenicity (`MoC`, 428 active / 927 compounds)
- Multicell Call (`MuC`, 553 active / 1067 compounds).

Mutagenicity and carcinogenicity are obviously of major importance for all aspects of hazard detection and risk assessment of chemicals, as well as to drug design. Corresponding studies have been routinely carried out using laboratory animals (*in-vivo* testing), which raises ethical problems. Legislation, e.g. in the EU, demands to reduce, refine and finally replace animal testing, wherever possible, which has been raising a substantial interest in computational (*in-silico*) methods for modeling such properties.

4.3.2 Large-Scale Datasets

For the large-scale analysis, experiments were performed on parts of the NCI Yeast Anticancer Drug Screen datasets at `http://dtp.nci.nih.gov/yacds/download.html` (April 2002 release). These datasets reports growth inhibition of yeast strains when exposed to chemicals as compared to solvent only:

- `AC-One` (stage 0): Total of 87,264 compounds, 12,068 active (active, if growth inhibition of at least 70 % in at least one strain)
- `AC-All` (stage 0): Total of 87,264 compounds, 5,777 active (active, if growth inhibition of at least 70 % in all strains)
- `AC-All` (stage 1): Total of 10,924 compounds at the high dose (50 microM), 5,433 active (active, if growth inhibition at least 70 % in all strains)

This data was gathered by the National Cancer Institutes of the United States to facilitate drug design. Chemicals were applied to cell lines with mutations in order to assess their potency as drug candidates for cancer treatment.

Chapter 5

Backbone Refinement Class Mining

This chapter treats *Backbone Refinement Class Mining*, which extracts a sparse class-correlated collection of subgraphs from a chemical database by combining statistical and frequency constraints with structural constraints.

Section 5.1: Introduction. Discusses what could be improved on current methods to summarize the search space of frequent significant trees, leading to an informal description of backbone refinement class mining. An intuitive account of the approach supports the sketched direction. Backbone refinement class mining is formally defined.

Section 5.3: Compression. Examines the compression potential of backbone refinement class mining by comparison to the set of all subtrees in an artificial -but reasonably chosen- search space.

Section 5.4: Coverage and Representativeness. What parameters influence coverage and number of instances a pattern represents most? Does the sparsity of backbone refinement class descriptors impair coverage compared to other descriptors? The questions are treated using several values for minimum frequency and two different chemical representations.

Section 5.5: Diversity In Structure And Occurrence. Predictive models and human experts need descriptors that are diverse, so that they can describe data in a variety of aspects. The diversity and distribution of patterns produced by backbone refinement class mining are investigated here.

Section 5.6: Runtime Analysis. Backbone refinement class mining employs an effective pruning strategy which we analyze in detail. Its connection to top-k mining is discussed and its power assessed in experiments.

Section 5.7: Classification Accuracy. Re-iterates on the question of chemical representation from section 5.4 by comparing three representations according to classification accuracy. Then backbone refinement class descriptors are compared against other compressed substructural representations.

5.1 Introduction

Current methods for subgraph mining still suffer from scalability problems and, quite related, problems with excessively large solution sets. Most of the predominant approaches employ minimum frequency and possibly statistical correlation criteria such as χ^2 values [6, 31, 36, 44, 66]. An increasing minimum frequency tends to favor a higher pattern entropy, whereas statistical constraints retrieve subgraphs that are correlated with the target classes. However, the thresholds used still lead to an explosion in the number of frequent patterns, which is a significant drawback of these approaches. As a result, the size of the resulting pattern set is usually multiple times larger than the original database, rendering it useless for subgraph-based classification models and individual inspection by domain experts, at least for very large datasets. In order to build a more sparse representation with a significantly reduced number of patterns, the solution set is often constrained to open or closed patterns. This might however prune important structural information, since only the support is considered, which may not be sufficient to adequately summarize the diversity of subgraphs.

Hence, the implementation of an explicitly structural summarization strategy (defined intensionally) should yield a higher compression ratio while still retaining all essential information and achieving better running times. This approach, called *Backbone Refinement Class Mining* (or *BBRC mining* for short) combines principles from correlated subgraph mining [6, 42] with a novel strategy to ensure diversity in structure rather than in subgraph occurrence. It proposes to increase structural dissimilarity in order to both increase inter-pattern entropy and decrease the number of solution patterns. A natural property of tree-shaped subgraphs (the backbone) is used to represent classes, which renders the set suitable for computational models even for large-scale datasets.

The remainder of this chapter discusses related work, the intuition behind backbone refinement class mining, and concludes with a formal definition.

5.1.1 Related Work

Enumerating all frequent subtrees in a database can be done efficiently by using depth-sequences, see section 2.2.4.3. In order to summarize the mined patterns in the unsupervised setting, i.e., when no target class information is available, most methods consider (only) the support of patterns. For instance, the solution can be restricted to open (closed) subgraphs of various types. By definition, these

techniques represent refinements with identical occurrences by the most general (the most specific) pattern. They can be easily integrated into graph mining with minimum support.

Rückert and Kramer [49] presented a solution to class-correlated pattern mining based on occurrence lists, which aims for extensionally diverse sets of structural patterns. *Stochastic Local Search (SLS)* is used to optimize the dissimilarity of patterns, more specifically, to minimize the dot product between occurrence vectors. The main drawback of the approach is the excessively long running time of *SLS* to find small sets of diverse patterns.

Generally, the common strategy in most of the current approaches is to represent a large part of frequent patterns by representatives that form a summary of their occurrences. However, ignoring the wealth of structural information may be a drawback for three main reasons:

1. Occurrence-based representation is not directly related to structure. For instance, refinements on different support levels are not observed together, which might prune important data or could lead to redundant patterns.

2. For the same reason they also do not distinguish between different subgraph types for occurrence-based methods, e.g., paths are refined to trees arbitrarily without changing the mining strategy. Given the variability of graphs, structural invariants may be desirable to be able to process different parts of the search space differently, for example to be able to concentrate on subtrees only and speed up the search, or to use the structural information to process the gathered patterns even during the search, which could help to avoid postprocessing.

3. Occurrence-based methods are forced to mine a representative for any frequent support level, which could impact performance.

Al Hasan *et al.* [2] employ sparse representations of maximal frequent subgraphs obtained by sampling. The approach aims for structural diversity (*orthogonality*) of the sampled patterns while enforcing a certain level of structural similarity (*representativeness*) at the same time. It is a supervised method that repeatedly evaluates candidate sets using a score function.

In contrast to occurrence-based representations, such as open/closed patterns, I propose to partition the search space directly by structure. The `ORIGAMI` patterns are defined structurally, and therefore bear some similarity to the descriptors proposed here. Backbone refinement class representatives are compared favorably

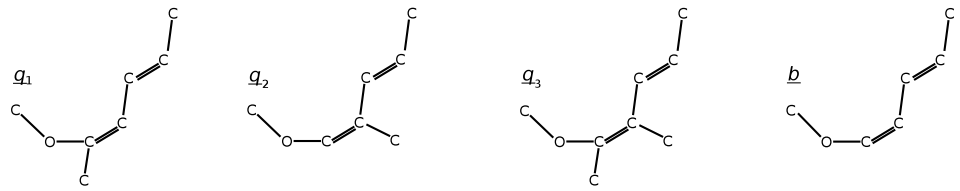


FIGURE 5.1: Three example trees q_1, q_2, and q_3 with the same backbone b. It holds that $q_1 \preceq_b q_3$ and $q_2 \preceq_b q_3$, but neither $q_1 \preceq_b q_2$ nor $q_2 \preceq_b q_1$. Therefore, q_1 and q_2 are not in the same backbone refinement class.

against those patterns, which will be underpinned by theoretical and empirical evidence for their potential to provide a compressed representation of frequent and significant patterns, and to significantly improve classification accuracy over occurrence-based techniques. Moreover, it is shown that a structure-aware search can be implemented efficiently and that the method is scalable to datasets larger than those previously considered in correlated class mining.

5.1.2 A New Class of Substructures

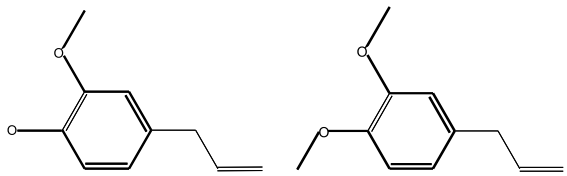
Recall the definition of path from section 2.2.1. As discussed in section 2.2.4.4, depth sequences can be used to canonically enumerate trees that grow from a specific backbone. An (immediate) tree refinement of $t \in T$ is an addition of an edge and a node to t, such that the result t' is still acyclic, i.e. $t' \in T$. A *backbone refinement* is a tree refinement that is backbone-preserving, i.e. $b(t') = b(t)$.

The following idea is central to backbone refinement class mining: A given backbone spans a maximal tree, i.e., no tree refinement may be added to that tree without changing the backbone. Since every tree has exactly one backbone, by maximally tree-refining all backbones, the partial order of trees is disjointly partitioned across different backbones.

The classes used here are induced by the conjunction of a backbone and a refinement, hence the name *backbone refinement classes* (*BBRCs*).

Example 5.1. *In Fig. 5.1,* q_1 : `C-C=C-C=C(-C)-O-C` *and* q_2 : `C-C=C-C(-C)=C-O-C` *are in different classes, but* q_3 : `C-C=C-C(-C)=C(-C)-O-C` *is in the respective classes of both* q_1 *and* q_2 [1]*. Assume that the search starts with the backbone* b : *C-C=C-C=C-O-C from Fig. 5.1. Assume further that the search then refines to* q_1*. It may further refine to* q_3*. However, to enumerate* q_2*, it will have to backtrack*

[1] See `http://www.daylight.com/dayhtml/doc/theory/theory.smarts.html` for notation.



(a) Eugenol (4-allyl-2-methoxyphenol, Pubchem CID 3314)

(b) Methyleugenol (4-allyl-1,2-dimethoxybenzene, Pubchem CID 7127)

FIGURE 5.2: Small structural modifications turn a harmless substance (left) into a carcinogen (right). The backbones are marked bold.

to the backbone position and branch and initiate a new class. Since q_3 is a supergraph of q_1 and q_2, it is in both classes (but it is only enumerated once due to the enumeration strategy).

5.1.3 Intuition

According to section 5.1.2, backbone refinement classes partition the search space structurally. Fig. 5.2 shows an example where structural summarization directly detects information that other methods may miss. It depicts two structurally similar compounds: eugenol, Fig. 5.2(a), and methyleugenol, Fig. 5.2(b). Eugenol is used widely as flavoring agent and in medicine as a local antiseptic and anesthetic. Methyleugenol, however, is "*reasonably anticipated to be a human carcinogen*"[2]. In experiments with rodents, methyleugenol induced cancer in multiple organs, especially in the liver. Note that the structure graph of the latter is a refinement of the former. Such deviations might therefore not be detected by methods that represent sets of patterns by most specific or most general patterns. In contrast, the structural method is able to detect this case: Due to the backbone change between the two structures, the structures belong to different backbone refinement classes[3] and would therefore be represented differently. It can be argued that the backbone change in the example was selected on purpose. In practice, however, it is useful in a surprisingly large number of cases. Also, other structural invariants are conceivable, e.g. constraints on branching.

Therefore, I propose to select the most significant representative from every class induced by the structural criterion. Fig. 5.3 schematically depicts the general

[2]See `http://ntp.niehs.nih.gov/ntp/roc/eleventh/profiles/s109meth.pdf` for details.

[3]The approach is currently limited to tree-shaped fragments, but that should not invalidate the argument.

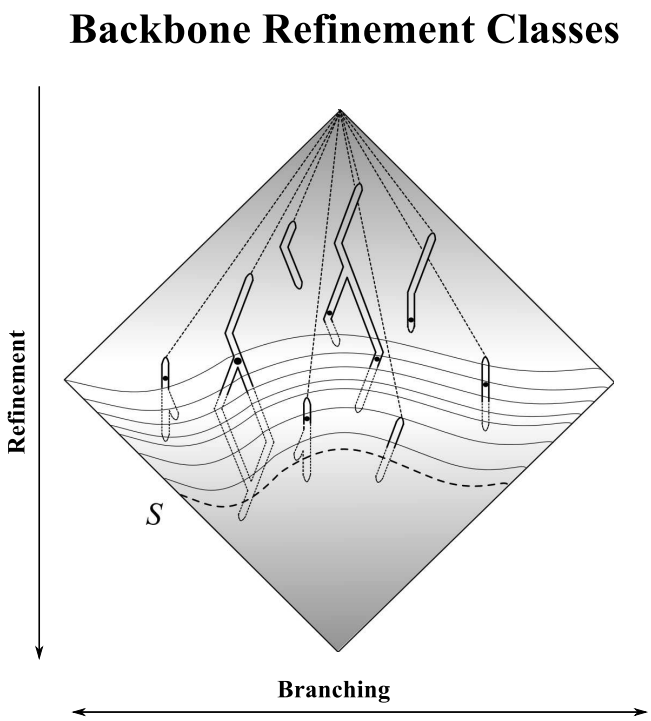


FIGURE 5.3: Backbone Refinement Class search space, spanning multiple support levels as opposed to occurrence-based methods. Backbone Refinement Class Representatives are more probable in lighter regions of the hypothesis space.

search space for backbone refinement classes (see section 5.1.2). The search branches into different directions, corresponding to mutually exclusive sets of patterns with respect to "$\preceq$" and sometimes joins again.

Classes may spread across many support levels, due to their structural definition-however, class representatives are more likely in the lighter regions of Fig. 5.3, i.e., patterns with medium support tend to be most significant. Due to the convexity of the χ^2 function, patterns with very low or high support either have too little weight, or their occurrences' class distribution resembles closely the overall database distribution. More formally, for high-support patterns, the weight factor (see section 3.3.1.1) $\frac{x}{n}$ is close to 0, while for low-support patterns y is close to m, and $x - y$ is close to $n - m$, where x (y) is the number of instances (active instances) covered by the pattern, and n (m) is the number of instances (active instances), see section 5.6.1 for the exact definitions. To summarize, the dichotomy between classes leads to a sparse (by taking only the maximal class member) and structurally diverse (by the structural definition of classes) selection of patterns. Most backbone refinement class representatives are sampled from the middle of the search space, where the majority and the most diverse patterns reside. This distribution also allows for early pruning of the search as well as robustness against increasing minimum frequency, since not every frequent support level is visited.

5.2 Backbone Refinement Class Mining

The concepts are formally defined now.

5.2.1 Structurally Defined Classes

Definition 5.1 (Backbone Refinement Classes)**.** Consider the *Backbone Refinement Classes* of backbone $b \in P$, denoted by $BBRC_b = \{BBRC_{b_1}, \ldots, BBRC_{b_n}\}$, where each $BBRC_{b_i}$ is the set of trees that are backbone refinements of each other with respect to b, i.e. for all $t, t' \in BBRC_{b_i}$ the following conditions hold:

1. $b(t) = b(t')$ and
2. $t \preceq t'$ or $t' \preceq t$.

The backbone refinement relation given by conditions 1. and 2. is denoted by $\preceq_b$. The set of all backbone refinement classes for a graph database R is called $BBRC_R$.

It follows directly from the definition, that backbone refinement classes are in general not disjoint for the same backbone. More precisely, given two BBRCs $BBRC_{b_i}$ and $BBRC_{c_j}$, they are associated in one of two possible ways: $BBRC_{b_i}$ and $BBRC_{c_j}$ are either

1. not disjoint, iff $b = c$, or
2. disjoint, iff $b \neq c$.

For case 1, members of the two classes $BBRC_{b_i}$ and $BBRC_{c_j}$ are enumerated such that the refinement process starts at the same backbone, but at some later point branches into different directions. It is possible that $BBRC_{b_i}$ and $BBRC_{c_j}$ have a common maximal supergraph (see Fig. 5.1 and 5.3). For case 2, no common elements exist, since b and c occupy disjoint parts of the search space. Fig. 5.4 gives an example of two backbone refinement classes with the same backbone. They have a common maximal supergraph (case 2).

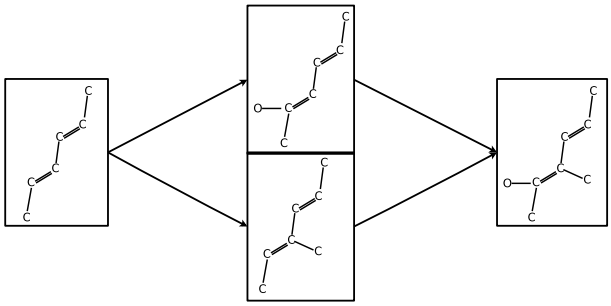


FIGURE 5.4: Backbone Refinement Classes. In this example, the two classes have a common maximal supergraph.

5.2.2 Mining Representatives

The objective of Backbone Refinement Class Mining is to find the most significant representative for each backbone refinement class, formally:

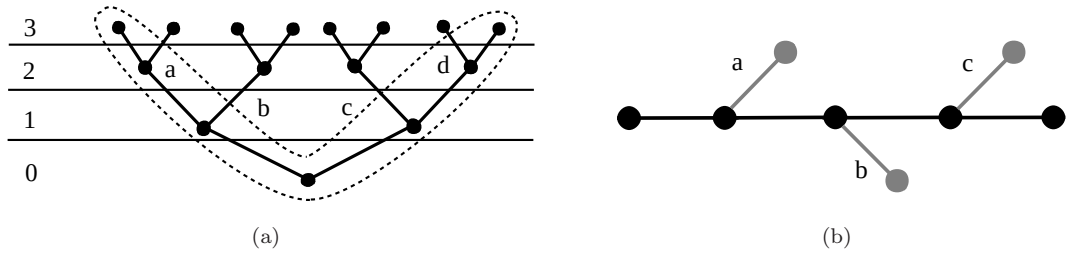
Definition 5.2 (Backbone Refinement Class Mining)**.** Given a graph database R, a user-defined minimum support *minsup* and user-defined minimum χ^2 value u, for all $B \in BBRC_R$, find the most significant $t \in B$ that is *frequent*, i.e. $supp(t, R) \geq minsup$, and *significant* with respect to occurrence in the target classes, i.e. $\chi^2_t \geq u$. If more than one such patterns exist, return the most general one.

Note that the selection process is in fact not oriented on frequency levels (see section 5.1.3). Instead, it is not clear a priori across how many levels of frequency a class stretches and where the representative patterns reside.

5.3 Compression

Backbone refinement classes address the problem of exponentially sized result sets right during the mining process, not by post-processing. Also, compared to other graph mining methods that output condensed representations of the fragment pattern space, it shows a much higher degree of compression.

In the following, a formula for the number of backbone refinement classes induced by a rooted perfect binary tree will be derived. We will compare this number to the complete set of subtrees containing the root node, which shows the amount of compression possible with backbone refinement classes. Then, those properties are put to the test in experiments.



(a) (b)

FIGURE 5.5: Left: Rooted perfect binary tree with height 3. A longest path β^* of length 6 has been marked by dashes. It has branches $B_{\beta^*} = \{a, b, c, d\}$, where the subtrees induced by b and c have longest paths of length $\sigma(b) = \sigma(c) = 2$. The path β^* induces $\rho(\beta^*) = 4! = 24$ backbone refinement classes. Right: A backbone with branches a, b, c (gray) attached.

First, a theoretic bound on the degree of compression with respect to the full set of all subtrees will be derived in sections 5.3.1 - 5.3.4. Following that, we will see the actual degree of compression backbone refinement class representatives yield for several medium-sized CPDB datasets and for parts of the NCI dataset, the largest dataset that has been used in correlated graph mining.

5.3.1 Induced Backbone Refinement Classes

A *rooted perfect binary tree* is a perfectly balanced binary tree with a designated root node of degree 2. An example of height 3 is shown in Fig. 5.5(a).

Of course, in real datasets, the search space will most likely not have the perfect binary property. However, we observed that trees for typical carcinogenicity and mutagenicity databases have a branching factor < 2. Therefore, it seems legitimate to simplify the theoretical setting and use binary trees for the estimation of compression. Furthermore, intuition tells that the relationships described in the following also hold for higher branching factors.

Example 5.2. *Consider a backbone β of a certain length such as the one in Fig. 5.5(b) with length 4. A branch (gray) is either present or not present (Fig. 5.5(b) shows all branches present). The number of branches is $\sigma(\beta) = length(\beta) - 1$, where $length(\beta)$ is the number of edges in β.*

Let the branches be labeled $a, b, c, \ldots$. Consider the set B of branches $\{a, b, c, \ldots\}$ and all its subsets including the empty set. The subsets can be partially ordered in a directed set graph according to "$\subset$" (set inclusion), such as shown in Fig. 5.6 for the set $\{a, b, c\}$. Note that there is alway a node corresponding to the full set of branches (with *indeg* $\sigma(\beta)$ and *outdeg* 0), as well as a node corresponding to

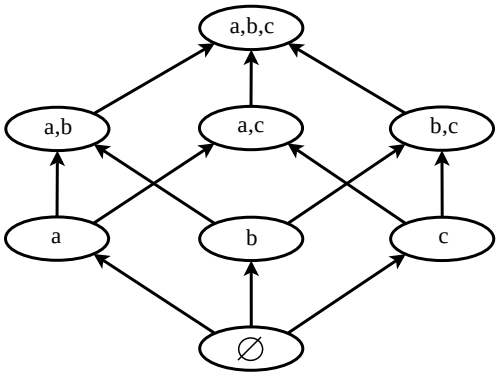


FIGURE 5.6: Set graph corresponding to the subsets of $\{a, b, c\}$ and the partial order induced by the subset relation.

the empty set (with *indeg* 0 and *outdeg* $\sigma(\beta)$), where *in(out)deg* of a node is the number of edges going into (emanating from) that node.

Lemma 5.3. *The number of paths from the empty set node to the full set node in the set graph of a backbone equals the number of backbone refinement classes induced by this backbone. It is* $\rho(\beta) = \sigma(\beta)!$.

Proof. Starting at the empty set node, after having traversed i edges, there are $\sigma(\beta) - i$ ways to add exactly one item to the current set, which in combination yields $\sigma(\beta)!$ different paths. Let $B_1, \ldots, B_n$ denote all the subsets of B, including the full and empty sets. Associate $x \in B_i$ with the following meaning: x is present on the backbone. It follows that two sibling nodes in the set tree induce two different backbone refinement classes, since all elements in one class contain a branch that no element from the other class contains (no two trees are refinements of each other). On the other hand, the subgraph relation between a child and a parent node does not induce a new backbone refinement class, since all branches on the parent are still present on the child. □

5.3.2 Number of Backbone Refinement Classes

Lemma 5.4. *The number of backbone refinement classes induced by a subtree of height* h *with root node on the backbone of a rooted perfect binary tree is recursively given by*

$$b(h) = (h-1)! + (h-1)\sum_{i=1}^{h-1} b(i). \tag{5.1}$$

Proof. From Lemma 5.3: The number of backbone refinement classes induced by a longest path β of this branch is $\rho(\beta) = (h-1)!$.

Then, for every branch b_i of the branches $b_1, \ldots, b_{h-1}$ of β, we recursively add its number of induced classes. Every branch can be combined uniquely with the $h-2$ other branches, or combined with none, thus appearing in $h-1$ induced classes. □

We are now in the position to state the result of this section.

Theorem 5.5. *The number of backbone refinement classes in the search space of rooted perfect binary trees of height h is recursively given by*

$$B(1) = 1 \tag{5.2}$$

$$B(h) = \sum_{i,j=2}^{h} 2^{i-1}2^{j-1} \left[(i+j-2)! + (i-2)\sum_{s=1}^{i-1} b(s) + (j-2)\sum_{t=1}^{j-1} b(t) \right], \qquad h \geq 2 \tag{5.3}$$

Proof. There are $\sum_{i,j=2}^{h} 2^{i-1}2^{j-1}$ paths containing the root in a rooted perfect binary tree of height h. For each pair (i,j), the corresponding path has $i+j-2$ subtree inducing branches (because of the missing branch at the root node) and $(i+j-2)!$ induced backbone refinement classes. Then, for every branch, we add its number of induced classes. On each side of the root, every branch can be combined uniquely with the $i-2$ and $j-2$ other branches, respectively. □

5.3.3 Number of Subtrees

Szekely and Wang [61] showed that a rooted perfect binary tree with height h has

$$F(h) = \lfloor q^{2^{h+1}} \rfloor - 1 \tag{5.4}$$

non-empty subtrees containing the root, where $q \approx 1.502837$.

5.3.4 Comparison of Backbone Refinement Classes and Tree Set Sizes

For different values of height h we calculate the number of backbone refinement classes $B(h)$ according to Theorem 1, as well as the complete tree set size $F(h)$ according to the result of Szekely and Wang. Fig. 5.7 compares the values for heights 1 to 8. Clearly, F grows much faster than B. Note that there is a double exponential in F which does not occur in B. The log-scaled chart in Fig. 5.7

h	B(h)	F(h)
1	1	4
2	8	25
3	632	676
4	60,280	458,330
5	11,922 296	2.101e+11
6	4.126e+09	4.413e+22
7	2.138e+12	1.947e+45
8	1.537e+15	3.792e+90

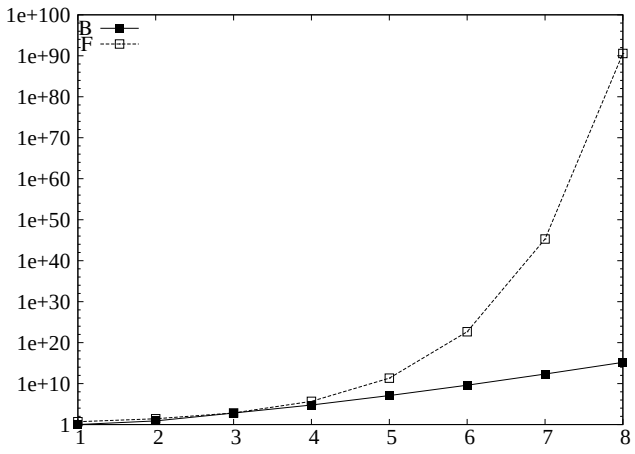


FIGURE 5.7: Comparison of backbone refinement classes and full subtree set sizes for heights 1 to 8 (log-scaled).

reveals this extra exponential growth of F, while B is nearly linear on the investigated interval. In summary, we have a compression by backbone refinement class representatives of at least from double exponential to exponential, compared to the full set of subtrees.

5.3.5 Experiments

The compression of three common types of fragment descriptors, namely all linear subgraphs, significant trees, and open trees, were compared to that of backbone refinement class representatives. Note, that open trees and backbone refinement class representatives form a summarization of the significant trees (see section 4.1). Minimum frequencies of 6 and 200 were employed for medium-sized and large-scale data, respectively, to avoid excessive numbers of subgraphs. This was well below 1 % of the respective data set sizes for the medium-sized and 2 % for the large-scale datasets. For the linear subgraphs, no minimum frequency and no significance threshold was used; however, refinement was stopped at frequency 1, i.e. a fragment with single occurrence was included in the pattern set but not further refined. All datasets were given to the algorithm in non-aromatic representation, i.e. without special labels for carbons and edges that are part of an aromatic ring.

Table 5.1 compares the resulting pattern set sizes for the different fragment types. Fig. 5.8 summarizes this information by mean values of relative fragment counts across the different datasets, revealing high compression for backbone refinement class representatives. On large-scale data, the pattern set sizes of maximal trees

	SM	RC	MoC	MuC
1. Linear Fragments	48,259	86,300	49,816	70,802
2. Significant Trees	27,093	94,991	22,395	29,970
3. Open Trees	8,062	4,569	1,937	5,122
4. BBRC Representatives	2,715	5,183	3,083	3,636

TABLE 5.1: Pattern set sizes for all linear fragments, significant trees, open trees, backbone refinement class representatives for the four CPDB datasets.

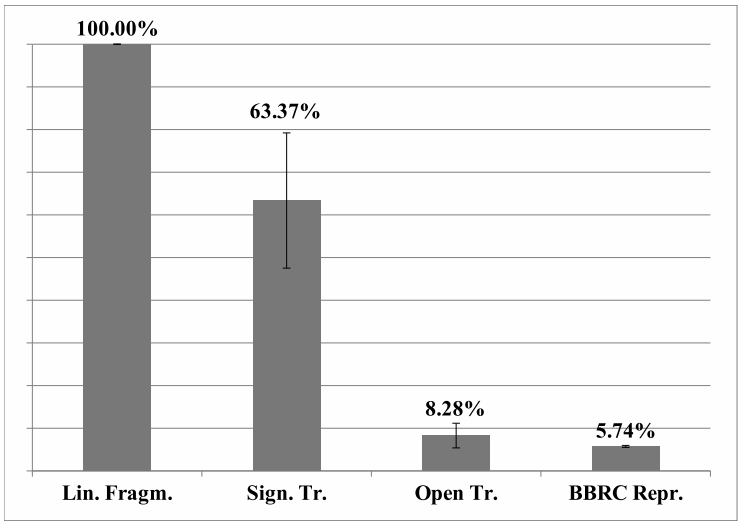


FIGURE 5.8: Fragment count mean reduction.

were investigated instead that of linear fragments due to combinatorial explosion for the latter on a dataset with this size (see Table 5.2). Here, a subset of `AC-one` (stage 0), composed of all actives and an equal number of inactives sampled randomly from the dataset (2*11,700 = 23,400 compounds) was used. The second run, on `AC-All` (stage 1), used all actives and inactives (in total, 5,248+5,300 = 10,548 compounds). Moreover, we used aromatic perception here to keep the pattern set as concise as possible (see section 5.4).

The empirical findings suggest that fragment set sizes may be reduced by 94 %, 91 % and 31 % through the use of backbone refinement class representatives compared to all linear subgraphs, significant trees and open trees, respectively. On large-scale data, the effect was even more pronounced: Backbone refinement class representatives had a very condensed representation of below 5%, whereas maximal and open trees achieved a reduction up to only ~ 50%. Thus, backbone refinement class representatives reduced the fragment sets much more drastically for these large-scale datasets than for the smaller validation datasets.

	`AC-One` **(stage 0)**	`AC-All` **(stage 1)**
Significant Trees	1,190,763	291,729
Open Trees	?	216,206
Maximal Trees	556,673	148,562
BBRC Representatives	31,450	14,381

TABLE 5.2: Pattern set sizes for large-scale datasets `AC-One` (stage 0) and `AC-All` (stage 1). '?' indicates that computation terminated with an error.

.

5.3.6 Conclusions

Backbone refinement class representatives are able to compress the pattern space very efficiently. It can be shown that the number of backbone refinement class representatives grows only exponentially, while the set of all subtrees grows double-exponentially with the height of the artificial search space. The joint effect of the two key components of backbone refinement class mining explains the sparse representation. On the one hand, the search space is structurally partitioned into equivalence classes, conversely to occurrence-based approaches. Such a partition may contain multiple levels of occurrences in the search space. On the other hand, the pattern set for each partition is reduced to a single representative: its most correlated member. Therefore, the "granularity" of backbone refinement classes is much higher than occurrence-based approaches, while structural dissimilarity of the representatives is favorable (see section 5.5).

In the experiments, we have compared the compression of open and maximal trees to those of backbone refinement class representatives. Given that the former methods have been standard approaches for pattern set reduction in graph mining – even without the reduction to significant patterns – and the huge differences shown in the experiments, the structural compression approach of backbone refinement class mining may interesting for condensed representations in models as well as for human experts.

5.4 Coverage and Representativeness

The measures of *coverage* (number of patterns per instance) and *representativeness* (number of patterns per class), which we will investigate in this section, plays an important role in the question how sparse we can make our pattern sets without

loosing much descriptive potential: The higher the coverage and the lower the representativeness, the richer each instance is described, which in turn is beneficial for models (at least those that can handle high dimensionality). A pattern set should, on the other hand, allow for fast, memory saving computational models, as well as for interpretable representations of the data set- raising a demand for a reduced set of significant, non-redundant patterns.

While this section gives insights into coverage and representativeness of backbone refinement class descriptors *per se*, it also investigates how stable these properties are. Intuitively, backbone refinement class representatives should occur more frequently than at minimum frequency since they are not the most specific, but the most significant members of their respective classes. Therefore, the identity of backbone refinement class representatives (and thus their occurrences) should be somewhat independent of minimum frequency. In a chemical context, it is interesting to assess the effect of aromatic perception, i.e. if molecules are represented with aromatic bindings or not.

5.4.1 Experiments

Coverage and representativeness were assessed under the factors of varying minimum frequency and (non-)aromatic perception on two large-scale datasets. Different degrees of coverage and representativeness were not actually tested in models at this time- rather, those characteristics were studied when parameters of the mining process were changed. Besides medium-sized datasets, large-scale data was used to make the effects as pronounced as possible, i.e. a large number of instances should render differences in minimum frequency maximally visible.

5.4.1.1 Coverage

According to the intuition from section 5.1.3, backbone refinement class mining is relatively robust in terms of coverage, since only the most significant member of each backbone refinement class is mined and since the most significant fragments occur in the middle of the search space. Thus, growing minimum frequencies should remove fewer patterns when the dataset is represented by backbone refinement class representatives, as compared to when it is represented by the set of significant trees (see section 4.1).

Table 5.3 shows how coverage decreases (in percentages of the original set size), when minimum frequency is raised from 3 to 4, 5, 6, 8, 10, 20, and 40‰ of the

		SM		**RC**		**MoC**		**MuC**
MF	ALL	BBRC	ALL	BBRC	ALL	BBRC	ALL	BBRC
‰	%	%	%	%	%	%	%	%
3	100.00	100.00	100.00	100.00	.	.	100.00	100.00
4	100.00	97.92	100.00	100.00	.	.	100.00	100.00
5	100.00	95.83	94.64	96.00	.	.	91.38	95.65
6	100.00	95.83	91.07	96.00	.	.	84.48	91.30
8	91.04	95.83	82.14	88.00	100.00	100.00	79.31	91.30
10	85.07	91.67	80.36	88.00	100.00	100.00	77.59	82.61
20	73.13	87.50	69.64	76.00	76.47	83.33	65.52	73.91
40	56.72	83.33	48.21	64.00	52.94	58.33	50.00	60.87

TABLE 5.3: Coverage Table for Backbone Refinement Class Representatives and Significant Trees for the CPDB datasets and different values of minimum frequency. Dots indicate missing values due to combinatorial explosion.

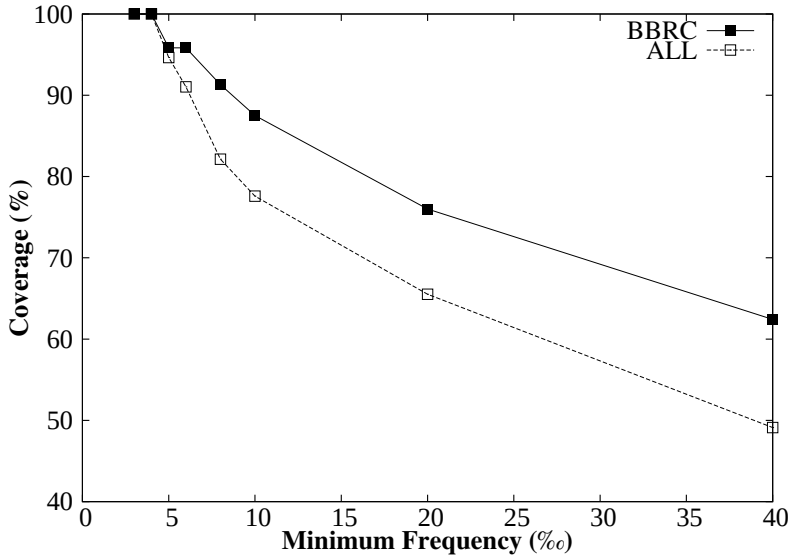


FIGURE 5.9: Mean values of Table 5.3, taken across the CPDB datasets `SM`, `RC`, and `MuC`.

dataset size. For *MoC*, we had to leave out the lowest frequencies to combinatorial explosion of the pattern set. Clearly, for all four datasets, backbone refinement class representatives maintained a higher relative coverage. The effect is quite pronounced, although it varies between the datasets. Fig. 5.9 summarizes Table 5.3 by taking mean values across `SM`, `RC`, and `MuC`. It suggests that, on average, backbone refinement class representatives are able to maintain about 10% higher

Dataset	Aromatic?	Min-freq	Mdn	M (SD)	# C
`AC-One` (stage 0)	y	200	1.623	1.650 (0.479)	84,414
		100	1.633	1.673 (0.509)	84,500
	n	200	1.903	1.878 (0.501)	84,415
		100	1.914	1.896 (0.523)	84,422
`AC-All` (stage 0)	y	200	1.580	1.582 (0.499)	84,342
		100	1.613	1.631 (0.516)	84,458
	n	200	1.785	1.751 (0.488)	84,437
		100	1.820	1.800 (0.493)	84,485

TABLE 5.4: Coverage Table for `AC-One` (stage 0) and `AC-All` (stage 0) datasets ($log10$), with and without aromatic ring perception.

coverage relative to the lowest frequency than the complete set of significant trees.

For the large-scale data, first a minimum frequency of 200 (≈ 2.3 ‰) was examined for the `AC-One` (stage 0) and `AC-All` (stage 0) datasets to see how well backbone refinement class representatives would cover the data set, i.e. numbers of descriptors per compound were assessed, and compared to the respective figures for a minimum frequency of 100. This experiment was performed twice to also check the impact of aromatic perception, i.e., whether the algorithm uses special labels for carbons and edges that are part of an aromatic ring, or whether it uses Kekulé notation (employing alternating single and double bonds).

Table 5.4 shows that almost all instances are covered by at least one pattern at all times (column "#C"), regardless of minimum frequency. However, from the mean and median values, we see that minimum frequency has hardly any effect on mean coverage, whether with or without aromatic perception. The `AC-One` (stage 0) dataset with aromatic perception has a mean log value of 1.650 for a minimum frequency of 200, and a mean log value of 1.673 for a minimum frequency of 100. Moreover, we observe that the difference between median and mean shrinks from lower to higher minimum frequency. In contrast, missing aromatic perception raises mean coverage substantially: Here, the `AC-One` (stage 0) dataset has a mean log value of 1.878 for a minimum frequency of 200 and a mean log value of 1.896 for a minimum frequency of 100. Without aromatic perception, however, the difference between median and mean does not shrink, but grows instead. In Table 5.4, we observe a similar trend for the `AC-All` (stage 0) data . Therefore, the mean number of descriptors per compound is about $10^{1.63} \approx 42.6$ [$10^{1.61} \approx 40.7$]

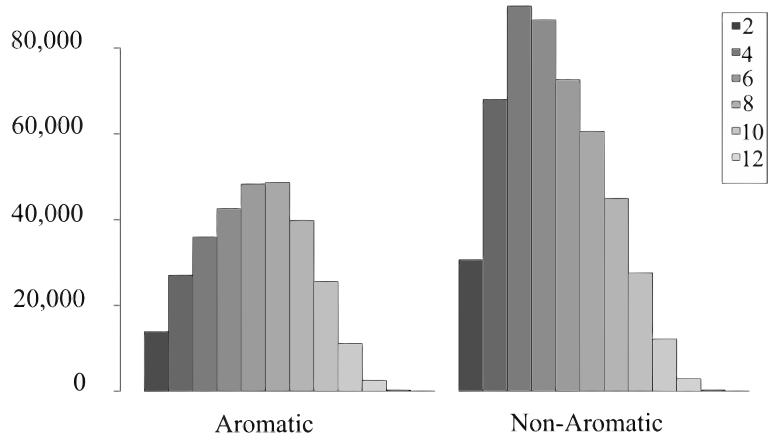


FIGURE 5.10: Backbone refinement class sizes for `AC-One` (stage 0)

with aromatic ring perception and about $10^{1.91} \approx 81.28$ [$10^{1.82} \approx 66.07$] without for `AC-One` (stage 0) [`AC-All` (stage 0)].

5.4.1.2 Representativeness

The higher coverage for the non-aromatic setting raises the question whether this is an expression of low representativeness, i.e. whether backbone refinement classes are smaller for the non-aromatic setting. In a second experiment, we assessed backbone refinement class sizes by putting every significant and frequent subtree into exactly one bin, where each bin gathers the subtrees with the same support, and then counting the trees per bin. The result is summarized by the histogram of Fig. 5.10. The frequency distribution of classes with different sizes shows clearly that the non-aromatic setting produces far more classes of the same size, and moreover, that mean class size is significantly lower (~ 5 patterns) than for the aromatic setting (~ 7 patterns). We obtained similar results for a minimum frequency of 100 (not shown).

5.4.2 Conclusions

In this section, we found that backbone refinement class representatives were indeed able to maintain a high coverage when minimum frequency was increased. This was different for the significant trees, whose coverage deteriorated much faster. The set of open and closed subtrees is a lossless compression of the latter. It still models all levels of frequency above minimum frequency, however. Likewise, they would more rapidly lose coverage with growing minimum frequency, too.

Consistent with the findings for medium-sized data, a lower minimum frequency did not increase coverage for the large-scale datasets. Here, backbone refinement class representatives were also very descriptive, given their coverage of > 40 for the minimum frequency 200, which was achieved with only about 30,000 patterns for about 23,000 compounds (see section 5.3). The results thus underpin the intuition from section 5.1.3, namely that backbone refinement class mining is robust in terms of coverage, allowing for fast mining in the search space implied by medium and high frequency thresholds.

The aspect of aromatic perception brought quite ambiguous results. Aromatic perception induced both lower coverage and higher representativeness than the non-aromatic representation. The fact that the higher frequency lowered the difference between mean and median in the aromatic setting indicates that a substantial set of patterns, which had previously covered only a small fraction of compounds, were removed (see the normality of the corresponding density curves in Appendix B). This observation is intuitively plausible, since patterns near the positive border are always the largest fragments considered, and here relatively many new, but sparsely populated backbone refinement classes may be created. In contrast, without aromatic perception, the difference between median and mean was growing.

On the other hand, the coverage without aromatic perception was found to be higher, while its representativeness was lower. Also, without aromatic perception, a higher total number of patterns are generated (note, however, that the compression experiments in section 5.3.5 were conducted without aromatic perception). This may be attributed to lower expressiveness of non-aromatic perception, where always two permutations exist to describe atoms and bonds in an aromatic ring. We note that the tradeoff induced by the aromatic *vs.* non-aromatic perception has to be investigated further with regards to their classification performance.

5.5 Diversity in Structure and Occurrence

So far, all presented properties were obtained by counting, not by looking at (pairs of) actual patterns and/or instances. In this section, we will experimentally try to verify the intuition of diversity of backbone refinement class representatives by investigating two more intrinsic properties.

We will start with *structural diversity*, which exploits their property of being graphs. From their definition, backbone refinement class representatives should

have quite diverse scaffolds. To verify this, we will investigate diversity by maximum common (induced) subgraphs that are shared between representatives.

Second, structural diversity should reflect in diversity of occurrences, i.e. structurally diverse subgraph patterns should describe different instances. I consider this important to prove correct the hypothesis that structural partitioning aids classification. Although we know from their definition that backbone refinement class representatives are highly target class correlated, the setting could still be unfortunate because we know not much about the diversity of the representation:

- Patterns could be highly co-occurring, i.e. pairs of patterns have highly similar occurrences.
- Groups of instances could have indistinguishable representations, i.e. are covered mostly by the same patterns.

Unfortunately, co-occurrences are generally not easy to convey, because the data is high-dimensional. For example, each pair of patterns and each pair of instance and pattern is considered. Thus, we will look at a low-dimensional presentation of the data, where distance represents co-occurrence.

5.5.1 Experiments

The datasets were represented without aromatic information and the thresholds for minimum frequency and correlation were the same as in section 5.3.5.

5.5.1.1 Structural Diversity

Backbone refinement class representatives were compared to the class of "α - *orthogonal*, β - *representative patterns*" introduced by Al Hasan *et al.* and their `ORIGAMI` approach [2] in terms of structural similarity.

Their approach is motivated by the idea of mining a small set of representative subgraphs that is supposed to summarize the frequent subgraphs. In the `ORIGAMI` algorithm, they approached the problem by sampling from the positive border, i.e. from among the maximal patterns, as induced by minimum frequency. They formalized their ideas by forcing the mined representatives to be structurally dissimilar, while maintaining a minimum similarity between any frequent subgraph to

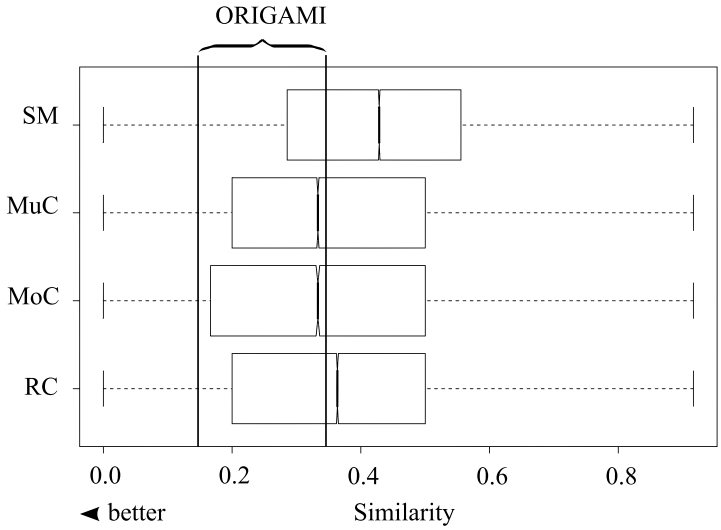


FIGURE 5.11: Pairwise pattern similarity distribution for the CPDB datasets.

at least one mined representative. Specifically, they defined the similarity between two subgraphs p and q as

$$sim(p,q) = \frac{|m_{pq}|}{max(|p|,|q|)}, \tag{5.5}$$

where m_{pq} is the maximum common (edge-induced) subgraph of p and q and $|x|$ is the number of edges of x. Therefore sim is the size of the maximum common subgraph relative to the larger graph.

Al Hasan *et al.* investigated α-orthogonal patterns with pairwise similarity values with $\alpha \in [0.15, 0.35]$. Running times reported in their analysis were $\approx 2750s$ $(= 45m48s)$ for a set of 1000 patterns obtained from a database of 1084 compounds using a minimum frequency threshold of 2.3 % (pp. 9-10).

To assess the structural diversity of backbone refinement class representatives, their similarity was measured *post hoc*, according to the definition of sim. Fig. 5.11 shows boxplots of the pairwise pattern similarity distribution for the four CPDB datasets. It gives the median (bold bar in the middle), the upper and lower quartile (box boundaries, comprising 50 % of the data) and therefore also the inter-quartile distance. The boxplots show that, except for the `SM` dataset, backbone refinement class representatives had a median similarity of < 0.4, and that, in two cases, median values were inside the `ORIGAMI` interval. The running time for mining the patterns on the four datasets were 0.22s, 0.34s, 0.24s, and 0.37s for mining 519, 413, 263, and 447 patterns, respectively.

5.5.1.2 Co-Occurrence and Entropy

This section uses the visualization method for relational datasets by Schulz *et al.* to embed patterns and instances into a 2-D plane [55]. The embedding methodology originates from *ILP (inductive logic programming)*. There, clausal theories are checked for fulfillment in logical interpretations. Here, however, the setting is transformed to patterns (backbone refinement class descriptors) occurring in instances (molecules). This representation may yield insights into the relational structure of the data, especially about the quality of the pattern representation.

The goal of the embedding is to provide the human user with easily understandable insights into the relations "pattern-covers-instance" and "pattern-occurs-with-pattern". More specifically, in the 2-D representation patterns and instances are points, and their coordinates are influenced by:

1. Pattern-pattern co-occurrence: Co-occurring patterns are placed close to each other.
2. Pattern-instance co-occurrence: Instances are placed close to the patterns by which they are covered. Moreover, the lower the entropy of a pattern, the closer a covered instance is placed to it.

Thus, the image visualizes high-dimensional data by Euclidean distance. Besides representation as an image, it also provides an interactive layer (see Appendix C).

The main characteristics of the embedding approach [55] are described as follows. For the pattern-instance co-occurrences, an instantiation matrix C is used, where $C_{PI} = 1$ if pattern P occurs in instance I. Then, the joint probability of observing P in I is

$$Pr(P, I) = C_{PI} / \sum_P \sum_I C_{PI} \tag{5.6}$$

Note that in our setting, $Pr(P, I)$ is uniform for all patterns P that occur in I. The embedding approach then relates distance $d_{P,I}$ between P and I to statistical dependency by Gaussian decay:

$$\frac{Pr(P, I)}{Pr(I)} \propto e^{-d_{P,I}^2} \tag{5.7}$$

Note the cancellation of the marginal $Pr(I)$. It then maximizes the probability $Pr'(P, I) = \frac{1}{Z} Pr(I) e^{-d_{PI}^2}$, where Z is a normalization factor ensuring that Pr' is a probability distribution. The maximization is done with a gradient ascent method using random restarts.

However, to account for entropy of the patterns, the negative entropy of each pattern is multiplied on the associated entries C_{PI} for all i prior to maximizing Pr', which also removes uniformity. The entropy of a pattern P is defined as

$$H(P) = -\sum_{C\in\{0,1\}} Pr(C,P)\ log\ Pr(C,P), \tag{5.8}$$

and $Pr(C,P)$ is the empirical (induced) probability of P covering an instance with target class C. Finally, pattern-pattern co-occurrences are integrated into Pr' in a way similar to equation 5.7 where a second pattern takes the role of I, weighted by a constant factor.

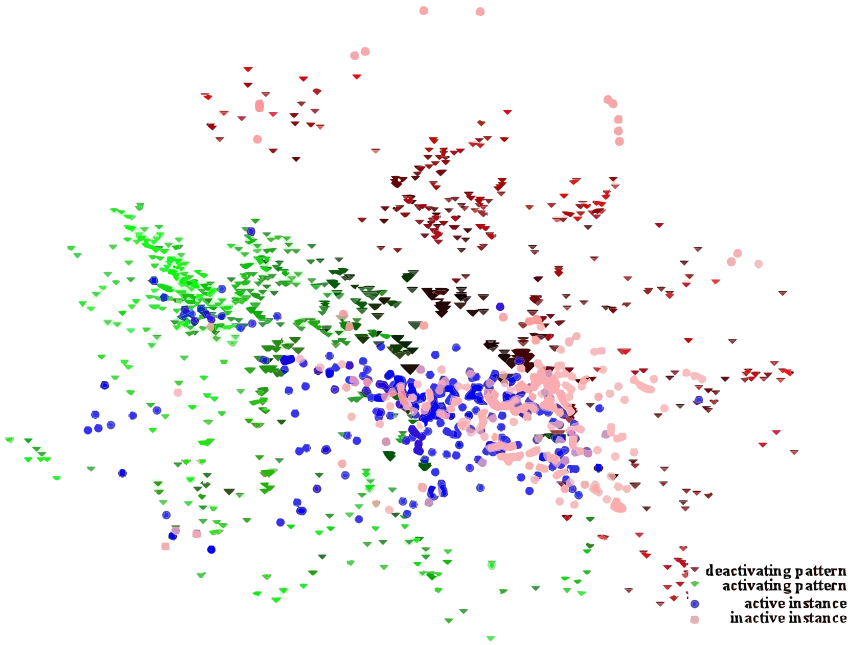
Fig. 5.12 shows the Euclidean embedding: The data points consisted of backbone refinement class representatives (triangles) and molecules (circles), the distances were found by locally optimizing the log-likelihood of the data.

The lower the entropy of a pattern the brighter was its color (green for active, red for inactive), indicating its potential to discriminate between classes. The corresponding instance colors are blue for active, salmon for inactive.
For the `SM` dataset, the patterns were in general well distributed across the plane, with few clusters. Instances and patterns did not overlap much. Instead, the majority of patterns surrounded the instances. The activating patterns were isolated in the lower left half space.
For the `MuC` dataset, there were a few homogeneously colored pattern clusters, but the majority of patterns were well distributed. Instances and classes did not overlap much, but patterns surrounded instances. The activating patterns were isolated in a convex subspace on the lower right.

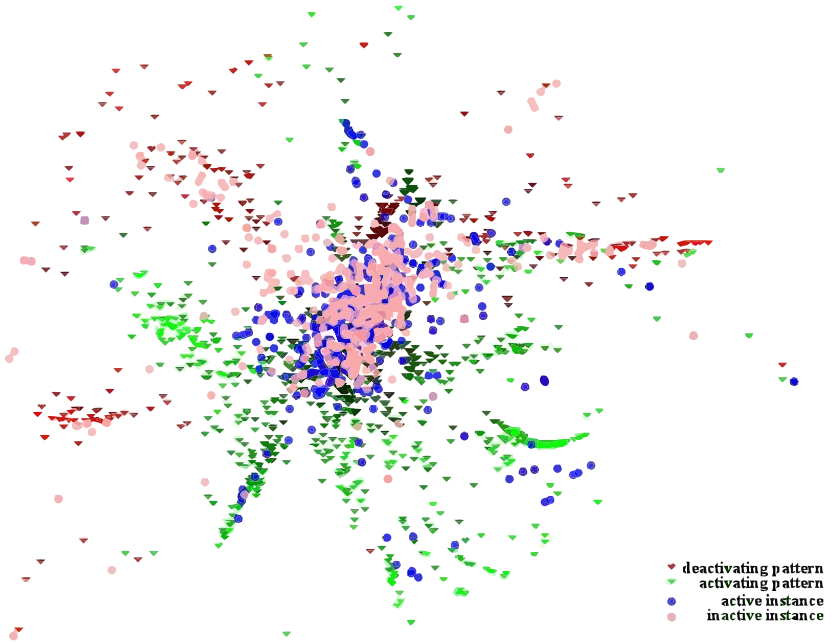
5.5.2 Conclusions

The structural diversity of backbone refinement class representatives, obtained by a simple structural criterion, has been shown to translate in diverse occurrences in this chapter. This makes backbone refinement class representatives interesting candidates for descriptors in classification tasks. The results have underpinned those of section 5.4, namely that relatively low coverage may still yield a usable representation.

For the structural similarity, the tradeoff between mining time and structural diversity has been found favorable. An important factor is that larger graph structures are more likely to be dissimilar than smaller ones. Given that backbone refinement class representatives are not generally found among the largest frequent



(a) *Salmonella Mutagenicity* (SM)



(b) *Multicell Call* (MuC)

FIGURE 5.12: Euclidean embedding of backbone refinement class representatives and instances of the graph database based on co-occurrence and entropy towards the target classes.

subgraphs (as opposed to the maximal patterns considered by ORIGAMI) their structural diversity thus seems high. Compared to the latter, however, backbone refinement class representatives can be derived efficiently enough to be even used in "on demand"-settings, i.e. in ad-hoc situations.

The 2-D embedding combined occurrences and entropy towards the target classes into a graphical model, in that it placed co-occurring patterns close to each other and instances close to associated patterns (the lower the entropy of patterns, the closer). However, instances and patterns have appeared quite separated from one another in the experiments. More specifically, there were few clusters of instances and associated patterns. This indicates low co-occurrence between instances and patterns, and thus different instances should be described in a distinguishable way. Next, disregarding instances, also patterns appeared well-distributed, which indicates low co-occurrence between patterns. Therefore, pairs of patterns seem to have mainly dissimilar occurrences.

5.6 Runtime Analysis

We now investigate empirically how efficiently backbone refinement class representatives can actually be mined. The method builds on the principles of correlated pattern mining outlined in chapter 3. Actual running times are assessed experimentally, using several differently aggressive pruning methods.

5.6.1 Dynamic Upper Bound Pruning

Corresponding to section 3.3.1, the χ^2 distribution test (checking the adherence of a variable to a given distribution) was used.

Definition 5.6 (Dynamic Upper Bound Pruning)**.** For any frequent subtree t, let χ^2_t and $\chi^2_{u,t}$ denote the χ^2 value for t, and χ^2 upper bound for refinements of t, respectively. Let $u_{max}(t) = max\{\chi^2(t', R) \mid t' \preceq_b t\}$, the maximal χ^2 value seen so far. Then, if $u_{max}(t) > u$, the user-defined upper-bound threshold u may be increased to $u_{max}(t)$, since we only search for the maximal class element.

The method for mining backbone refinement class representatives using dynamic upper bound pruning is shown in algorithms 1 and 2.

The procedure is invoked as $p.init()$ for any path p of length 1. As can easily be seen, procedure $init()$ starts the search procedure $expand()$ with the longest,

Algorithm 1: *init*():
Calculation of paths as candidate backbones.

Input: A global, user defined significance threshold u; a global, user defined minimum frequency m
Output: Immediate path refinements of the current structure that satisfy the minimum frequency constraint.

```
1: for all q' ∈ this.ImmediatePathRefinements do
2:    if q'.frequency > m then
3:       q'.init()
4:    end if
5: end for
6: for all q' ∈ this.ImmediateTreeRefinements do
7:    if q'.frequency > m then
8:       q'.expand(q', χ²_q')
9:    end if
10: end for
```

i.e., non-path-refineable, paths and subsequently backtracks to the shorter paths. Thus, every path p satisfying the user defined minimum frequency serves as a backbone once.

The tree search procedure *expand*() in algorithm 2 works as follows: Lines 1-4 output the maximal element, if no further refinements are available. Otherwise, for every refinement on the same level, a new class is instantiated for every refinement q'. In line 6 $cmax$, the maximal χ^2 value seen so far (including q') is calculated. Line 7 then implements dynamic upper bound pruning by checking q''s upper bound value against $cmax$.

- If the upper bound value is lower, the search is truncated and the maximal pattern at that point is output (lines 15-18).
- Otherwise, if the upper bound value is not lower, the bound is updated and the iteration continues.

Therefore, once a subtree is output as backbone refinement class representative, all the refinements that are in backbone refinement relation to it are pruned away. Note: the *print*() routine checks (not shown) that the pattern to be printed has an absolute χ^2 value of at least u wherever it is called, since this property is in general not guaranteed.

Algorithm 2: $expand(q_{max}, \chi^2_{q_{max}})$:
Implementation of backbone refinement class mining via dynamic upper bound pruning.

Input: A tree q_{max} with significance $\chi^2_{q_{max}} > u$ above the global user defined significance threshold u, a global user defined minimum frequency m, a global variable $updated = true$
Output: The most significant backbone refinement of q_{max}.

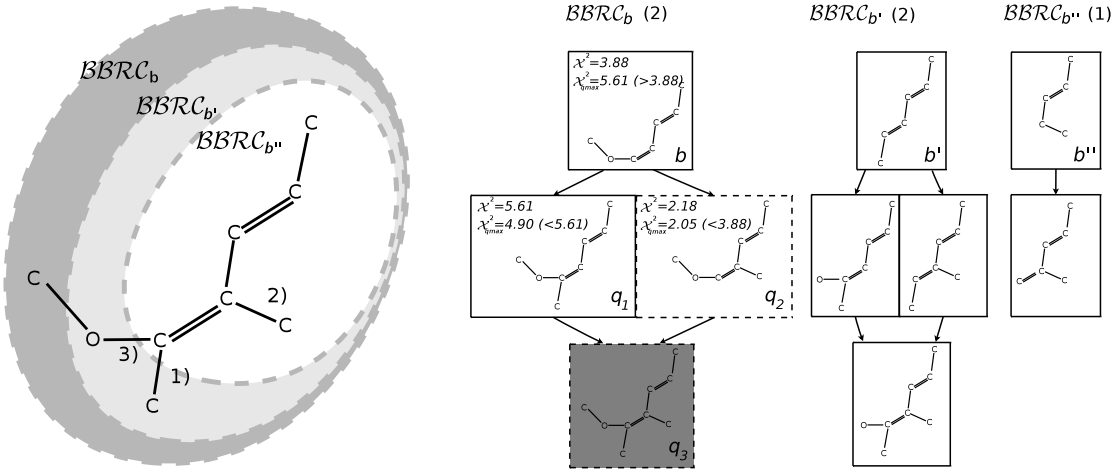
```
1: if this.ImmediateTreeRefinements == ∅ { then
2:    print(q_max)
3:    updated=false
4: end if
5: for all q' ∈ this.ImmediateTreeRefinements do
6:    cmax = max(χ²_q', χ²_q_max)
7:    if χ²_u,q' ≥ cmax ∧ q'.frequency > m then
8:       if χ²_q_max < χ²_q' then
9:          q_max = q'
10:         χ²_q_max = χ²_q'
11:         updated = true
12:      end if
13:      q'.expand(q_max,χ²_q_max)
14:   else
15:      if updated then
16:         print(q_max)
17:         updated = false
18:      end if
19:   end if
20: end for
```

5.6.2 Implementation

We modified the graph miner `Gaston` [44] to support backbone refinement class mining[4]. Two specific properties allow for an efficient implementation on top of `Gaston`:

- As discussed in section 2.2, modern graph miners enumerate subgraphs canonically. `Gaston` uses a canonical code representation for graphs that enables checking for allowable path and tree refinements in constant time. Specifically, no refinement is enumerated twice (for details, see the work of Nijssen and Kok [44, 45]).

[4]We used version 1.1 (with embedding lists), see `http://www.liacs.nl/~snijssen/gaston/`.



(a) q_2

(b) Paths b, b', b'' and associated BBRCs.

FIGURE 5.13: Left: Backbone Refinement Classes for q_2 from Fig. 5.1 with numbered edges 1), 2), and 3). Dashed borders indicate their boundaries due to condition 1. from definition 5.1. Right: Associated search space for non-refineable paths b, b', and b''. Insignificant (95% ~ 3.84) nodes are dashed, pruned nodes are gray.

- `Gaston` first enumerates all frequent path refinements $\mathcal{P}_{|R}$, and only thereafter starts enumerating all tree refinements $\mathcal{T} \setminus \mathcal{P}_{|R}$ growing from all $p \in \mathcal{P}_{|R}$. This is performed by using p as backbone of the innate tree, i.e. by prohibiting backbone changes while applying tree refinements recursively.

To allow for efficient pruning, backbone refinement classes are kept disjoint once they have branched, i.e., given two backbone refinement classes A and B with patterns $a \in A$ and $b \in B$, with $a \not\preceq b$ and $b \not\preceq a$, every c with $a \preceq c$ and $b \preceq c$ is put either in A or B, whichever backbone refinement class is enumerated first. For example, q_3 would be either assigned to the class of q_1 or q_2 in Fig. 5.1. However, any pattern more general than the branch point can represent A or B or both.

5.6.3 Example Session

Fig. 5.13 visualizes the refinement process for tree q_3 by continuing the example from Fig. 5.1. Fig. 5.13(a) indicates backbone refinement class boundaries due to condition 1. from definition 5.1 by dashed borders, while Fig. 5.13(b) shows how the three different non-refineable paths in q_3 are used as backbones while the structure is explored. Backbone refinement class sizes are given in brackets. Specifically,

- $BBRC_b$ contains two backbone refinement classes corresponding to the exclusive inclusion of either edge 1) or 2), $BBRC_{b_1}$ and $BBRC_{b_2}$, with 4 subtrees in total: $BBRC_{b_1} = \{b, q_1, q_3\}$, $BBRC_{b_2} = \{b, q_2, q_3\}$.
  We detail the expansion and pruning process for $BBRC_b$. Note the associated χ^2 and upper bound ($\chi^2_{q_{max}}$) values in Fig. 5.13(b). Pattern b has a χ^2 value of 3.88, and no refinement of b can have a χ^2 value > 5.61. Assume that b is the most discriminative pattern seen so far. Then, $\chi^2_{q_{max}} = 3.88$. Assume further that the original threshold given by the user was $u = 3.84$ ($\approx$ 95% significance for 1 degree of freedom). Since $5.61 > 3.88$, we expand b. Subtree q_1 is expanded before q_2, since it has the lower canonical depth sequence. It has a χ^2 value of 5.61, increasing the upper bound threshold $\chi^2_{q_{max}}$ to that value, thus making q_1 representative for $BBRC_{b_1}$. However, it has an upper bound value of 4.90, which is below the current threshold of 5.61. Thus it is not expanded. Subtree q_2, the right child, has a χ^2 value of 2.18. Since it is not in backbone refinement relation to the left child, it initiates a new class. However, it cannot be representative of that class, since b has the higher χ^2 value. We therefore mark it with a dashed border (even if b had a lower value, it could not be a representative, since its significance is below u). Thus, b will be the current representative for $BBRC_{b_2}$. Also, it has an upper bound value of 2.05, which is below the current threshold of 3.88. Thus it is not expanded (additionally, in our implementation, q_3 would have already been considered as refinement of q_1 and not be enumerated a second time). In summary, q_3 (gray) will never be reached, due to dynamic upper bound pruning. More specifically, although it could be a significant pattern judging from the q_1 position (its χ^2 value could be > u), the upper bound threshold has increased too far already, making q_1 the final representative for $BBRC_{b_1}$ already at that point. For $BBRC_{b_2}$, the representative is b.

- $BBRC_{b'}$ contains two backbone refinement classes corresponding to the exclusive inclusion of either edge 1) or 3), $BBRC_{b'_1}$ and $BBRC_{b'_3}$, with 4 subgraphs in total.

- $BBRC_{b''}$ contains a single backbone refinement class with 2 subgraphs.

The class members are enumerated by subjecting paths b, b', and b'' from Fig. 5.13(b) to algorithm 2. After backtracking, the same concept is applied to the other (shorter) paths in q_2.

			SM		**RC**		**MoC**		**MuC**
	APL to	s	%	s	%	s	%	s	%
None	1,2,3,4	2.63	100.00	21.23	100.00	3.71	100.00	5.17	100.00
Static	2,3,4	2.55	96.97	21.11	99.43	2.98	80.32	4.76	92.07
Dynamic	4	0.44	16.73	6.63	31.22	2.13	57.41	1.76	34.04

TABLE 5.5: Comparison of running time for mining Backbone Refinement Class Representatives using different pruning techniques for the four CPDB datasets.

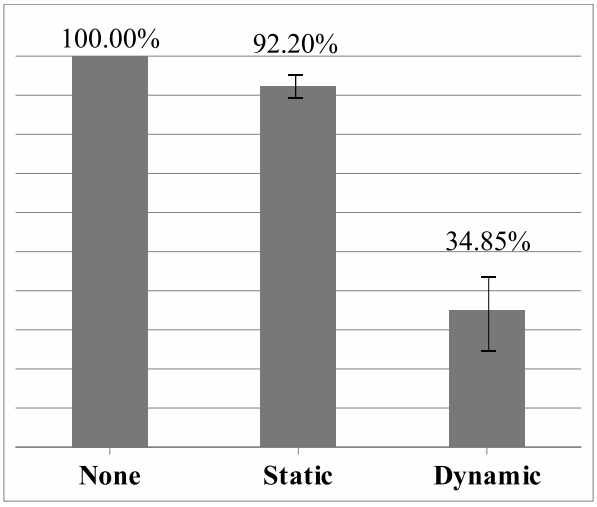


FIGURE 5.14: Mean values of Table 5.5, taken for the four CPDB datasets.

5.6.4 Experiments

Again, the datasets were represented without aromatic information and the thresholds for minimum frequency and correlation were the same as in section 5.3.5.

5.6.4.1 Runtime

For fixed minimum frequency and significance thresholds, Table 5.5 compares mining performance of backbone refinement class representatives (4) on the one hand to the same subgraph descriptors as in section 5.3.5, namely linear fragments (1), significant trees (2), and open trees (3) on the other hand. It indicates the mining times we obtained for BBRC representatives with the different statistical metric pruning techniques that apply to the different fragment types (column "APL to"). More specifically, those times correspond to backbone refinement class mining using no statistical pruning, static upper bound pruning and dynamic upper bound

pruning, respectively. Mining times for open trees using `sfgm` were as follows: 1,899 s (`SM`), 28,537 s (`RC`), 1,744 s (`MoC`), and 2,594 s (`MuC`).

Fig. 5.14 summarizes the information from Table 5.5 by mean values of relative running time reduction across the different datasets. The application of dynamic upper bound adjustment was associated with a reduction in running time by 63.34 % and 60.92 % compared to using no statistical pruning and static upper bound pruning, respectively. The "no pruning" setting corresponds to ordinary fragment search with only minimum frequency as anti-monotonic constraint, i.e. to the original `Gaston` implementation with significance value calculation added.

5.6.4.2 Profiling

To assess the amount of overhead incurred by our method, running time analysis was performed by comparing its profile to that of the original `Gaston` algorithm. Here, the OProfile system profiler for Linux recorded how much time each method spent carrying out a certain function. The results indicate that our algorithm was about 6 % of the time concerned with χ^2 and upper bound calculations, and about 3 % with additional control overhead due to the more sophisticated *expand*() routine.

5.6.5 Conclusions

Dynamic upper bound pruning, as applied to backbone refinement class mining, incurs a drastic decrease in runtime, as compared to ordinary (static) upper bound pruning. The key to the performance gain is to raise the threshold for the upper bound of refinements of the current pattern. It can only be raised so quickly, however, due to the top-1 mining performed inside each backbone refinement class. The more general concept of top-k mining uses the same principle, see the work by Bringmann *et al.* [5, 6]. Here, the threshold can only be raised to the value of the k-th best pattern. Thus, the larger k, the smaller the effect. Since $k = 1$ in our case, backbone refinement class mining is optimal in this respect.

The structural partitioning of the search space into backbone refinement classes enables the approach of top-1 mining discussed in the previous paragraph. The `Gaston` algorithm delivers this partitioning "for free", due to the "quickstart principle", which grows trees from backbones only. Since running time gains of over 60 % were obtained, the additional effort for the more sophisticated expand routine seems fully justified.

The set of significant trees cannot use dynamic upper bound pruning, because it does not offer the possibility for top-k mining. Thus, static upper bound pruning applies here. The same holds for the set of open trees- mining open trees merely compresses the result set, but offers no further gain in efficiency compared to the complete set (see the work of Bringmann *et al.* [6]). Finally, for the set of all frequent subtrees, no upper bound pruning is possible. Performance comparisons in this class-blind setting, e.g., to gSpan, can be found in the literature [45, 65].

5.7 Classification Accuracy

This section compares *accuracy*, *sensitivity* and *specificity* of the different descriptors discussed so far in a binary classification task. Sensitivity (specificity) measures the proportion of target class positives P (target class negatives N) which are correctly identified as such (TP and TN, respectively). Accuracy is the overall fraction of correct predictions, calculated as $|TP| + |TN|/(|P| + |N|)$.

Obviously, when evaluating subgraph descriptors in a classification task, the classifier should be able to handle large amounts of subgraphs without suffering from the high-dimensional pattern space. As discussed earlier, support vector machines, nearest-neighbor methods, or decision trees are among the suitable algorithms for this setting. The nearest-neighbor method described in section 3.3.2 was used.

5.7.1 Experiments

The figures reported in this section were derived as follows. For any instance x,

1. *ALL* includes all unweighted predictions, denoted $f(x)$ (*cf.* eq. 1.1),
2. *AD* (or *Applicability Domain predictions*) considers the top 80 % unweighted predictions as ranked by confidence $conf(x)$ [21], and
3. *WT* includes again all predictions, but this time the contribution of every prediction is weighted by its associated confidence value. In this case, accuracy is calculated as

$$\frac{\sum_{x \in TP \cup TN} conf(x)}{\sum_{x \in P \cup N} conf(x)} \tag{5.9}$$

The rationale for the *WT* measure is that errors of high-confidence predictions should be penalized more heavily than errors of low-confidence predictions. Therefore, *WT* aggregates both types of information into one measure. Again, the datasets were represented without aromatic information and the thresholds for minimum frequency and correlation were the same as in section 5.3.5.

5.7.1.1 Evaluating Different Representations

Section 5.4 revealed some issues with several ways to represent the data: when aromatic perception was used, the representativeness increased strongly, while the number of patterns shrank. This was in contrast to minimum frequency, which did not have any pronounced effect. To clarify the effects on predictive performance before comparing backbone refinement class representatives to other types of descriptors, both parameters were varied at the same time and the effects were systematically assessed. The results from our previous study [41] showed that *WT* was the most stable and meaningful in validation, thus this validation method was also used here. The plots in Fig. 5.15 reveal that

- *Reduced* generally performed sub-optimal.
- Classification accuracy could be maintained at a high level even for high minimum frequencies for *Kekulé* (e.g. the rightmost point denotes a minimum frequency of 10 % of the dataset size).
- *Aromatic* was associated with more jitter and non-monotonicity then *Kekulé*. However, for very low frequencies it often performed better than aromatic representation.

In view of these results (no clear winner) and since the data could be most conveniently represented in *Kekulé* notation for `sfgm` (see section 5.7.1.2), this representation was chosen to compare backbone refinement class descriptors against other representations.

5.7.1.2 Validation Against Compressed Representations

Table 5.6 compares the accuracy values of backbone refinement class representatives to linear fragments, significant trees, and open trees. As discussed in section 5.7.1.1, Kekulé representation was chosen. Additionally, a minimum frequency of 6 was selected. The results show that tree-shaped subgraphs always performed

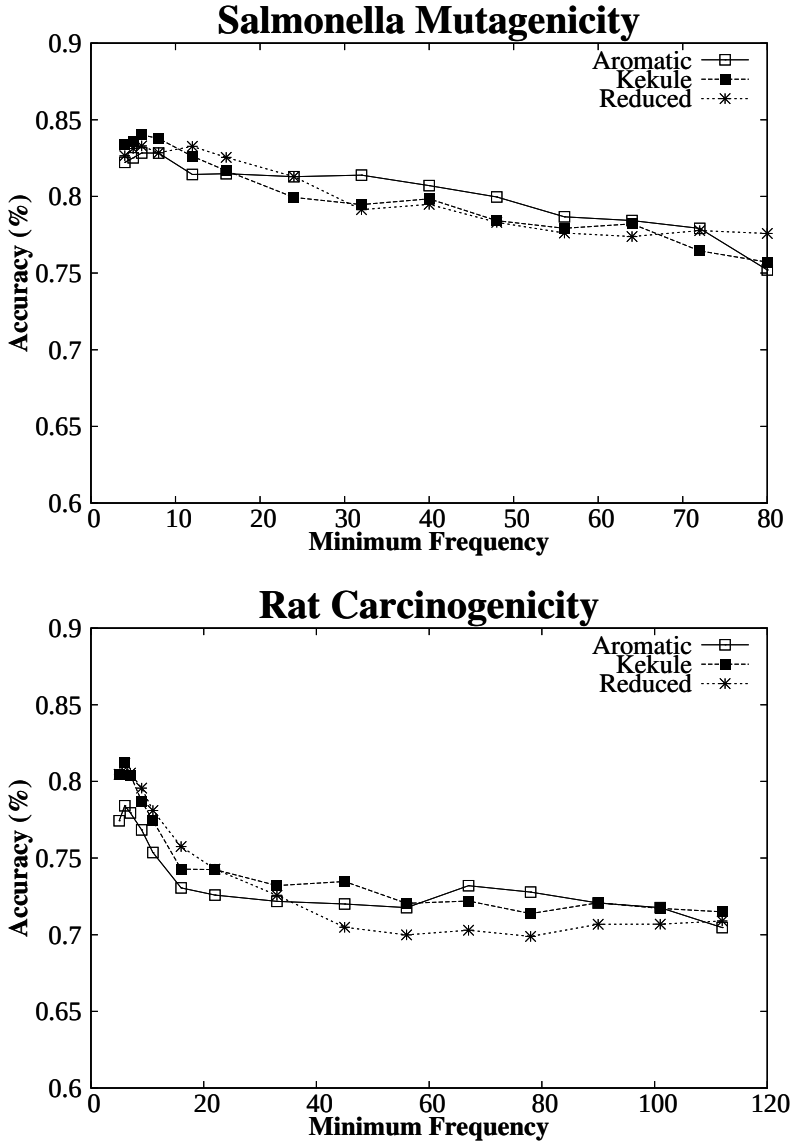


FIGURE 5.15: Validation against different representations using *WT* validation method, see section 5.7.1.

better than linear subgraphs, whether for all or AD predictions or for weighted accuracy. Backbone refinement class representatives outperformed open trees in 10 out of 12 cases. The mean accuracy difference between backbone refinement class representatives and significant trees was -0.27 ±1.47, whereas it was 1.1 ± 1.44 compared to open trees and 2.77 ± 1.66 compared to linear fragments, respectively. A paired *t*-test on the accuracy values revealed that backbone refinement class representatives performed significantly better than open trees (t = 2.65, df = 11, p-value = 0.02267), while no significant difference between backbone refinement class representatives and the complete set of trees (t = 0.65, df = 11, p-value = 0.5302) was found.

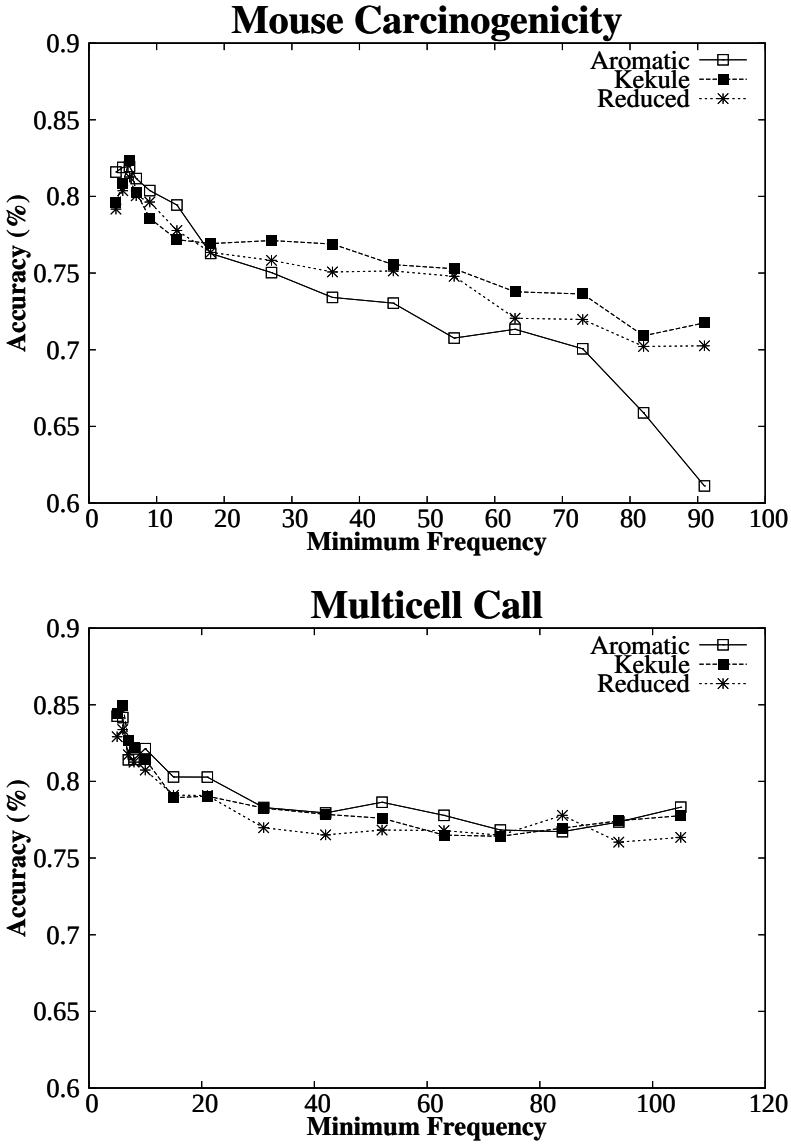


FIGURE 5.15: Validation against different representations using WT validation method, see section 5.7.1.

Fig. 5.16 compares the four different types of descriptors in ROC space, showing the differences in sensitivity and specificity for the prediction of active compounds. There is a trend for better values when tree-shaped fragments were used, clearly signaling the higher information content present in those descriptors. Backbone refinement class representatives seemed to exhibit a lower false alarm rate compared to open trees. Indeed, a paired t-test on the false positive ratios confirmed that backbone refinement class representatives significantly improved on specificity ($t = -4.60$, $df = 11$, p-value $= 0.00077$). A significant difference in sensitivity could not be detected. Test results were confirmed with Wilcoxon signed rank tests [64].

	SM			RC		
	ALL %	*AD* %	*WT* %	*ALL* %	*AD* %	*WT* %
1. Linear Fragments	75.0	77.8	83.0	63.7	67.0	77.5
2. Significant Trees	74.6	**80.7**	**86.8**	64.4	70.0	81.8
3. Open Trees	**75.5**	80.6	84.5	64.5	68.7	80.0
4. BBRC Representatives	74.6	79.4	85.4	**67.2**	**70.4**	**82.2**
	MoC			MuC		
	ALL %	*AD* %	*WT* %	*ALL* %	*AD* %	*WT* %
1. Linear Fragments	67.6	72.7	79.9	69.3	72.4	79.6
2. Significant Trees	**73.3**	75.7	**83.7**	**71.9**	**75.6**	83.5
3. Open Trees	71.5	74.4	80.8	70.2	73.5	81.3
4. BBRC Representatives	71.7	**76.5**	82.0	70.3	74.1	**84.9**

TABLE 5.6: Accuracy table for the CPDB datasets, obtained with leave-one-out crossvalidation. Bold figures indicate the best results.

5.7.1.3 Validation Against Supervised Selection

We also evaluated backbone refinement class representatives on the three datasets from the study of Rückert and Kramer [49]. Their work frames the selection of pattern sets suitable for classification as a combinatorial optimization problem based on entropy (*cf.* section 5.5.1.2). Define conditional entropy of pattern p as

$$H(P|C) = -Pr(C,P)\ log\ Pr(C,P), \tag{5.10}$$

where $Pr(C,P)$ is the empirical (induced) probability of P covering an instance with target class C. Consequently, the entropy of pattern P is

$$H(P) = -\sum_{C\epsilon\{0,1\}} Pr(C,P)\ log\ Pr(C,P). \tag{5.11}$$

Similar to Eq. 5.10, the mutual pattern entropy is defined as

$$H(P_i|P_j) = -Pr(P_i,P_j)\ log\ Pr(P_i,P_j), \tag{5.12}$$

where $Pr(P_i,P_j)$ is the probability of P_i and P_j both covering an instance. The approach combines the three measures by looking for high correlation of patterns

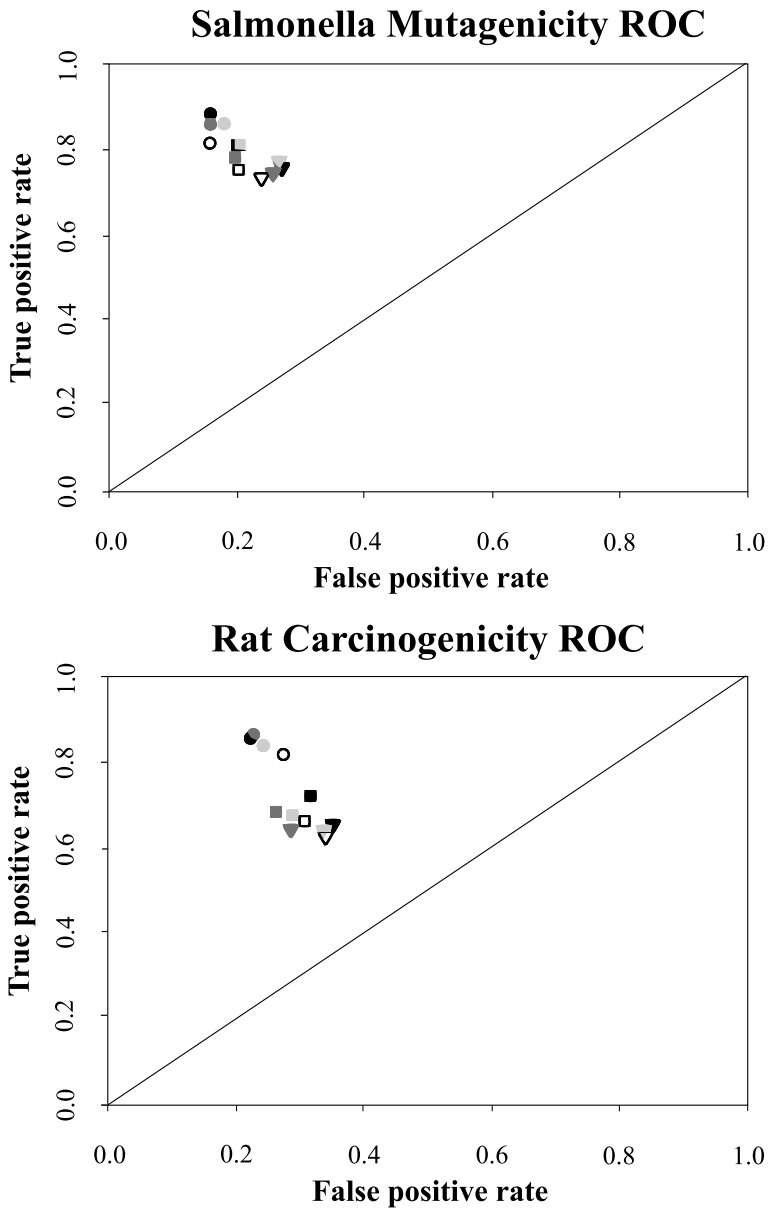


FIGURE 5.16: ROC plots for the CPDB datasets comparing Significant Trees (black), Backbone Refinement Class Representatives (dark gray), Open Trees (light gray) and Linear Fragments (hollow). Circles denote predictions weighted by confidence, squares predictions in applicability domain, and triangles all predictions.

towards the target classes (which is equivalent to low entropy $H(P|C)$), high pattern entropy $H(P)$, as well as high inter-pattern entropy $H(P_i|P_j)$ for all pairs of patterns i, j into a common measure, the so-called *dispersion score*. Then, using stochastic local search (*SLS*), i.e. a forward selection method with random restarts, it assembles a set with the k highest scoring patterns it encounters, where k is set by the user. The approach is applied as a post-processing step and is not

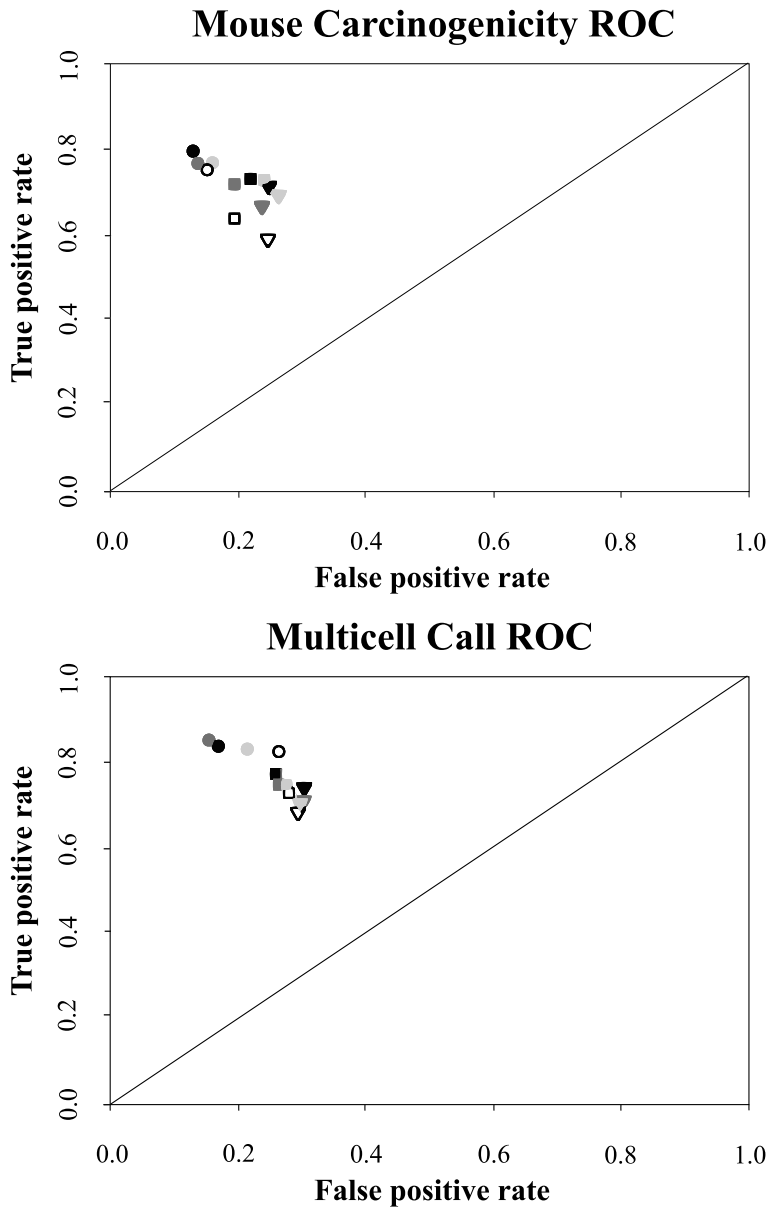


FIGURE 5.16: ROC plots for the CPDB datasets comparing Significant Trees (black), Backbone Refinement Class Representatives (dark gray), Open Trees (light gray) and Linear Fragments (hollow). Circles denote predictions weighted by confidence, squares predictions in applicability domain, and triangles all predictions.

limited to subgraph descriptors, but is applicable to general sets.

To render the results as comparable as possible, we also used 6 % of the dataset size as minimum frequency threshold and performed 10-fold crossvalidation. With that, we obtained the accuracies shown in Table 5.7. It also displays the results the original authors obtained using a special kernel model and that they reported

	ALL %	*AD* %	*WT* %	#P	t s	SLS %	*k*
NCTRER	0.74	0.79	0.82	1782	0.65	0.82	150
Bloodbarr	0.72	0.75	0.84	616	0.22	0.75	150
Yoshida	0.55	0.59	0.58	377	0.16	0.68	150

TABLE 5.7: Accuracy table for datasets used in the study of Rückert and Kramer, obtained with 10-fold crossvalidation.

as having "good predictive performance for all pattern set sizes" (MMV with the class-correlated dispersion score).
Apart from the `yoshida` dataset, the results seem to be competitive to the respective figures from the original publication. In particular within the applicability domain (AD), results were very similar to those of the *SLS* method. The table also gives the mean number of patterns and the running time for pattern calculation per fold. In terms of running times, the method proposed here is much faster, as the construction of the trie for the *SLS* method typically takes a few minutes, whereas the *SLS* run itself may take hours.

5.7.1.4 Validation Against Large-Scale Data

Crossvalidation runs were also conducted with the large-scale datasets from section 5.3.5. The mining settings were the same, i.e. involved the same compound selection, aromatic perception and a minimum frequency of 200. A prediction included the derivation of the training set similar to the query instance based on patterns occurring in the compounds and the calculation of the prediction itself.

In accordance with the findings in section 5.3.5, backbone refinement class representatives turned out to be the only practically useful pattern type for validation on the (quite powerful) computer we used. For the significant trees, the pattern set was way too large to be even read into RAM. With open trees, we obtained impractical prediction times of > 60 s, whereas backbone refinement class representatives gave a mean of 4.7 s and 11.1 s, respectively. Also, RAM usage was unacceptable with open trees. Table 5.8 shows the accuracy results we obtained in our validation with backbone refinement class representatives.

For `AC-All` (stage 1), the extraction of open trees with `sfgm` took > 10h, whereas backbone refinement class representatives took 1 m 13 s. For `AC-One` (stage 0),

	ALL %	*AD* %	*WT* %
`AC-One` **(st. 0)**	68.4	71.9	78.3
`AC-All` **(st. 1)**	67.7	71.2	77.5

TABLE 5.8: Validation results for large-scale datasets `AC-One` (stage 0) and `AC-All` (stage 1).

`sfgm` terminated with an error (see Table 5.2), while backbone refinement class representatives took 4 m 52 s.

5.7.2 Conclusions

In the experiments, the different representations revealed a slight advantage for Kekulé representation, since the aromatic representation seemed more unstable, showing a non-monotonic behavior in classification accuracy when minimum frequency was raised. Generally, the results underpin robustness towards increasing minimum frequencies, which has been already confirmed in section 5.4.

When compared to other types of descriptors, the classification accuracy of backbone refinement class descriptors was not only similar to that of the complete set of significant trees, but significantly better than that of the other compressed representations we investigated.

For large-scale data, backbone refinement class representatives were even the only practically useful descriptor type. Note that we deliberately selected a model that is able to handle high-dimensional pattern spaces and does not prefer or work better with sparse descriptors such as backbone refinement class descriptors. As an *ad-hoc* method, the selection of patterns obtained by backbone refinement class mining is not "tuned" towards classification in the sense of a post-processing step on the result set. Still, the method seems competitive to pattern sets composed in such a supervised fashion for classification purposes. Contrary to such approaches, however, backbone refinement class mining does not try to approximate an NP-hard problem and can thus run in seconds or even fractions of seconds, whereas the other approaches may take hours to run.

Chapter 6

Latent Structure Pattern Mining

This chapter treats *Latent Structure Pattern Mining*, which fully automatically converts a large set of basic fragment descriptors (obtained by frequent and correlated subgraph mining) into a condensed set of weighted, elaborate, chemically interpretable patterns, containing previously hidden (or latent), essential motifs present in the original set.

Section 6.1: Introduction. Discusses work related to elaborate pattern mining, in which, however, no fully automatic approaches have been proposed so far. This seems surprising, since the intuitively appealing idea of aligning basic fragments is also computationally feasible. A pipelined approach for latent structure calculation is presented.

Section 6.2: Latent Structure Pattern Mining. A strategy to integrate latent structure pattern mining into any depth-first branch-and-bound algorithms is developed. It exploits the partial order that the subgraph relation induces and solves situations where basic patterns provide conflicting information. Suitable "adjacent" basic fragments are chosen to equally contribute to a latent pattern, which limits the selection in a sensible way. Aligned basic fragments induce a weighted edge graph, from which the latent information is extracted by means of spectral analysis.

Section 6.3: Algorithm. A detailed account of the proposed algorithm elaborates on the merging strategy and on computational effort.

Section 6.4: LAST-SMARTS. Three different strategies for converting latent structures to so-called *SMARTS* patterns, which are commonly used in chemoinformatics to describe substructures, are evaluated.

Section 6.5: Experimental Evaluation. LAST-PM descriptors are compared to other complex and elaborate chemical fragment representations in terms of classification accuracy using crossvalidation. They are also contrasted against QSAR models based on physico-chemical descriptors, which is especially interesting regarding the complex biological endpoints used. The tradeoff between reduction of the resulting pattern set and runtime that LAST-PM faces is examined.

6.1 Introduction

Graph mining algorithms have focused almost exclusively on basic patterns (so-called *ground features*, subgraphs that are frequent and significantly correlated with the target classes) so far, such as frequent or correlated substructures. In the biochemical domain, Kazius *et al.* [33] have demonstrated the use of more elaborate patterns that can represent several ground features at once. Such patterns bear the potential to reveal latent information which is not present in any individual ground feature. So far, their identification requires expert knowledge [33] or extensive pre-processing of the data (annotating certain nodes or edges by wildcards or specific labels) [22].

6.1.1 Related Work

Latent structure pattern mining allows deriving basic motifs within ground features. The approach falls into the general framework of graph mining.

Kazius *et al.* [33] created two types of (fixed) high-level molecule representations (aromatic and planar) based on expert knowledge. These representations are the basis of graph mining experiments.

Inokuchi [29] proposed a method for mining generalized subgraphs based on a user-defined taxonomy of node labels. Thus, the search extends not only due to structural specialization, but also along the node label hierarchy. The method finds the most specific (closed) patterns at any level of taxonomy and support. Since the exact node and edge label representation is not explicitly given beforehand, the derivation of abstract patterns is semi-automatic.

Hofer, Borgelt and Berthold [22] present a pattern mining approach for ground features with class-specific minimum and maximum frequency constraints, that can be initialized with arbitrary motifs. All solution features are required to contain the seed. Moreover, their algorithm `MoSS` offers the facility to collapse ring structures into special nodes, to mark ring components with special node and edge labels, or to use wildcard atom types: Under certain conditions (such as if the atom is part of a ring), multiple atom types are allowed for a fixed position. It also mines cyclic structures at the cost of losing double-free enumeration.

All approaches have in common that the (chemical expert) user specifies high-level motifs of interest beforehand via a specific molecule representation. They integrate user-defined wildcard search into the search tree expansion process, whereas the

approach presented here derives abstract patterns automatically by resolving conflicts during backtracking and weighting. Importantly, it does so on the *ground* representation of the data, without lifting it first to a different representation.

6.1.2 Mining Latent Structure

The following presents a modular graph mining algorithm, called Latent Structure Pattern Mining (or LAST-PM for short), to identify higher level (latent) and mechanistically interpretable motifs for the first time in a fully automated fashion. The two-fold goal of LAST-PM is to find chemical substructures that are chemically informative and ultimately useful for prediction.

Technically, the approach is based on so-called *alignments* of features, i.e. orderings of nodes and edges with fixed positions in the structure. Such alignments may be obtained for features by controlling the pattern generating process in a graph mining algorithm with a canonical enumeration strategy. This is feasible, for instance, on top of current a-priori based graph mining algorithms. Subsequently, based on the canonical alignments, ground features can be stacked onto each other, yielding a weighted edge graph that represents the number of occurrences in the fragment set (see the left and middle panel of Fig. 6.2). In a final step, the weighted edge graph is reduced again (in this case by singular value decomposition) to reveal the latent structure of the pattern (see the right panel of Fig. 6.2). Thus, the approach can be described as a pipeline with the steps (a) align, (b) stack, and (c) compress.

6.1.3 Intuition

The following discusses an actual pattern found by LAST-PM to convey an idea what form those elaborate patterns take. The molecular fragments generated for the classification of the `bloodbarr` dataset (see section 4.3) were inspected for this purpose. A total number of 310 fragments was found by LAST-PM in the *msa* variant (see section 6.4) and 19 of these have ambiguities in the atomic number which represents a node of the structure. Taking into account its statistical significance [1], we chose the highest relevant fragment with ambiguities for a closer inspection; its expression in LAST-SMARTS reads

```
[#7,#8]-[#6]-[#6]=[#6]-[#7](-[#6])-[#6]-[#6].
```

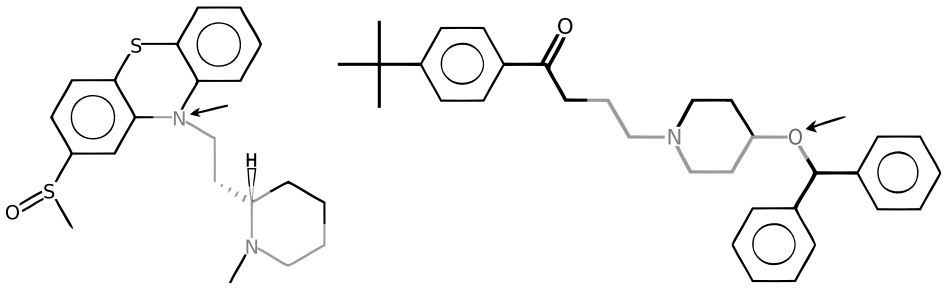


FIGURE 6.1: Two molecules with strong polarity, induced by similar fragments (gray).

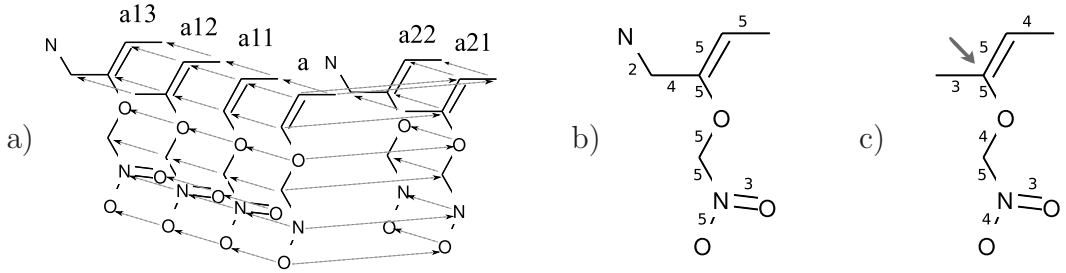


FIGURE 6.2: Illustration of the pipeline with the three steps (a) align, (b) stack, and (c) compress. Left: Aligned ground features in the partial order. Center: Corresponding weighted graph. Right: Latent structure graph.

A significant number of molecules containing this fragment are unable to cross the blood-brain barrier according to the binary classification of the dataset. Fig. 6.1 depicts two examples for molecules which contain the fragment in two different embeddings. It is well-suited to be analyzed by a chemical expert and to extract a general relationship between the molecular structure and its physiological activity. Chemically interesting is the interchangeability of the hetero atoms nitrogen (`[#7]`) and oxygen (`[#8]`, arrow-marked positions in Fig. 6.1). Although different in their chemical behavior, these atoms have in common a strong electronegativity. They become negatively charged when bound to aliphatic carbon atoms, as is the case in these fragments. In addition, the ambiguous position is separated by three carbon atoms from another negatively charged amine group. Therefore, the generated pattern represents molecules which possess the characteristic electrostatic structure of two negatively charged sites separated by a fixed distance. The occurrence of the pattern is a clear indication for a polar molecule, resulting in a decreased ability to cross membranes in the body, such as the blood-brain barrier. The pattern found in this case thus also confirms the correlation of electrostatic molecular properties with the drugs' ability to cross the blood-brain barrier [38].

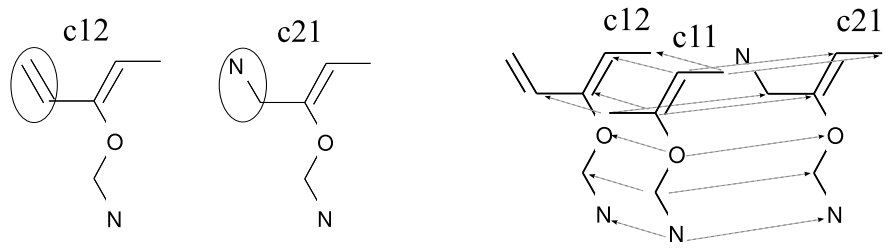


FIGURE 6.3: Left: Conflicting siblings $c12$ and $c21$. Right: Corresponding partial order.

6.1.4 Stacking Features

Two (distinct) features obtained by node refinements of a specific parent pattern are called *siblings*. Note that we have defined this term already differently in section 2.2.1 in the context of nodes of a graph, but the meaning will be clear from the context. Two aligned siblings r and s are called *mutually exclusive*, if they branch at different locations of the parent structure, i.e. let v_i and v_j be the nodes where the corresponding node refinements are attached in the parent structure, then $\phi_r(v_i) \neq \phi_s(v_j)$. Conversely, two siblings r and s are called *conflicting*, if they refine at the same location of the parent structure.

For several ground features, alignments can be visualized by overlaying or *stacking* the structures. It is possible to count the occurrences of every component (identified by its position), inducing a weighted graph. Assume a collection of aligned ground features with occurrences significantly skewed towards a single target class, as compared to the overall activity distribution. A "heavy" component in the associated weighted graph is then due to many ground features significant for a specific target class. Assuming correct alignments, the identity of different components is guaranteed, hence multiple adjacent components with equal weight can be considered equivalent in terms of their classification potential.

Fig. 6.2 illustrates the pipeline consisting of the three steps (a) align, (b) stack, and (c) compress, which exploits these relationships. It shows aligned ground features a, $a11$, $a12$, $a13$, $a21$, and $a22$ in the partial order (*search tree*) built by a depth-first algorithm. The aligned features can be stacked onto each other, yielding a weighted edge graph. Subsequently, latent information (such as the main components) can be extracted by SVD. Inspecting the partial order, note that refining a branches the search due to the sibling pair $a11$ and $a21$. Siblings always induce a branch in the partial order. Note that the algorithm will have to backtrack to the branching positions.

However, in general, the proposed approach is not directly applicable. In contrast to $a11$ and $a21$, which was a mutually exclusive pair, Fig. 6.3 shows a conflicting sibling pair, $c12$ and $c21$, together with their associated part of the partial order (matching elements are drawn on corresponding positions). It is not obvious how conflicting features could be stacked, thus conflict resolution is necessary.

The introduced concepts (alignment, conflicts, conflict resolution, and stacking) will now be used in the workflow and algorithm of LAST-PM.

6.2 Latent Structure Pattern Mining

This section explains the main steps of latent structure pattern mining:

1. Ground features are repeatedly stacked, resolving conflicts as they occur. A pattern representing several ground features is created.
2. The process in step 1. is bounded by a criterion to prevent the incorporation of too diverse features.
3. The components with the least information are removed from the structure obtained after step 2. Then the result (*latent structure*) is returned.

In the following, the basic components of the approach are described in some detail.

6.2.1 Efficient Conflict Detection

Conflict detection is based primarily on edges and secondarily on nodes. A node list is a vector of nodes, where new nodes are added to the back of the vector during the search. The edge list first enumerates all edges emanating from the first node, then from the second, and so forth. For each specific node, the order of edges is also maintained. Note, that for this implementation of alignment, the ground graph algorithm must fulfill certain conditions, such as partial order on the ground features as well as canonical enumeration (see section 6.3). In the following, the *core component* of two siblings denotes their maximum subgraph, i.e. the parent.

id	label
0	7
1	6
2	8
3	6
4	6
5	6
6	8
7	8

id1	id2	label
0	1	1
0	6	1
0	7	2
1	2	1
2	3	1
3	4	2
4	5	1

(a) $a11$ node and edge lists

id	label
0	7
1	6
2	8
3	6
4	6
5	6
6	8
7	6

id1	id2	label
0	1	1
0	6	1
1	2	1
2	3	1
3	4	2
3	7	1
4	5	1

(b) $a21$ node and edge lists

FIGURE 6.4: Node and edge lists for conflicting nodes $c12$ and $c21$, sorted by id (position). Underlined entries represent core nodes and adjacent edges.

Example 6.1. *Fig. 6.4 shows lists for features* $a11$ *and* $a21$*, representing the matching alignment. Underlined entries represent core nodes and adjacent edges. In line with previous observations, no distinct nodes and no distinct edges have been assigned the same position, so there is no conflict. The node refinement involving node identifier 7 has taken place at different positions. This would be different for the pattern pair* $c12$,$c21$*.*

Due to the monotonic addition of nodes and edges to the lists, conflicts between two ground features θ_1 and θ_2 become immediately evident through checking corresponding entries in the alignment for inequality. Three cases are observed:

1. Edge lists of θ_1 and θ_2 differ, but all elements with identical positions, i.e. pairs of ids, are equal. This does not indicate a conflict.

2. There exists an element in each of the lists with the same position that differs in the label. This indicates a conflict.

3. No difference is observed between the edge lists at all. This indicates a conflict, since the difference is in the node list (due to double-free enumeration, there must be a difference).

For siblings $a11$ and $a21$, case 1. applies, and for $c12$ and $c21$, case 2. applies. A conflict is equivalent to a missing maximal pattern for two aligned search structures (see section 6.2.2). Such conflicts arise through different embeddings of the conflicting features in the database instances. Small differences (e.g., a difference by just one node/edge), however, should be generalized.

6.2.2 Conflict Resolution

Let r and s be graphs. A *maximum refinement* m of r and s is defined as $(r \preceq m) \wedge (s \preceq m) \wedge (\forall n \succeq r : m \succeq n) \wedge (\forall o \succeq s : m \succeq o)$.

Lemma 6.1. *Let r and s be two aligned graphs. Then the following two configurations are equivalent:*

1. *There is no maximum refinement m of r and s with alignment ϕ_m induced by ϕ_r and ϕ_s, i.e. $\phi_m \supseteq \phi_r \cup \phi_s$.*

2. *A conflict occurs between r and s, i.e. either*

   (a) *$v_i \neq v_j$ for nodes $v_i \in r$ and $v_j \in s$ with $\phi_r(v_i) = \phi_s(v_j)$, or*

   (b) *$e_i \neq e_j$ for edges $e_i \in r$ and $e_j \in s$ with $\phi_r(e_i) = \phi_s(e_j)$.*

Proof. Two directions:
"1. $\Rightarrow$ 2.": Assume the contrary. Then the alignments are compatible, i.e. no unequal nodes $v_i \neq v_j$ or edges $e_i \neq e_j$ are assigned the same position. Thus, there is a common maximum pattern m with $\phi_m \supseteq \phi_r \cup \phi_s$.
"1. $\Leftarrow$ 2.": Since ϕ is a bijection, there can be at most one value assigned by ϕ for every node and edge. However, the set $\phi_m \supseteq \phi_r \cup \phi_s$ violates this condition due to the conflict. Thus, there is no m with $\phi_m \supseteq \phi_r \cup \phi_s$. □

In Fig. 6.3, the refinements of $c11$ have no maximum element, since they include conflicting ground features $c12$ and $c21$. In contrast, refinements of a in Fig. 6.2 do have a maximum element (namely pattern $a13$).

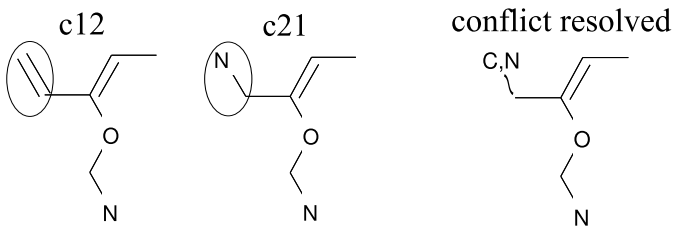


FIGURE 6.5: Conflict resolution by logical OR.

As a consequence of Lemma 6.1, conflicts prove to be barriers when several features should be merged to patterns, especially in case of patterns that stretch beyond the conflict position. A way to resolve conflicts and to incorporate two conflicting features in a latent pattern is by logical OR, i.e. any of the two labels may be present for a match.

Example 6.2. *Patterns* $c12$ *and* $c21$ *can be merged by allowing either single or double bond and either node label of* $\{N, C\}$ *at the conflicting edge and node, as shown in Fig. 6.5, represented by a curly edge and multiple node labels.*

Conflicts and mutually exclusive ground features arise from different embeddings of the features in the database, i.e. the locations (in terms of nodes and edges) where those patterns occur in the database instances may overlap, but are not in subset relation. Thus, the anti-monotonic property of diminishing support is lost between pairs of conflicting or mutually exclusive features. This also poses a problem for directly calculating the support of latent patterns.

6.2.3 Stopping Criterion

Since the alignment, and therefore equal and unequal parts, are induced by the partial order of the mining process, which is in turn a result of the embeddings of ground features in the database, they are employed to mark the boundaries within which merging should take place. Given a ground feature θ, its support in the positive class is defined as $y = |\{r \in \mathbf{r} \mid covers(\theta, r) \wedge a(r) = 1\}|$, its (global) support as x. Values of χ^2 tests are used to bound the merging process, since they incorporate a notion of *weight*: a pattern with low (global) support is down-weighted, whereas the occurrences of a pattern with high support are similar to the overall distribution. Assuming the notation of section 3.3.1 with $n = |\mathbf{r}|$ being the number of graphs, define the *weight* of a pattern as $w = \frac{x}{n}$. Moreover, assuming $m = \{r \in \mathbf{r} \,|a(r) = 1\}$, define the *expected support* in the positive [negative] class as

wm $[w(n-m)]$. Thus, equation 3.2 may be written as

$$\chi^2_d(x,y) = \frac{(y-wm)^2}{m} + \frac{(x-y-w(n-m))^2}{w(n-m)}, \tag{6.1}$$

which emphasizes the weight.

Definition 6.2 (Ground Features)**.** Given a graph database R, a user-defined minimum support f and user-defined minimum χ^2 value u, any $t \in B$ that is *frequent*, i.e. $supp(t,R) \geq f$, and *significant* with respect to occurrence in the target classes, i.e. $\chi^2_t \geq u$ is called a ground feature.

Given this definition, we can now collect associated ground features:

Definition 6.3 (Patch)**.** Given a graph database $R = \{r, a\}$, a patch P is a set of significant ground features, where for each ground feature θ there is a ground feature in P that is either sibling or parent of θ, and for each pair of ground features $(\theta_X, \theta'_Y) : X = Y, \quad X, Y \in \{\oplus, \ominus\}$.

The contour map for equally balanced target classes, a sample size of 20 and occurrence in half of the compounds in Fig. 3.3 illustrates the (well-known) convexity of the χ^2 function and a particular refinement path in the search tree with features partially ordered by χ^2 values as $1_\ominus > 2 < 3 < 4_\oplus < 5_\oplus$. Thus, the method indeed singles out "patches" of ground features in the search space significant for a particular target class. The members of these patches are fused to form a meta-pattern. Fig. 6.6 gives a schematic depiction of the hypothesis space.

6.2.4 Latent Structure Graph Calculation

In order to find the latent (hidden) structures, a "mixture model" for ground features can be used, i.e. elements (nodes and edges) are weighted by the sum of ground features that contain this element. It is obtained by stacking the aligned features of a specific patch, followed by a compression step. To extract the latent information, singular value decomposition (SVD) can be applied. It is recommended by Fukunaga to keep $80\% - 90\%$ of the information [13].

The first step is to count the occurrences of the edges in the ground features and put them in an adjacency table. For instance, Table 6.7(a) shows the pattern that results from the aligned features $a11$, $a12$, $a13$, $a21$, and $a22$ (see Fig. 6.2). As a specific example, edge $1-2$ was present in all five ground features, whereas edge $9-10$ occurred in two features only.

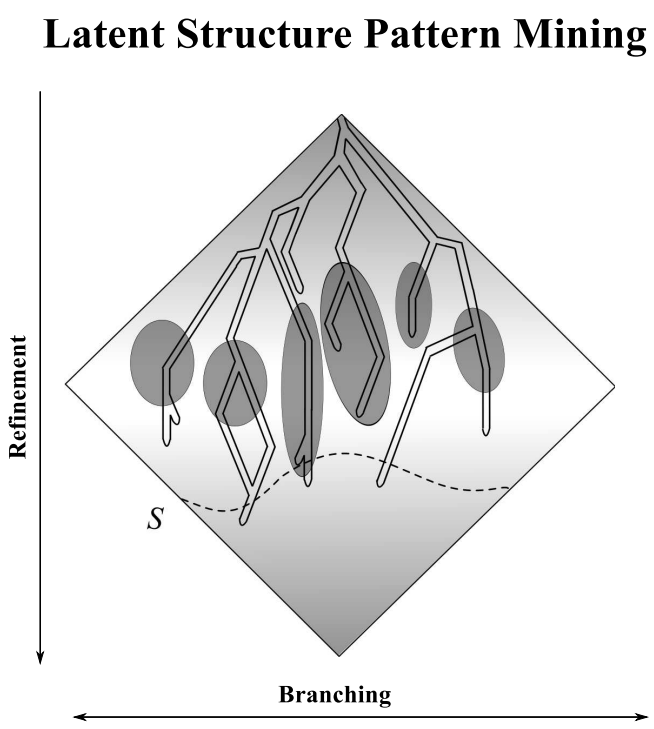


FIGURE 6.6: Hypothesis space of LAST-PM patterns

A SVD with 90%, applied to the corresponding matrix, yielded the latent structure graph matrix in Fig. 6.7(b). Here, spurious edges that were introduced by SVD (compression artifacts) were removed. As can be seen, the edges leading to the two nodes with degree 3 are fully retained, while the peripheral ones are downweighted. In fact, edge $9-10$ is even removed, since it was downweighted to weight 0. In general, SVD downweights weakly interconnected areas, corresponding to a blurred or downsampled picture of the original graph, which has previously proven useful in finding a basic motif in several ground patterns [70].

Definition 6.4 (Latent Structure Pattern Mining (LAST-PM))**.** Given a graph database R, and a user-defined minimum support *minsup*, calculate the latent structure graph of all patches in the search space, where for each ground feature θ, $supp(\theta) \geq minsup$.

The next section describes how the building blocks of this section can be embedded in a graph mining algorithm. The proposed technique requires a branch-and-bound algorithm and re-uses the partial order and canonical enumeration provided by such algorithms (see chapter 2, especially sections 2.1.2 and 2.2.4).

	1	2	3	4	5	6	7	8	9	10
1	0	5	0	0	0	0	0	0	0	0
2	5	0	5	0	0	0	0	3	0	0
3	0	5	0	5	0	0	0	0	0	0
4	0	0	5	0	5	0	0	0	0	0
5	0	0	0	5	0	5	0	0	4	0
6	0	0	0	0	5	0	5	0	0	0
7	0	0	0	0	0	5	0	0	0	0
8	0	3	0	0	0	0	0	0	0	0
9	0	0	0	0	4	0	0	0	0	2
10	0	0	0	0	0	0	0	0	2	0

(a) Weighted original adjacency matrix.

	1	2	3	4	5	6	7	8	9	10
1	0	4	0	0	0	0	0	0	0	0
2	4	0	5	0	0	0	0	3	0	0
3	0	5	0	4	0	0	0	0	0	0
4	0	0	4	0	5	0	0	0	0	0
5	0	0	0	5	0	5	0	0	3	0
6	0	0	0	0	5	0	4	0	0	0
7	0	0	0	0	0	4	0	0	0	0
8	0	3	0	0	0	0	0	0	0	0
9	0	0	0	0	3	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

(b) Latent structure adjacency matrix.

FIGURE 6.7: Input (above) and output (below) of latent structure graph calculation, obtained by aligning the features $a11 - a22$.

6.3 Algorithm

Given the preliminaries and description of the individual steps, this section presents a unified approach to latent structure pattern mining, combining alignment, conflict resolution, and component weighting. The method assumes (a) a partial order on ground features (vertical ordering), and (b) canonical representations for ground features, avoiding multiple enumerations of features (horizontal ordering). A depth-first pattern mining algorithm, possibly driven by anti-monotonic constraints, can be used to fulfill these requirements. It follows a strategy to extract

latent structures from patches. A latent structure is a graph more general than defined in section 2.2.1: the edges are attributed with weights, and the label function is replaced by a label relation, allowing for multiple labels. Since patches stretch horizontally (sibling relation), as well as vertically (parent relation), a recursive updating scheme is needed to embed the construction of the latent structure in the ground graph mining algorithm.

6.3.1 Merging

We first inspect the horizontal merging: given a specific level of refinement i, we start with an empty latent structure l^i and aggregate siblings from low to high in the lexicographic ordering, starting with empty l^i. For each sibling s and innate l^i, it holds that either

1. s is not significant for any target class, or
2. s is significant for the same target class as l^i, i.e. $X = Y$, for s_X, l^i_Y (if empty, s initializes l^i to its class), or
3. s is significant for the other target class.

In cases 1. and 3., l^i is subjected to latent structure graph calculation and output, and a new, empty latent l^i is created. For case 3., it is additionally initialized with s. For case 2., however, s and l^i are merged, i.e. subjected to conflict resolution, aligning s and l^i, and stacking s onto l^i.

For the vertical or top-down merging, we return l^i to the calling refinement level $i-1$, when all siblings have been processed as described above. Structures l^i and l^{i-1} are merged, if l^i is significant for the same target class as l^{i-1}, i.e. $X = Y$, for l^i_X, l^{i-1}_Y. Also, condition 1. must not be fulfilled for the current sibling on level $i-1$. Otherwise, both l^i and l^{i-1} are subjected to latent structure graph calculation and output, and a new l^{i-1} is created.

Alignment calculation (Algorithm 3) works recursively: In lines 3-9, it extracts mutually exclusive edges leaving core positions to non-core positions, i.e. there is a distinction between edges leaving the core, but are shared by $l1$ and $l2$ (conflicting edges, E), vs. edges that are unique to either l_1 or l_2 (non-conflicting edges, E_{l_1}, E_{l_2}). The overall minimum edge is remembered for the next iteration, ordered by "to"-node position (lines 11-12). The minimum edge of E_{l_1} and E_{l_2} (line 10; in

Algorithm 3: Alignment Calculation

Input : Latent structures l_1, l_2; an interval C of core node positions.
Output: Aligned and stacked version of l_1 and l_2, conflicts resolved.

```
1  repeat
2      E.clear() ; E_{l_1}.clear() ; E_{l_2}.clear() ;
3      for j = 0 to (size(C)-1) do
4          index = C[j] ;
5          I = (l_1.to[index] ∩ l_2.to[index]) ;
6          E.insert(I ∖ C) ;
7          E_{l_1}.insert(l_2.to[index] ∖ I) ;
8          E_{l_2}.insert(l_1.to[index] ∖ I) ;
9      end
10     if min(E_{l_1}) ≤ min(E_{l_2}) then M_1 = E_{l_1} else M_1 = E_{l_2} ;
11     if min(E) < min(M_1) then M_2 = E else M_2 = M_1 ;
12     core_new.insert(min(M_2)) ;
13     if M_1 == E_{l_1} then l_2.add_edge(min(M_1)) else l_1.add_edge(min(M_1)) ;
14 until E.size==0 ∧ E_{l_1}.size==0 ∧ E_{l_2}.size==0 ;
15 l_1 = stack(l_1, l_2) ;
16 l_1 = alignment(l_1, l_2, core_new) ;
17 return l_1;
```

case of equality, E_{l_1} takes precedence) is added to the other structure where it was missing (line 13).

The procedure can be seen as inserting pseudo-edges into the two candidate structures that were only present in the other one before, thus creating a canonical alignment. For instance, in Fig. 6.4, exclusive edge 0-7 from $a11$ would be first inserted into $a21$, *pushing* node 7 to node 8 and edge 3-7 to edge 3-8 in $a21$. Subsequently, vice versa, exclusive edge 3-8 would be inserted into $a11$, leaving no more exclusive edges, i.e. the two structures are aligned.

This process is repeated until no more edges are found, resulting in the alignment of l_1 and l_2. Line 15 then calls the stacking routine, a set-insertion of l_2's node and edge labels into l_1's and the addition of l_2's edge weights to l_1's, and line 16 repeats the process for the next block of core ids. Due to the definition of node and edge lists, the following invariant holds in each iteration: For the node list, core components are always enumerated in a contiguous block, and for each edge e, the core components are always enumerated at the beginning of the partition of the edge list that corresponds to e. For horizontal (vertical) merging, we call Algorithm 3 with $l_1 := l^i$, $l_2 := s$ ($l_1 := l^{i-1}$, $l_2 := l^i$). This ensures that l_1 comprises only ground features lower in the canonical ordering than l_2. Thus, Algorithm 3 correctly calculates the alignments.

6.3.2 Complexity

The `Gaston` algorithm by Nijssen and Kok [44] was modified to support latent structure pattern mining[1]. It is especially well-suited for the purpose of latent structure pattern mining: First, `Gaston` uses a highly efficient canonical representation for graphs. Specifically, no refinement is enumerated twice. Second, `Gaston` employs a canonical depth sequence formulation that induces a partial order among trees (cycle-closing structures are not considered due the complexity of the isomorphism problem for general graphs). Siblings in the partial order can be compared lexicographically.

Latent Structure Pattern Mining (LAST-PM) allows the use of anti-monotonic constraints for pruning the search in the forward direction, such as minimum frequency or upper bounds for convex functions, e.g χ^2. The former is integrated in `Gaston`, the latter is implemented via statistical metric pruning, more specifically as static upper bound pruning using a χ^2 upper bound as described in section 3.3.2. Obviously, the additional complexity incurred by LAST-PM depends on conflict resolution, alignments, and stacking (see Algorithm 3), as well as weighting (SVD).

- Algorithm 3 for latent structures $l1$, $l2$ takes at most $|l1| + |l2|$ insert operations, i.e. is linear in the number of edges (including conflict resolution).
- For each patch, an SVD of the $m \times n$ latent structure graph is required ($mn^2 - n^3/3$ multiplications).

Thus, the overhead compared to the underlying `Gaston` algorithm is rather small (for an empirical analysis, see section 6.5.2).

6.4 LAST-SMARTS

Given the output of LAST-PM, SMARTS patterns for instantiation are created by parsing patterns in pre-order (depth-first). Those patterns are referred to as *LAST-SMARTS* (see Appendix A). This section presents different possible parsing mechanisms and evaluates their predictive power, employing the nearest-neighbor approach described in section 3.3.2.

[1] Version 1.1 (with embedding lists), see `http://www.liacs.nl/~snijssen/gaston/`.

6.4.1 Weighted Depth-First Parsing

Focusing on a node, all outgoing edges have weights according to section 6.2.4. This forms weight levels of branches with the same weight. We may choose to make some branches optional, based on the size of weight levels, or demand all branches to be attached:

- *nop*: demand all (**n**o **op**tional) branches.
- *msa*: demand number of branches equal to **m**aximum **s**ize of **a**ll levels
- *nls*: demand number of branches equal to highest (**n**ext) **l**evel **s**ize

Example 6.3. *Variant nop would disregard weights and require all of the three bonds leaving the arrow-marked atom of Fig. 6.2 (right), while nls (here also msa) would require any two of the three branches to be attached.*

With *msa* and *nls*, combinations of important branches may be captured better. The two methods allow, besides simple disjunctions of atomic node and edge labels such as in Fig. 6.1, for (nested) optional parts of the structure (see Appendix A).

6.4.2 Experiments

This section presents experimental results on three small to medium-sized `OFS` datasets with binary class labels from the study by Rückert and Kramer [49], see section 4.3. The same framework as in section 5.7.1 for backbone refinement classes was used. This time again, only the *WT* measure was employed to rank the different variants. Fig. 6.8 compares the performance of LAST-PM variants. For `nctrer` and `bloodbarr`, *nls* and *msa* performed best (with slight advantages for *nls*). On the `yoshida` data set, however, *nop* was the winner.

6.4.3 Conclusions

The variants *nls* and *nsa* perform a more detailed analysis of the structure than *nop* by taking the weight levels into account. Moreover, they are able to declare larger parts of the structure as optional, which *nop* cannot.

I want to stress here that the non-marginal differences found between the three variants can be also be seen as a validation of the principle of weighted graph

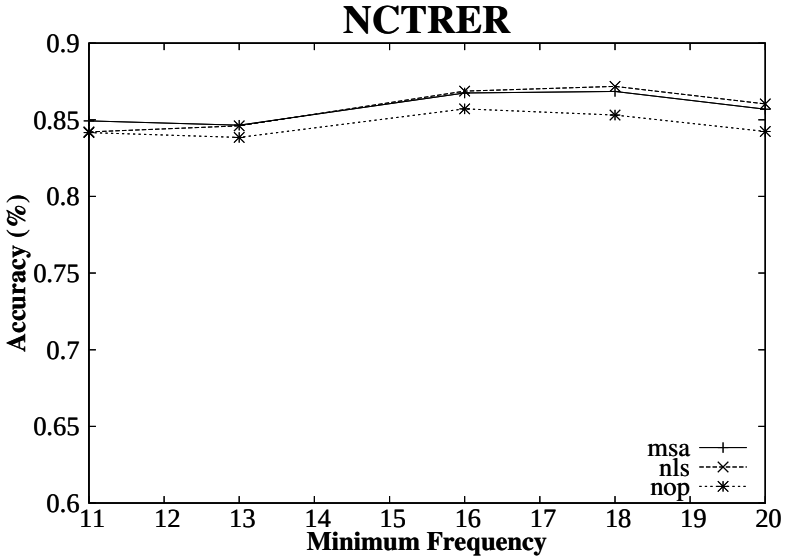


FIGURE 6.8: LAST-PM: Validation against different variants.

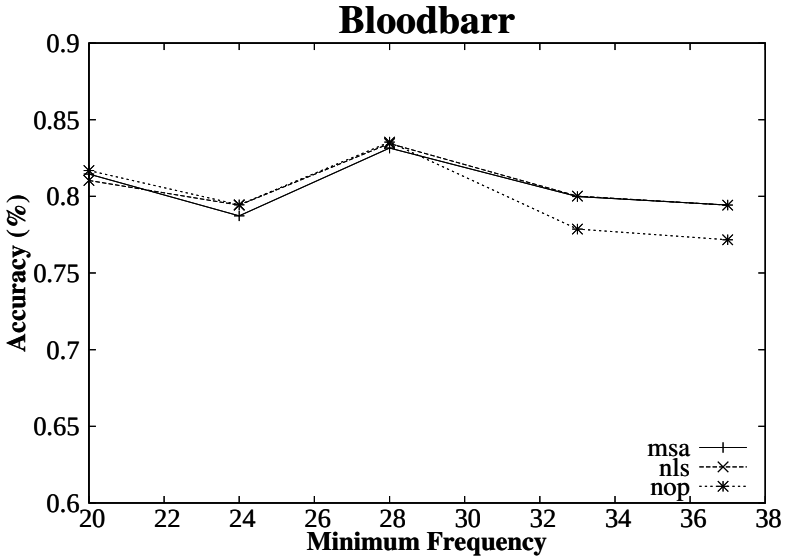


FIGURE 6.8: LAST-PM: Validation against different variants.

components in general. Although on two of the three data sets the intuition of higher expressiveness could be corroborated, in the rest of the experiments the best method available was used for each dataset.

6.5 Classification Accuracy and Runtime

This section presents experimental results on three small to medium-sized OFS datasets with binary class labels from the study by Rückert and Kramer [49],

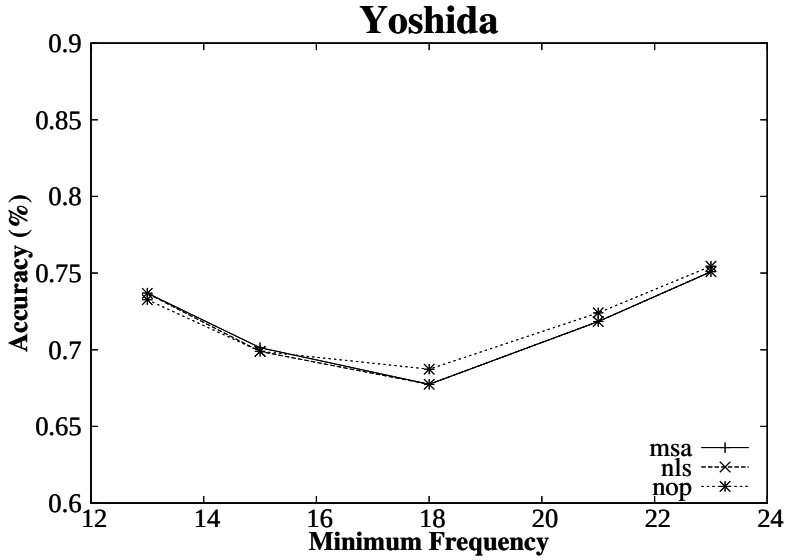


FIGURE 6.8: LAST-PM: Validation against different variants.

see section 4.3. Those datasets deal with complex biological endpoints, namely estrogen receptor activity, blood-brain-barrier activity, and bioavailability. The mode of action of a compound is usually explained directly with physico-chemical properties, such as electrotopology, quantum chemical properties, or 3-D structure, or hydrophobic properties. Accordingly, those endpoints were modeled with such descriptors by the original authors [12, 38, 68], while purely structural 2-D descriptors, such as those created by LAST-PM, are not used by most studies in this area.

6.5.1 Classification Accuracy

Edges were attributed as single, double and triple, or as aromatic bond, as inferred from the molecular structure. Features were converted to SMARTS according to the variants *msa*, *nls*, and *nop*, and matched onto training and test instances, yielding instantiation tables. A 20% SVD compression (percentage of sum of singular value squares) is reported for the LAST-PM features, since this gave the best AUROC values of 10, 15, and 20% in preliminary trials in two out of three times. Significance was determined by the 95% confidence interval.

6.5.1.1 Validation Against Compressed and Elaborate Representations

The results were obtained from repeated (two times in total) ten-fold stratified crossvalidation (two times with different folds). Edge-induced subgraphs were

Dataset		LAST-PM		ALL	BBRC	MOSS	SLS
	Variant	%Train	%Test	%Test	%Test	%Test	%Test
`bloodbarr`	nls+nls	84.19	72.20	70.49[a]	68.50[a]	67.49[a]	70.4[b]
`nctrer`	nls+msa	88.01	80.22	79.13	80.22	77.17[a]	78.4[b]
`yoshida`	nop+msa	82.43	69.81	65.19[a]	65.96[a]	66.46[a]	63.8[b]

[a] significant difference to LAST-PM.

[b] result from the literature, no significance testing possible

TABLE 6.1: Comparative analysis (repeated 10-fold crossvalidation).

used as ground features. For each training set in a crossvalidation, descriptors were calculated using 6% minimum frequency and 95% χ^2-significance on ground features to ensure features are selected ignorant of test sets. Unoptimized linear SVM models with a constant parameter $C = 1$ for each pair of training and test set were employed. The statistics in the tables were derived from pooling the twenty test set results into a global table first.

The performance of LAST-PM descriptors is compared in Table 6.1 to that of

1. ALL ground features from which LAST-PM descriptors were obtained (baseline comparison).
2. BBRC descriptors to relate to structurally diverse and class-correlated ground features.
3. MOSS descriptors by Borgelt and Berthold [22] to see the performance of another type of abstract patterns.
4. SLS descriptors by Rückert and Kramer [49] to see the performance of ground features compressed according to the so-called dispersion score.

For ALL and BBRC, a minimum frequency of 6% and a significance level of 95% were used. For the MOSS approach, features were obtained with MoSS [22]. This involves cyclic fragments and special labels for aromatic nodes. In order to generalize from ground patterns, ring bonds were distinguished from other bonds. Otherwise (including minimum frequency) default settings were used, yielding only the most specific patterns with the same support (closed features). For SLS, the overall best figures are reported for the dispersion score and the SVM model from Table 1 in their paper. As can be seen from Table 6.1, using the given variants for the first and second fold, respectively, LAST-PM outperforms ALL, BBRC

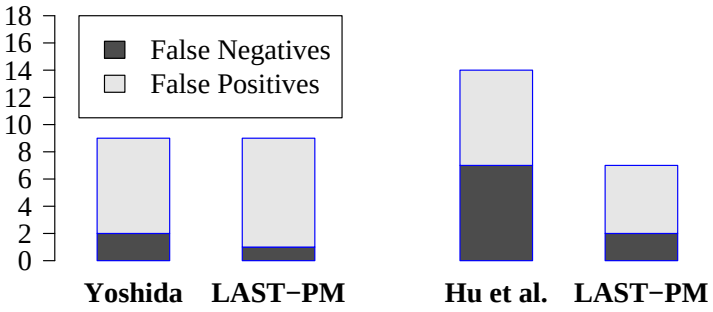


FIGURE 6.9: LAST-PM: Performance on External Test Sets.

and MOSS significantly for the `bloodbarr` and `yoshida` dataset (paired corrected t-test, $n = 20$), as well as MOSS for the `nctrer` dataset (seven out of nine times in total).

6.5.1.2 Validation Against Original QSAR Models

In their original paper [68], Yoshida and Topliss report on the prediction on an external test set of 40 compounds with physico-chemical descriptors, in which they achieved a false negative count of 2 and false positive count of 7. The authors provided the test set and their exact accuracy with 1 false negative and 8 false positives could be reproduced, using LAST-PM features.

Hu and co-workers [38], authors of the `bloodbarr` dataset study, provided us with the composition of their "external" validation set, which is in fact a subset of the complete dataset, comprising 64 positive and 32 negative compounds. Their SVM model was based on carefully selected physico-chemical descriptors, and yielded only seven false positives and seven false negatives, an overall accuracy of 85.4%. Using LAST-PM features and the unoptimized linear polynomial kernel, only five false positives and two false negatives were predicted, an overall accuracy of 91.7%. Fig. 6.9 summarizes the results.

Further experiments with another 110 molecule blood-brain barrier dataset (46 active and 64 inactive compounds) by Hou and Xu [26] were conducted, obtained together with pre-computed physico-chemical descriptors. Here, an AUROC value of 0.78 was reached using LAST-PM features in repeated 10-fold crossvalidation, close to the 0.80 that the authors obtained with the former. However, when combined, both descriptor types give an AUROC of 0.82. In contrast to this,

Dataset	LAST-PM	ALL	PCR/RTR
`bloodbarr`	249 (1.23s)	1613 (0.36s)	0.15 /3.41
`nctrer`	193 (12.49s)	22942 (0.13s)	0.0084 /96.0769
`yoshida`	124 (0.28s)	462 (0.09s)	0.27 /3.11

TABLE 6.2: Analysis of pattern count and runtime.

AUROC could not be improved in combination with BBRC instead of LAST-PM descriptors.

6.5.2 Runtime Analysis

According to section 6.3.2, the additional work compared to ground feature mining is induced by conflict resolution, alignments, and stacking in LAST-PM. It is clear that this effort will be non-marginal, given the time complexity of SVD. However, it bears the potential to drastically reduce the resulting pattern result.

An analysis investigating the tradeoff between runtime and pattern set reduction was performed [17]. The same settings as in section 6.5.1 were applied.

Table 6.2 relates pattern count and runtime (in braces) of LAST-PM and ALL (median of 20 folds). PCR is the pattern count ratio, defined as the ratio of LAST-PM patterns to ALL patterns, while RTR the runtime ratio between LAST-PM and ALL, as measured for descriptor calculation on the 2.4 GHz Intel Xeon test system with 16GB of RAM, running Linux 2.6. It turned out that $1/PCR$ always exceeded RTR. Especially, for the `nctrer` dataset, the result set could be reduced from $> 20,000$ ground features to merely 193 patterns. Profiling showed that most CPU time was spent on alignment calculation, while SVD can be neglected.

6.5.3 Conclusions

LAST-PM descriptors are well-suited for binary classification tasks involving complex biological endpoints. They were found to be not only competitive to other compressed representations – more importantly, they are able to improve over the complete set of ground features from which they were derived. This corroborates the intuition that the pipelined process of LAST-PM may extract latent (hidden) information from the ensemble of ground features in a patch that cannot be found in any individual ground feature.

A further, very important conclusion is that 2-D fragments can be used competitively to physico-chemical descriptors, as was shown on the external test sets from the original studies. The extraction of latent information from the ensemble of ground features seems to have closed the gap in expressiveness to the latter that has been arguably present before. The information seems to be complementary and beneficial in addition to physico-chemical properties.

Given the favorable tradeoff between runtime and pattern set reduction, it may well be concluded that the additional computational effort is justified. Combined with the classification accuracy results from section 6.5.1, this result implies that the crucial bits of information needed to classify molecular data can be embedded in just a very small set of chemical fragments.

6.6 Conclusions

Latent Structure Pattern Mining (LAST-PM), a method for generating abstract non-ground descriptors for large databases of molecular graphs was introduced. The approach differs from traditional graph mining approaches in that it incorporates several similar descriptors into a larger pattern reveals additional (*latent*) information, e.g., on the most frequently or infrequently incorporated parts, emphasizing a common interesting motif. It can thus be seen as graph mining on subgraphs.

In traditional frequent or correlated pattern mining, sets of ground features are returned, including groups of very similar ones with only minor variations of the same interesting basic motif. It is, however, hard and error-prone (or sometimes even impossible) to appropriately select a representative from each group, such that it conveys the basic motif. Latent structure pattern mining can also be regarded as a form of abstraction, which has been shown to be useful for noise handling in many areas. It is, however, new to graph and substructure mining.

The key experimental results were obtained on blood-brain barrier, estrogen receptor binding and bioavailability data, which have been hard for substructure-based approaches so far. The experiments showed that the non-ground pattern sets improve over the set of all ground features from which they were derived, but also over MOSS descriptors [22], BBRC descriptors, and compressed [49] ground feature sets when used with SVM models. In seven out of nine cases, the improvements are statistically significant. Also, a favorable tradeoff between pattern count

and runtime for computing LAST-PM descriptors compared to the complete set of frequent and correlated ground features was found.

Bioavailability and blood-brain barrier data were taken together with corresponding QSAR models from the literature and it was shown that, on three test sets obtained from the original authors, the purely substructure-based approach is on par with or even better than their approach based on physico-chemical properties only. It was also shown that LAST-PM descriptors can enhance the performance of solely physico-chemical properties. Therefore, latent structure patterns show some promise to make hard (Q)SAR problems amenable to graph mining approaches.

Chapter 7

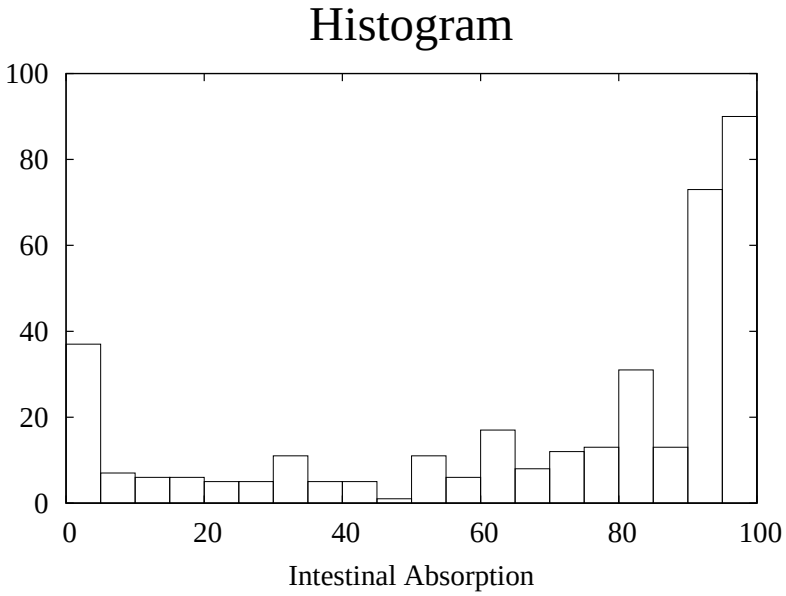
Case Study: A Biological Dataset

This chapter shows in a paradigmatic way, how backbone refinement class mining and latent structure pattern mining may be applied to a chemical dataset with a biological endpoint. It will be investigated how well BBRC and LAST-PM predict the dataset, with a focus on chemical similarity.

7.1 Dataset

In a recent study, intestinal drug absorption in humans was the subject of QSAR modeling [59]. This topic is of great interest, since the ability of chemicals to pass from the gastrointestinal tract into the systemic circulation (blood) is central to drug discovery. Any chemical failing to do so sufficiently will not be able to reach the target site (organ), and thus will fail to exhibit its desired mechanism of action. Key factors that limit the transport are low solubility, chemical instability, high hydrogen bonding ability, as well as high gastrointestinal metabolism.

The dataset used was put together from the literature and consists of 458 small, druglike compounds with FDA approval, for which experimental data were available and sufficiently documented. The variable describing the endpoint (*intestinal absorption*) is defined as the percentage of the dose absorbed from the gastrointestinal tract following oral administration. Fig. 7.1 shows that the endpoint variable is quite skewed in the dataset. In the study, three classes were defined according to which the variable was binned: *false* for $\geq 80\%$, *unknown* between 30% and 80%, and *true* for $\leq 30\%$. Only the class labels *true* and *false* were considered, molecules with label *unknown* were disregarded, yielding a *true*/*false* ratio of 73/303 (≈ 0.241).



Class ratio *true* (≤30) *vs. false* (≥80): $73/303 \approx 0.241$

FIGURE 7.1: Histogram for the Intestinal Absorption Dataset

MF	BBRC	LAST-PM
25	101	74
20	140	132
15	267	336
10	498	794
8	605	1,043
7	710	1,260
6	792	1,413
5	1,586	2,430
4	3,423	3,656

TABLE 7.1: Minimum Frequency Table

7.2 Parameter Selection

In order to find acceptable parameter values for minimum frequency and minimum correlation (BBRC only), BBRC and LAST-PM were run several times against the whole dataset in ascending minimum frequencies (see Table 7.1). The runtimes never exceeded one minute: For the lowest minimum frequency of 4, BBRC finished in 6.8s, and LAST-PM in 34.3s.

Further preliminary testing, now using crossvalidation runs, revealed that 95% would be acceptable as significance threshold (the default for both BBRC and LAST-PM). For a good tradeoff between fingerprint coverage and runtime, minimum frequencies of 20 for BBRC and 25 for LAST-PM were finally chosen.

7.3 Algorithms Validation

BBRC and LAST-PM were validated with the `Lazar` framework [21]. `Lazar` uses local models built at prediction time in order to improve on predictive accuracy (*lazy learning*). Conceptually, it employs a weighted Tanimoto index to determine neighbors to the query structure and derives a prediction from them. The Tanimoto index is built using fingerprints which indicate the presence or absence of substructures in molecules, or the number of times substructures occur in molecules.

The left hand side of Table 7.2 defines basic statistics (TP = Number of True Positives, FP = Number of False Positives, analogously for the negative class), as well as some associated ratios. Importantly, `Lazar` provides a confidence value with every prediction, ranging between 0 and 1, based on the mean neighbor Tanimoto similarity. A measure that integrates ACC (*accuracy*) and neighbor similarity into a single numeric value can be derived by weighting each prediction with its associated confidence, referred to as WACC (*weighted accuracy*, see section 5.7.1). The hypothesis is that WACC would be higher than ACC since better predictions should occur for high confidences.

Three different factors were assessed for influence on model quality. Besides using BBRC *vs.* LAST-PM descriptors, quantifying each pattern in the fingerprints with the number of times ("hits") it occurs in a molecule *vs.* simple binary indication of occurrence as well as the effect of learning an SVM model *vs.* weighted majority vote on the neighbors, were investigated. The right hand side of Table 7.2 encodes these factors with capital letters.

7.4 Results

Since three binary factors were considered, eight validation settings were mutually compared in total: For any of the eight settings, five times ten-fold crossvalidation was conducted, yielding 50 data points each. Each pair of such vectors was subsequently subjected to a paired t-test with significance level set to $\alpha = 0.05$.

Prediction →	*true*	*false*	Σ
true	TP	FN	P
false	FP	TN	N

TPR = TP/P
TNR = TN/N
PPV = TP/(TP+FP)
NPV = TN/(TN+FN)
ACC = (TP+TN)/(P+N)

Code	Description
L	LAST-PM
B	BBRC
S	SVM
M	Majority Vote
H	Number of Hits
O	Occurrence

TABLE 7.2: Algorithm Comparison Code Table

	BSO	BMO	BSH	BMH	LSO	LMO	LSH	LMH
BSO		w		w		w	wa	wa
BMO	W				W		a	a
BSH								w
BMH	W				W		a	a
LSO		w		w		w		w
LMO	W				W			
LSH	WA	A		A				
LMH	WA	A	W	A	W			

TABLE 7.3: Algorithm Comparison: Significant Differences

Table 7.3 gives the relative performance results for the two accuracy measures. A capital "W" in cell i, j indicates that the setting in row i scored significantly higher than the one in column j for the WACC measure, a lowercase "w" signals significantly lower score. The same scheme is used for ACC with letters "A", "a".

From the results, three major trends may be observed: Best performance was seen using LAST-PM descriptors with number of hits per compound. Weighted majority voting shows good performance, regardless of descriptor type and occurrence indicator (*M* combinations). The *SO combinations (support vector machine and binary occurrence) generally perform worst.

Fig. 7.2 shows a chart of ACC and WACC, along with TPR (*true positive rate* and TNR (*true negative rate*), as well as PPV (*positive predictive value*) and NPR (*negative predictive value*). TPR and TNR are referred to as *sensitivity* statistics, PPV and NPV as *selectivity* statistics.

While negatives are detected nearly completely for the combinations using LAST-PM, the sensitivity towards the actives is actually significantly better using BBRC descriptors, with BMH performing roughly four times as good as random guessing

	ACC	WACC	TPR	TNR	PPV	NPV
BSO	0.824	0.821	0.422	0.925	0.621	0.864
BMO	0.824	0.868	0.454	0.916	0.604	0.870
BSH	0.839	0.847	0.485	0.931	0.619	0.877
BMH	0.834	0.875	0.491	0.923	0.633	0.877
LSO	0.839	0.840	0.351	0.955	0.671	0.863
LMO	0.846	0.869	0.350	0.963	0.699	0.864
LSH	0.858	0.862	0.405	0.961	0.678	0.876
LMH	0.859	0.881	0.406	0.964	0.700	0.875

TABLE 7.4: Algorithm Comparison: Statistics

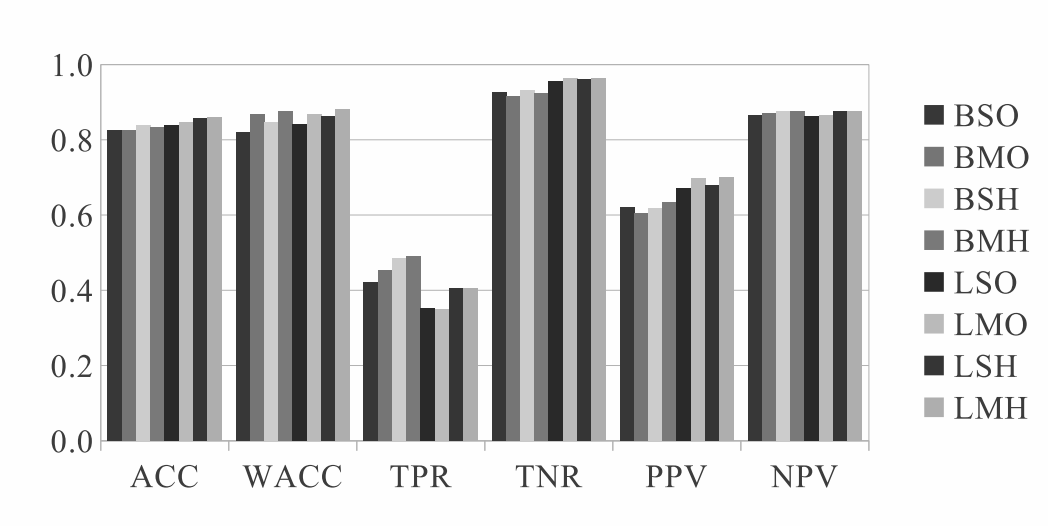


FIGURE 7.2: Algorithm Comparison: Statistics Chart

on the positive class. However, LAST-PM is able to predict the positive class much more specifically than BBRC, as the higher PPV values for the LM* combinations indicate.

To illustrate the practical use of confidences and the importance of the WACC measure, Fig. 7.3 plots confidence *vs.* ACC in a cumulative fashion (taken from one of the crossvalidation runs of the LMH setting): The leftmost data point represents a single prediction (the one with the highest confidence), the second leftmost point represents two predictions (those with the two highest confidences) and so forth. Confidences *per se* are not probabilities, but raw, uncalibrated measures. Using an analysis such as in Fig. 7.3 however, they could be converted into probability estimates. This would appear straightforward, since any point in the

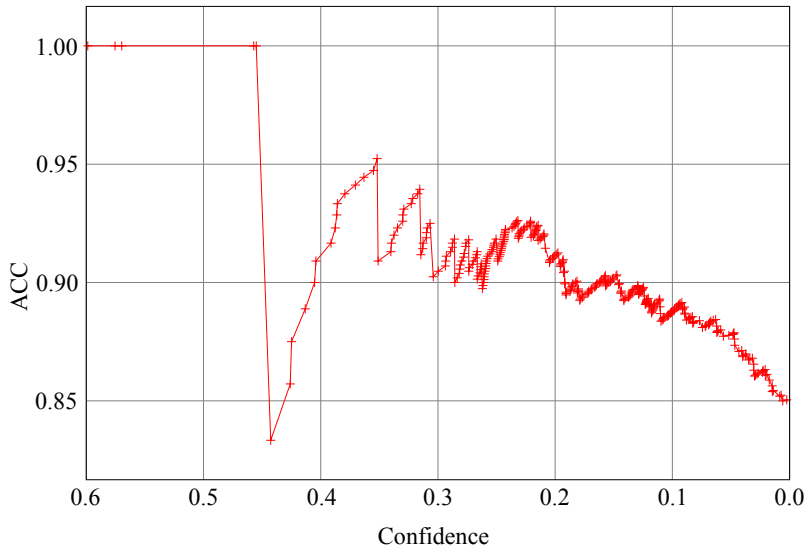


FIGURE 7.3: ACC *vs.* Confidence Chart

plot is the empirical probability of obtaining a correct prediction for a confidence at least that high. Doing so would enable users to specify comprehensible thresholds for minimum confidence values in terms of probabilities. A present drawback, however, is the jitter in the high confidence regions, where, due to the cumulative approach, only few data points lie.

7.5 Discussion

The modelling exercise in this chapter has found LAST-PM to perform overall better than BBRC on this dataset, probably due to its ability to mine more elaborate patterns. This backs up the experience made already in the experiments in the LAST-PM section, namely that LAST-PM performs often better on complex biological endpoints (*cf.* section 6.5.1). The results of the significant difference testing (*cf.* Fig. 7.2) suggest also that weighted majority vote should be used. They suggest furthermore that the number of hits for quantifying patterns in the fingerprints should be used instead of mere presence indication. Remarkably, these findings are consistent through all 26 factor combinations, in the sense that no single factor switch could be found contradicting these rules (there are exceptions for two factors, however, e.g. BMO beats LSO on WACC). Direct comparisons between the influence of descriptor type and prediction scheme indicate a slight advantage for the latter, thus it would be more important to use majority voting instead of SVM models than to use LAST-PM instead of BBRC.

The experiments revealed high sensitivity values on the negative class (TNR), while sensitivity for the positive class remained below 0.5. This may be attributed to the skewed distribution of the target classes. However, good specificity (NPV and PPV) was found for the overall winning LM* combinations. Therefore, despite many (in relation to the small positive class) false negatives, the positive predictions made were quite specific.

It has been shown in two ways that confidence is a meaningful way of describing prediction quality: First, WACC was found to be higher than ACC for nearly all compared models (see Table 7.4). Second, models employing majority vote generally showed superior performance compared to support vector machine models. In the former, the contribution of each neighbor was weighted according to its similarity to the query structure (remember that confidence was defined as mean neighbor similarity), whereas in the latter, it was not. BBRC seemed to profit more from confidence than LAST-PM, as the difference between ACC and WACC was consistently higher there. However, although BBRC descriptors were able to achieve a significantly higher number of true positives, they induced models with a significantly lower specificity and also lower total accuracy.

7.6 Resources Used

BBRC and LAST-PM are available as web services in the OpenTox framework (`http://www.opentox.org`, *cf.* Appendix C). OpenTox is a research project financed by the European Union with the aim to establish an open standards predictive toxicology framework, connecting chemoinformatics services via the web [17].

This chapter was produced using the OpenTox facilities for algorithm validation, specifically the `algorithm_comparison` module. The validation engine autonomously splits datasets in training and test folds and sends them over the network to the algorithm service. Only after the model has been built, validation sends the test dataset to the model to obtain predictions [1]. This approach effectively prohibits information flow from the test set at training time.

[1] In case of a lazy learning algorithm, such as Lazar, the training phase consists mainly in pattern generation.

Chapter 8

Conclusions and Future Work

8.1 Summary of Results

Chapter 1 prefaces the thesis with the basic requirements for predictive, interpretable, and time-efficient QSAR models. It shows that data mining methods made the process modular, but also introduced new complexity that has to be dealt with. Chapters 2 and 3 introduce graph mining and strategies that enable graph mining algorithms to search a database of graphs efficiently, with respect to general complexity results. The process of model building using data mining methods is elaborated with a focus on compression methods, among them selection according to class correlation.

In the next two chapters, two graph mining algorithms that address the above aspects are proposed – the following list compactly summarizes the results:

- In an artificial – but reasonably chosen – search space, backbone refinement class mining achieves high compression due to its unique partitioning of the search space (see section 5.3).
- Backbone refinement class descriptors are able to maintain high coverage for increasing minimum frequencies. The normality of the distribution is higher without aromatic perception, whereas compression (and thus representativeness) is lower (see section 5.4).
- The structural partitioning of the search and the statistical selection criterion in backbone refinement class mining produce highly diverse and well-distributed patterns (see section 5.5).

- The method scales nicely, it is even applicable for large-scale data, as exemplified on the (to the author's best knowledge) largest database in correlated graph mining yet (see section 5.6).
- Backbone refinement class representatives show very good classification performance with very few descriptors, which enables the use of large-scale datasets also in QSAR models (see section 5.7).

- There is a striking lack of graph mining algorithms for the calculation of latent (hidden) motifs in graph databases. A pipelined approach to latent structure pattern mining is proposed. An example shows, how such a pattern may be interpreted by a chemical expert (see section 6.1).
- The building blocks for latent structure pattern mining are defined and theoretically examined. This includes conflict detection and conflict resolution (introducing ambiguities), and bounding the search by structural and statistical constraints (see section 6.2).
- In latent structure pattern mining, the computational effort for aligning basic fragments is kept linear when embedded into a branch-and-bound graph mining algorithm (see section 6.3).
- Latent structure pattern mining is shown to outperform other compressed representations. Unoptimized SVM models using LAST-PM descriptors are on par or better to highly optimized QSAR models for complex biological endpoints. The tradeoff between compression and runtime indicates good scalability (see section 6.5).

In view of these results, I consider the hypothesis from section 1.5 confirmed.

8.2 Lessons Learned

The interplay of machine learning techniques in the process of creating a QSAR model is an enormously complex process that always needs optimization. In particular, the choices taken in any stepwise approach to model building (*cf.* chapter 3) must be all well-founded for the predictive model to generalize well to unseen data. This work has contributed to the state of the art in fragment-based descriptor generation by methods that produce compact, in practice efficiently computable, and

interpretable pattern sets with good expressiveness. They do so by employing a novel combination of user-defined and structural constraints.

My experience confirms the idea of *sparse selections*, i.e. that exhaustive searches on the complete descriptors space should be rather avoided in graph mining applied to chemical databases. Designing the BBRC algorithm was guided by the idea that descriptors can be calculated efficiently due to their distribution around the medium-frequent patterns. Thus, not all frequency levels above minimum frequency would have to be searched. Indeed, in the experiments on the largest labeled set of chemical compounds used so far in class-correlated graph mining, it was shown that BBRC descriptors can be computed within reasonable time and used in simple predictive learning schemes.

LAST-PM can be seen as "nested" graph mining, since it processes the results of a graph mining algorithm to find the most frequently or infrequently incorporated parts of fragments, emphasizing a common interesting motif (latent structure). Thus, it may aggregate and summarize subgraphs, forming a lifted representation of interesting subgraphs. This sophisticated approach has been useful for biological endpoints that describe processes in organisms involving metabolic activity. I conclude from the experiments that fragment-based descriptors may be useful in such settings, but that a more elaborate representation than traditional mining of frequent or correlated subgraphs would be needed. Specifically, none of the other types of fragment descriptors could compete in this setting. The question whether LAST-PM should be preferred over BBRC mining whenever possible, since it produces more elaborate patterns, must be answered with "no", however. In my experience, LAST-PM descriptors show superior predictivity on datasets describing complex biological endpoints (see chapter 7), but apart from that BBRC descriptors are competitive and can be computed very fast (given sensible constraint parameters). In this sense, LAST-PM cannot be considered a "successor" to BBRC mining.

Machine learning may extract relevant knowledge from structured databases, for example in the form of informative patterns. This can be shown on an implicit level through predictive ability when used in statistical models. However, patterns found by machine learning algorithms were also successfully matched with human expert knowledge on an explicit level (*cf.* section 6.1.3). I have observed that, oftentimes, patterns found by machine learning provides a more detailed, fine-grained perspective, whereas expert knowledge tends to generalize.

Quite obviously to me, not every pattern found by statistical methods is interpretable for experts. Sometimes, the automatically derived representation is quite

distant from what an expert may expect. This does not mean it is wrong – even experts need to constantly refactor their knowledge – but perhaps just hard to grasp. Of course, also the data could be insufficient in quality or size.
However, expert knowledge may be sometimes wrong too, due to over-generalization. In sophisticated settings, e.g. when complex metabolic processes are involved, this is not surprising.
I conclude that a combination of data-driven and expert techniques would be most useful when ultimately applied to real-world problems, mutually contributing to and controlling each other, as for example in the study by Wicker *et al.* [63].

8.3 Future Work

The algorithms presented in this work are depend critically on reasonably chosen constraint parameters. Most important are sensible selections of minimum frequency and minimum correlation thresholds. Finding appropriate settings is currently up to the user, which can be guess work and time consuming. Thus, efficient methods to find characteristics of a dataset at hand, associated with the quality of parameter choices would be very valuable. Such characteristics may include mean graph size, mean graph density (number of interconnected node pairs), or structural diversity of graphs, or others. A promising approach is structural graph clustering, for which recently a very efficient method has been proposed [56]. Here, the molecular graphs are clustered according to maximum common substructure (more specifically an approximation to it). The clustering is non-exclusive and non-exhaustive, so that instances can belong to more than one cluster, or to no cluster. The method illustrates some of the above properties, which could be functionally tied to parameter selection in a future study.

We have also shown that the descriptors in this work are useful to determine chemical similarity (*cf.* section 5.7.1). Thus, they could be also used for clustering chemical space, for example using Tanimoto similarity based on fingerprints representing the mined patterns. Since the presented methods work very efficiently, this could even be done in an on-demand setting, for example when screening a large database of chemical compounds for instances similar to a certain query compound.

Extensions of BBRC and LAST-PM for the multinomial and regression setting, i.e., with more than two target classes or a numeric endpoint variable, have already been implemented. In the multinomial setting, the method of dynamic upper bound pruning (*cf.* section 5.6.1) can be generalized from the binomial setting

in a straightforward manner. For regression, however, no such efficient pruning technique is available, since the corresponding correlation function (Kolmogorov-Smirnov test) lacks the convexity property.

Although the regression setting could be approximated by binning the response variable into classes, with n classes, the number of tests that must be performed on each pattern to calculate its upper bound is 2^n, which is prohibitive. Therefore, the current implementation is restricted to maximally five target classes. Techniques to overcome this will be the topic of future research.

Bibliography

[1] Amir Ahmad and Lipika Dey. A Feature Selection Technique for Classificatory Analysis. *Pattern Recognition Letters*, 26(1):43–56, 2005.

[2] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, Jeremy Besson, and Mohammed J. Zaki. ORIGAMI: Mining Representative Orthogonal Graph Patterns. *ICDM 2007. Seventh IEEE International Conference on Data Mining*, pages 153–162, Oct. 2007.

[3] Christopher G. Atkeson, Andrew W. Moore, and Stefan Schaal. Locally Weighted Learning. *Artificial Intelligence Review*, 11(1-5):11–73, 1997.

[4] Romualdo Benigni, Cecilia Bossa, Tatiana Netzeva, and Andrew Worth. *Collection and Evaluation of (Q)SAR Models for Mutagenicity and Carcinogenicity*, chapter 4.1. European Commission Joint Research Centre, 2007.

[5] Björn Bringmann, Siegfried Nijssen, and Albrecht Zimmermann. From Local Patterns to Classification Models. In Saso Džeroski, Bart Goethals, and Pance Panov, editors, *Inductive Databases and Constraint-Based Data Mining*, pages 127–154. Springer New York, 2010.

[6] Björn Bringmann, Albrecht Zimmermann, Luc De Raedt, and Siegfried Nijssen. Don't Be Afraid of Simpler Patterns. In *Proceedings of the 10th European conference on Principle and Practice of Knowledge Discovery in Databases*, PKDD'06, pages 55–66, Berlin, Heidelberg, 2006. Springer-Verlag.

[7] Yun Chi, Richard R. Muntz, Siegfried Nijssen, and Joost N. Kok. Frequent Subtree Mining - An Overview. *Fundamenta Informaticae*, 66(1-2):161–198, 2004.

[8] Yun Chi, Yi Xia, Yirong Yang, and Richard R. Muntz. Mining Closed and Maximal Frequent Subtrees From Databases of Labeled Rooted Trees. *IEEE Transactions on Knowledge and Data Engineering*, 17:190–202, February 2005.

[9] Diane J. Cook and Lawrence B. Holder. Substructure Discovery Using Minimum Description Length and Background Knowledge. *Journal of Artificial Intelligence Research*, 1(1):231–255, 1994.

[10] Stephen A. Cook. The Complexity of Theorem-Proving Procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM.

[11] Luc De Raedt. A Perspective on Inductive Databases. *SIGKDD Explorations Newsletter*, 4:69–77, December 2002.

[12] Hong Fang, Weida Tong, Leming M. Shi, Robert Blair, Roger Perkins, William Branham, Bruce S. Hass, Qian Xie, Stacy L. Dial, Carrie L. Moland, and Daniel M. Sheehan. StructureActivity Relationships for A Large Diverse Set of Natural, Synthetic, and Environmental Estrogens. *Chemical Research in Toxicology*, 14(3):280–294, 2001.

[13] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition (2nd Ed.)*. Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[14] Thomas Gärtner. Kernfunktionen für Strukturierte Daten. *Ausgezeichnete Informatikdissertationen 2005*, D-6:29–38, 2006.

[15] Rajarshi Guha, Michael T. Howard, Geoffrey R. Hutchison, Peter Murray-Rust, Henry S. Rzepa, Christoph Steinbeck, Joerg K. Wegner, and Egon L. Willighagen. The Blue Obelisk – interoperability in Chemical Informatics. *Journal of Chemical Information and Modeling*, 46:991–998, 2006.

[16] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1):10–18, 2009.

[17] Barry Hardy, Nicki Douglas, Christoph Helma, Micha Rautenberg, Nina Jeliazkova, Vedrin Jeliazkov, Ivelina Nikolova, Romualdo Benigni, Olga Tcheremenskaia, Stefan Kramer, Tobias Girschick, Fabian Buchwald, Joerg Wicker, Andreas Karwath, Martin Gutlein, Andreas Maunz, Haralambos Sarimveis, Georgia Melagraki, Antreas Afantitis, Pantelis Sopasakis, David Gallagher, Vladimir Poroikov, Dmitry Filimonov, Alexey Zakharov, Alexey Lagunin, Tatyana Gloriozova, Sergey Novikov, Natalia Skvortsova, Dmitry Druzhilovsky, Sunil Chawla, Indira Ghosh, Surajit Ray, Hitesh Patel, and Sylvia Escher. Collaborative Development of Predictive Toxicology Applications. *Journal of Cheminformatics*, 2(1):7, 2010.

[18] Kosuke Hashimoto, Ichigaku Takigawa, Motoki Shiga, Minoru Kanehisa, and Hiroshi Mamitsuka. Mining Significant Tree Patterns in Carbohydrate Sugar Chains. *Bioinformatics*, 24(16):i167–i173, 2008.

[19] Linnan He and Peter C. Jurs. Assessing the Reliability of A QSAR Model's Predictions. *Journal of Molecular Graphics and Modelling*, 23:503–523, 2005.

[20] Christoph Helma. *Predictive Toxicology*, chapter 1. CRC Press, Boca Raton, FLA, USA, 2004.

[21] Christoph Helma. Lazy Structure-Activity Relationships (lazar) for the Prediction of Rodent Carcinogenicity and Salmonella Mutagenicity. *Molecular Diversity*, pages 147–158, 2006.

[22] Heiko Hofer, Christian Borgelt, and Michael R. Berthold. Large Scale Mining of Molecular Fragments with Wildcards. *Intelligent Data Analysis*, 8(5):495–504, 2004.

[23] John D. Holliday, C. Hu, and Peter Willett. Grouping of Coefficients for the Calculation of Inter-Molecular Similarity and Dissimilarity Using 2D Fragment Bit-strings. *Combinatorial Chemistry and High Throughput Screening*, 5(2):155–66, 2002.

[24] Tamás Horváth and Jan Ramon. Efficient Frequent Connected Subgraph Mining in Graphs of Bounded Tree-width. *Theoretical Computer Science*, 411:2784–2797, June 2010.

[25] Tamás Horváth, Jan Ramon, and Stefan Wrobel. Frequent Subgraph Mining in Outerplanar Graphs. *Data Mining and Knowledge Discovery*, 21:472–508, 2010.

[26] Tingjun Hou and Xiaojie Xu. ADME evaluation in Drug Discovery. 3. modeling blood-brain barrier partitioning using simple molecular descriptors. *Journal of Chemical Information and Computer Sciences*, 43(6):2137–2152, Oct 2003.

[27] Jun Huan, Wei Wang, Jan Prins, and Jiong Yang. SPIN: Mining Maximal Frequent Subgraphs From Graph Databases. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 581–586, New York, NY, USA, 2004. ACM.

[28] Tomasz Imielinski and Heikki Mannila. A Database Perspective on Knowledge Discovery. *Communications of the ACM*, 39(11):58–64, 1996.

[29] Akihiro Inokuchi. Mining Generalized Substructures From A Set of Labeled Graphs. *IEEE International Conference on Data Mining*, 0:415–418, 2004.

[30] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data. In *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, PKDD '00, pages 13–23, London, UK, 2000. Springer-Verlag.

[31] Katharina Jahn and Stefan Kramer. Optimizing gSpan for Molecular Datasets. In *Proceedings of the Third International Workshop on Mining Graphs, Trees and Sequences (MGTS-2005)*, 2005.

[32] Joanna S. Jaworska, Mike H. Comber, C. Auer, and C. J. Van Leeuwen. Summary of A Workshop on Regulatory Acceptance of (Q)SARs for Human Health and Environmental Endpoints. *Environmental Health Perspectives*, 111(10):1358–1360, 2003.

[33] Jeroen Kazius, Siegfried Nijssen, Jost N. Kok, Thomas Bäck, and Adriaan P. Ijzerman. Substructure Mining Using Elaborate Chemical Representation. *Journal of Chemical Information and Modeling*, 46:597–605, 2006.

[34] Ross D. King, Ashwin Srinivasan, and Luc Dehaspe. WARMR: A Data Mining Tool for Chemical Data. *Journal of Computer-Aided Molecular Design*, 15(2):173–181, Feb 2001.

[35] Jeroen De Knijf. Mining Tree Patterns With Almost Smallest Supertrees. In *SIAM International Conference on Data Mining*, pages 61–71. SIAM, 2008.

[36] Stefan Kramer, Luc De Raedt, and Christoph Helma. Molecular Feature Mining in HIV Data. In *KDD '01: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 136–143, New York, NY, USA, 2001. ACM.

[37] Michihiro Kuramochi and George Karypis. Frequent Subgraph Discovery. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, ICDM '01, pages 313–320, Washington, DC, USA, 2001. IEEE Computer Society.

[38] Hu Li, Chun Wei Yap, Choong Yong Ung, Ying Xue, Zhi Wei Cao, and Yu Zong Chen. 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):1376–1384, Aug 2005.

[39] Yuhua Li, Quan Lin, Ruixuan Li, and Dongsheng Duan. TGP: Mining Top-k Frequent Closed Graph Pattern Without Minimum Support. In Longbing Cao, Yong Feng, and Jiang Zhong, editors, *Advanced Data Mining and Applications*, volume 6440 of *Lecture Notes in Computer Science*, pages 537–548. Springer Berlin / Heidelberg, 2010.

[40] Heikki Mannila and Hannu Toivonen. Levelwise Search and Borders of Theories in Knowledge Discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.

[41] Andreas Maunz and Christoph Helma. Prediction of Chemical Toxicity with Local Support Vector Regression and Activity-specific Kernels. *SAR and QSAR in Environmental Research*, 19(5-6):413–431, July 2008.

[42] Shinichi Morishita and Jun Sese. Traversing Itemset Lattice With Statistical Metric Pruning. In *Symposium on Principles of Database Systems*, pages 226–236, 2000.

[43] Siegfried Nijssen and Joost N. Kok. Efficient Discovery of Frequent Unordered Trees. In *In First International Workshop on Mining Graphs, Trees and Sequences*, pages 55–64, 2003.

[44] Siegfried Nijssen and Joost N. Kok. A Quickstart in Frequent Structure Mining Can Make a Difference. In *KDD '04: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 647–652, New York, NY, USA, 2004. ACM.

[45] Siegfried Nijssen and Joost N. Kok. Frequent Subgraph Miners: Runtime Don't Say Everything. In *Proceedings of the International Workshop on Mining and Learning With Graphs (MLG 2006)*, pages 173–180, 2006.

[46] William H. Press, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C*, chapter 14.3. Cambridge University Press, 1993.

[47] Ann M. Richard. Commercial Toxicology Prediction Systems: A Regulatory Perspective. *Toxicology Letters*, 102-103:611–616, Dec 1998.

[48] Ann M. Richard and ClarLynda R. Williams. *Quantitative Structure-Activity Relationship (QSAR) Models of Mutagens and Carcinogens*, chapter 5. CRC Press, Boca Raton, FLA, USA, 2003.

[49] Ulrich Rückert and Stefan Kramer. Optimizing Feature Sets for Structured Data. In *ECML '07: Proceedings of the 18th European Conference on Machine Learning*, pages 716–723, Berlin, Heidelberg, 2007. Springer-Verlag.

[50] Ulrich Rückert and Stefan Kramer. Towards a Framework for Relational Learning and Propositionalization. In M. Ceci D. Malerba, A. Appice, editor, *Proceedings of the 6th Workshop on Multi-Relational Data Mining at the 18th European Conference on Machine Learning*, 2007.

[51] Christine L. Russom, Steven P. Bradbury, Steven J. Broderius, Dean E. Hammermeister, and Robert A. Drummond. Predicting Modes of Action From Chemical Structure: Acute Toxicity in the Fathead Minnow (Pimephales Promelas). *Environmental Toxicology and Chemistry*, 16:948–967, 1997.

[52] Leander Schietgat, Fabrizio Costa, Jan Ramon, and Luc De Raedt. Effective Feature Construction by Maximum Common Subgraph Sampling. *Machine Learning*, 83(2):137–161, 2011.

[53] Leander Schietgat, Jan Ramon, Maurice Bruynooghe, and Hendrik Blockeel. An Efficiently Computable Graph-based Metric for the Classification of Small Molecules. In *Proceedings of the 11th International Conference on Discovery Science*, DS '08, pages 197–209, Berlin, Heidelberg, 2008. Springer-Verlag.

[54] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[55] Hannes Schulz, Christian Kersting, and Andreas Karwath. ILP, the Blind, and the Elephant: Euclidean Embedding of Co-proven Queries. In *Proceedings of the 19th International Conference on Inductive Logic Programming*, ILP'09, pages 209–216, Berlin, Heidelberg, 2010. Springer-Verlag.

[56] Madeleine Seeland, Simon Berger, Alexandros Stamatakis, and Stefan Kramer. Parallel Structural Graph Clustering. In *Proceedings of the 2011 European Conference of Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, volume 3 of *ECML PKDD'11*, pages 256–272. Springer-Verlag, Berlin, Heidelberg, 2011.

[57] Madeleine Seeland, Tobias Girschick, Fabian Buchwald, and Stefan Kramer. Online Structural Graph Clustering Using Frequent Subgraph Mining. In José Balcázar, Francesco Bonchi, Aristides Gionis, and Michèle Sebag, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 6323 of *Lecture Notes in Computer Science*, pages 213–228. Springer Berlin / Heidelberg, 2010.

[58] Ashwin Srinivasan, Ross D. King, Stephen H. Muggleton, and Michael J. E. Sternberg. The Predictive Toxicology Evaluation Challenge. In *Proceedings*

of the 15th International Joint Conference on Artifical Intelligence - Volume 1, IJCAI'97, pages 4–9, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[59] Claudia Suenderhauf, Felix Hammann, Andreas Maunz, Christoph Helma, and Jörg Huwyler. Combinatorial QSAR Modeling of Human Intestinal Absorption. *Molecular Pharmaceutics*, 8(1):213–224, 2011.

[60] S. Joshua Swamidass, Jonathan Chen, Peter Phung, Liva Ralaivola, and Pierre Baldi. Kernels for Small Molecules and the Prediction of Mutagenicity, Toxicity and Anti-cancer Activity. *Bioinformatics*, 21:i359–i368(1), June 2005.

[61] L.A. Székely and Hua Wang. On Subtrees of Trees. *Advances in Applied Mathematics*, 34(1):138 – 155, 2005.

[62] Lini T. Thomas, Satyanarayana R. Valluri, and Kamalakar Karlapalem. MARGIN: Maximal Frequent Subgraph Mining. *ACM Transactions on Knowledge Discovery from Data*, 4:10:1–10:42, October 2010.

[63] Jörg Wicker, Kathrin Fenner, Lynda Ellis, Larry Wackett, and Stefan Kramer. Predicting Biodegradation Products and Pathways: A Hybrid Knowledge-Based and Machine Learning-Based Approach. *Bioinformatics*, 26(6):814–821, 2010.

[64] Frank Wilcoxon. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[65] Marc Wörlein, Thorsten Meinl, Ingrid Fischer, and Michael Philippsen. A Quantitative Comparison of the Subgraph Miners MoFa, gSpan, FFSM, and Gaston. In *Proceedings of the 9th European conference on Principles and Practice of Knowledge Discovery in Databases*, PKDD'05, pages 392–403, Berlin, Heidelberg, 2005. Springer-Verlag.

[66] Xifeng Yan and Jiawei Han. gSpan: Graph-based Substructure Pattern Mining. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, ICDM '02, pages 721–, Washington, DC, USA, 2002. IEEE Computer Society.

[67] Xifeng Yan and Jiawei Han. Closegraph: Mining Closed Frequent Graph Patterns. In *KDD '03: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 286–295, New York, NY, USA, 2003. ACM.

[68] Fumitaka Yoshida and John G. Topliss. QSAR Model for Drug Human Oral Bioavailability. *Journal of Medicinal Chemistry*, 43(13):2575–2585, Jun 2000.

[69] Mohammed J. Zaki. Efficiently Mining Frequent Trees in A Forest. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 71–80, New York, NY, USA, 2002. ACM.

[70] Qiang Zhu, Xiaoyue Wang, Eamonn Keogh, and Sang-Hee Lee. Augmenting the Generalized Hough Transform to Enable the Mining of Petroglyphs. In *KDD '09: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1057–1066, New York, NY, USA, 2009. ACM.

Pre-Published Parts of this Dissertation

- Andreas Maunz, Christoph Helma, and Stefan Kramer. Efficient Mining for Structurally Diverse Subgraph Patterns in Large Molecular Databases [1]. *Machine Learning*, 83:193–218, 2011.
- Andreas Maunz, Christoph Helma, Tobias Cramer, and Stefan Kramer. Latent Structure Pattern Mining. In *Proceedings of ECML PKDD 2010*, pages 353–368. Springer Berlin / Heidelberg, 2010.
- Andreas Maunz, Christoph Helma, and Stefan Kramer. Large-Scale Graph Mining Using Backbone Refinement Classes. In *KDD '09: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 617–626, New York, NY, USA, 2009. ACM.
- Andreas Maunz and Christoph Helma. Prediction of Chemical Toxicity with Local Support Vector Regression and Activity-Specific Kernels. *SAR and QSAR in Environmental Research*, 19(5-6):413–431, July 2008.

[1] Proceedings article of: Andreas Maunz, Christoph Helma, and Stefan Kramer: Large Scale Graph Mining using Backbone Refinement Classes. In 7th International Workshop on Mining and Learning with Graphs (MLG 2009).

Appendix A

LAST-SMARTS

LAST-PM output is converted to a special type of SMARTS patterns (LAST-SMARTS). LAST-SMARTS are valid SMARTS strings according to the standard set forth by Daylight Inc. [1].

While SMILES (**S**implified **M**olecular **I**nput **L**ine **E**ntry **S**pecification), a popular ASCII string notation for chemicals, denotes molecules, a SMARTS (**SM**iles **AR**bitrary **T**arget **S**pecification) string describes molecular fragments. The SMILES/SMARTS language pair is supported by most computational chemistry software. SMILES strings are obtained by printing the symbol nodes encountered in a depth-first tree traversal of a chemical graph (see Fig. A.1[2]).

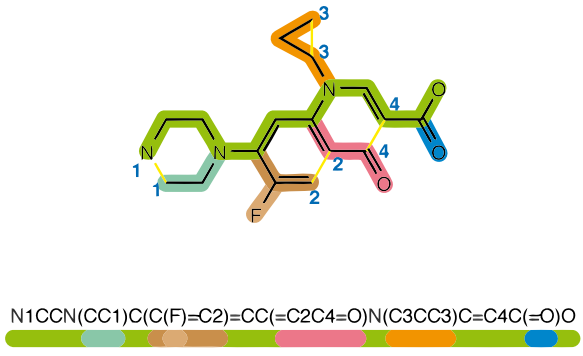


FIGURE A.1: An example for SMILES generation

[1] See `http://www.daylight.com/dayhtml/doc/theory/theory.smarts.html`.

[2] Taken from `http://en.wikipedia.org/wiki/Simplified_molecular_input_line_entry_specification`.

SMARTS strings can be thought of as "regular expressions for chemicals", i.e. a query language for molecular fragments. Besides the symbols allowed in SMILES, SMARTS allow a variety of wildcard patterns. LAST-SMARTS implement some of these. Formally, LAST-SMARTS are defined as follows in Extended Backus-Naur Form:

```
ANN := '17' | '35' | '5' | '6' | '7' | '8' | '15' | '16' | '9' | '53'
AN := ANN '&a' | ANN
A  := (AN ',' A) | AN
SB := '-' | '=' | '#' | ':'
E  := (SB ',' E) | SB
N  := '[#' A ']'
L  := N '(' E LS ')' ('(' E N ')')+
LR := (L ',' LR) | L
BN := '[#' A ';$' '(' LR ')' ']' '(~*)'+
LS := (N | BN) | LS E (N | BN)
```

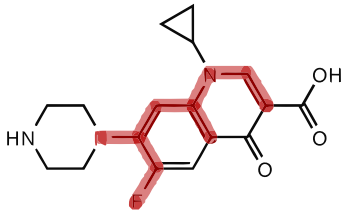
The definition uses an organic subset of atoms (ANN), possibly with aromatic annotation (AN), and four different bond types (SB). Bonds (E) and atoms (A) can be composed of several types, with the semantics of logical OR.

LAST-SMARTS employ *recursive SMARTS* to describe optional parts of the structure for ambiguities larger than a single atom or bond type. A recursive SMARTS is a SMARTS embedded in a SMARTS using the embeddeding operator '$'(BN). It is composed of several substructures, with the semantics of logical OR (LR). Thus this construct is analogous to E and A respectively, but describes fragments larger than a single atom or bond.

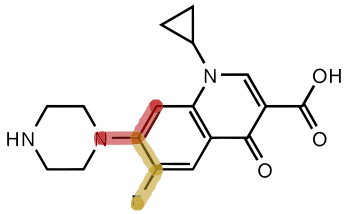
The embedded SMARTS consists of the atom itself, and so-called *back links* and *forward links*, describing precisely the one-step neighborhood (see example below). The back and forward links are needed, since the standard is truly recursive, i.e. nothing inside the `$(...)` is identified with the outside. For the same reason, `(~*)` (arbitrary branch) is used to enforce that at least one optional branch is actually attached.

Example A.1. *The following LAST-SMARTS describes two branches as optional (broken up on several lines for demonstration):*

```
1  [#7,#9]
2  [#6&a;$(
```



(a) Matches without optional parts.



(b) Matches with optional parts.

FIGURE A.2: Example for matching LAST-SMARTS.

```
3  [#6&a][#7]([#6&a][#9])=[#6&a]
4        ),$(
5  [#6&a][#7]([#6&a]=[#8])=[#6&a]
6        )
7  ](~*)
8  =[#6&a]
```

This denotes a nitrogen or fluorine (1) (single-)connected to an aromatic carbon (2) double-connected to an aromatic carbon (8). The middle carbon's local environment (2-6), attached via conjunction (';'), is described using two recursive SMARTS (3,5), denoted by `$(...)`, and a disjunction (',').

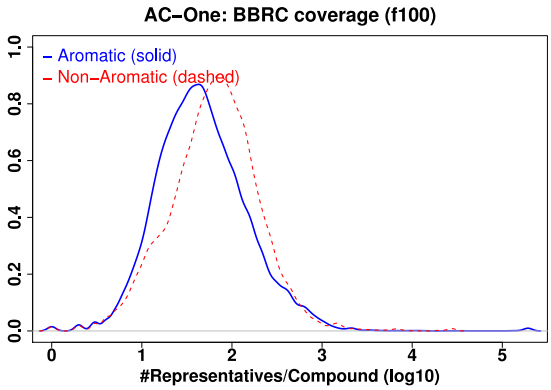
The example demonstrates optional fragments larger than a single atom or bond. Ambiguities on the atomic level do not require recursive SMARTS, the example demonstrates this in the first atom. Fig. A.2(a) shows all matchings without the recursive parts on the chemical from Fig. A.1. However, since at least one optional part is required, the match in Fig. A.2(b) is obtained, with the optional part marked lighter in contrast to the darker rest of the match.

Note: Variants *msa* and *nls* produce LAST-SMARTS with recursive SMARTS, while *nop* disallows optional parts of the structures and allows ambiguities only on the atom / edge level.

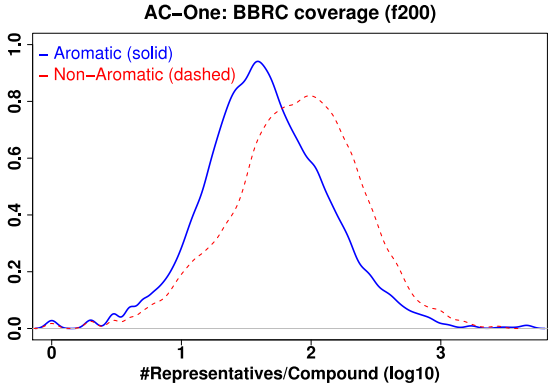
Appendix B

BBRC-Coverage

As discussed in section 5.4.1.2, the normality of the coverage distribution increased for higher minimum frequencies. Fig. B.1 shows the corresponding density curves.



(a) Minimum Frequency: 100



(b) Minimum Frequency: 200

FIGURE B.1: Coverage (density) of BBRC patterns.

Appendix C

Online-Resources

C.1 Source Code

The source code for both BBRC and LAST-PM is freely available. For each algorithm, a dedicated library has been developed (`libbbrc` and `liblast`). The code is hosted on Github, a collaborative source code management platform (`http://github.com/amaunz/fminer2`).

A thin frontend application is provided (called `fminer`) for use on the command line. Moreover, the exact same functionality is exposed as REST web service in the OpenTox framework [17], which allows for convenient remote use without having to install applications locally.

Visit the main websites at `http://bbrc.maunz.de` and `http://last-pm.maunz.de` for information on how to

- Download, build and use `fminer` locally on your computer.
- Query the `fminer` web service.
- Build the libraries for several scripting languages (currently Ruby, Python, and Java).
- Reproduce some main experiments of this work.

Note: Please check back frequently for bugfixes and new functionality, since the libraries are being developed constantly.

C.2 Datasets

The datasets that have been used in this work (apart from Chapter 7) can be obtained as follows: Visit `http://github.com/amaunz/` and branch to the subdirectories corresponding to the desired datasets (see section 4.3).

OFS Datasets	`http://github.com/amaunz/ofsdata`
CPDB Datasets	`http://github.com/amaunz/cpdbdata`
Large-Scale Datasets	`http://github.com/amaunz/data-yeast-acl`

Please refer to the README that accompanies each dataset for further information.

C.3 Animated Embeddings

The BBRC website at `http://bbrc.maunz.de` contains dynamic versions of the euclidean embeddings presented in section 5.5.1.2, allowing to zoom and pan the representation. An important feature that is not available in the printed version is that it allows to explore the actual embeddings of every single pattern by hovering over it with the mouse. Conversely, all patterns occurring in an instance will be marked when hovering over the instance, conveying a much more detailed picture of the embedding.

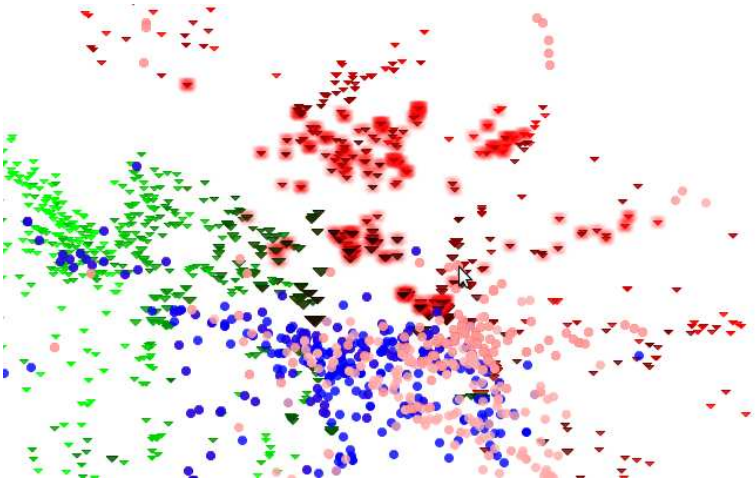


FIGURE C.1: Occurrences highlighted by hovering the mouse pointer.

List of Figures

List of Tables

Andreas Maunz

Oncotest GmbH
Institute for Experimental Oncology
Prof. Dr. H.-H. Fiebig
Am Flughafen 12-14
79108 Freiburg, Germany

Phone: +49 761 51559-0
Email: andreas@maunz.de

http://cs.maunz.de

Education

Current Status

2008 - **Ph.D. student** at Technische Universität München
Advisor: Prof. Dr. Stefan Kramer.

Degrees

July 2007 **Diplom** (german M. Sc.) in Computer Science
at Albert-Ludwigs-Universität Freiburg
Advisor: Prof. Dr. Rolf Backofen.

Awards

2011 Published article in "Machine Learning" special issue
2007 Appearance among the top third of Freiburg CS graduates in 2007
2003 - 2007 Scholarship granted by Hans-Böckler Stiftung

Employment

2013 - Bioinformatician at Oncotest GmbH, Freiburg
Institute for Experimental Oncology
Prof. Dr. Heinz-Herbert Fiebig
2007 - 2012 Scientific position at Physikalisches Institut der Universität Freiburg
Prof. Dr. Jens Timmer
2004 - 2005 Cooperation with the Machine Learning and Natural Language Processing Lab in Freiburg as student worker

International Research Projects

2011 - 2012	Cooperation with Nestlé AG (Vevey, CH) Structure based modelling of chronic toxicity and carcinogenicity. Contribution: Responsible for the complete work on our side (publication in progress).
2008 - 2011	,,OpenTox" (EU seventh framework programme) Development of an interoperable API framework for Predictive Toxicology. Contribution: Draft, evaluation, and implementation of data mining and machine learning algorithms.
2005 - 2007	,,Sens-it-iv" (EU seventh framework programme) Development of *in vitro* Alternatives to Animal Testing in risk assessment of allergens. Contribution: Implementation and operation of an inductive database for experimental data.

National Research Projects

2011 - 2012	Project ,,REACH" of the Federal Ministry of Education and Research

Research Focus And Interest

Generally: In toxicology, the aim is to understand the biochemical mechanisms involved and the degree to which chemicals induce toxicological activity in living organisms with respect to a well-defined endpoint. This mostly involves chemical expert knowledge, tailored to specific biochemical interactions (top-down). Quite conflicting, statistical approaches use general toxic endpoints and activity values gathered for a wide range of structures and are primarily driven by information inherently present in the data (bottom-up). My aim is the development of powerful, yet universally applicable algorithms that narrow this gap.

Specifically: I am mainly concerned with structural alerts, i.e. structural motifs in chemical structures associated with biological activity. The objective is to establish an inductive database of compounds and activity values that users can query for fragments relevant to the endpoint of interest, providing, among others,

frequency and statistical constraints. For this purpose, efficient subgraph enumeration including pruning techniques for statistical and frequency constraints are employed.

Publications

Journal Articles and Book Chapters

- Andreas Maunz, Martin Gütlein, Micha Rautenberg, David Vorgrimmler, Denis Gebele, and Christoph Helma. lazar: A modular Predictive Toxicology Framework. *Frontiers in Pharmacology*, 4(38), 2013.
- Claudia Suenderhauf, Felix Hammann, Andreas Maunz, Christoph Helma, and Jürgen Drewe. Combinatorial QSAR Modeling of Human Intestinal Absorption. *Molecular Pharmaceutics*, 8(1):213–224, 2011.
- Andreas Maunz, Christoph Helma, and Stefan Kramer. Efficient Mining for Structurally Diverse Subgraph Patterns in Large Molecular Databases [1]. *Machine Learning*, 83:193–218, 2011.
- Barry Hardy, Nicki Douglas, Christoph Helma, Micha Rautenberg, Nina Jeliazkova, Vedrin Jeliazkov, Ivelina Nikolova, Romualdo Benigni, Olga Tcheremenskaia, Stefan Kramer, Tobias Girschick, Fabian Buchwald, Joerg Wicker, Andreas Karwath, Martin Gutlein, Andreas Maunz, Haralambos Sarimveis, Georgia Melagraki, Antreas Afantitis, Pantelis Sopasakis, David Gallagher, Vladimir Poroikov, Dmitry Filimonov, Alexey Zakharov, Alexey Lagunin, Tatyana Gloriozova, Sergey Novikov, Natalia Skvortsova, Dmitry Druzhilovsky, Sunil Chawla, Indira Ghosh, Surajit Ray, Hitesh Patel, and Sylvia Escher. Collaborative Development of Predictive Toxicology Applications. *Journal of Cheminformatics*, 2(1):7, 2010.
- Felix Hammann, Heike Gutmann, Ursula Jecklin, Andreas Maunz, Christoph Helma, and Jürgen Drewe. Development of Decision Tree Models for Substrates, Inhibitors, and Inducers of p-Glycoprotein. *Current Drug Metabolism*, 10:339–346(8), May 2009.

[1] Proceedings article of: Andreas Maunz, Christoph Helma, and Stefan Kramer: Large Scale Graph Mining using Backbone Refinement Classes. In 7th International Workshop on Mining and Learning with Graphs (MLG 2009).

- Andreas Maunz and Christoph Helma. Prediction of Toxic Effects of Pharmaceutical Agents. In Konstantin V. Balakin, editor, *Pharmaceutical Data Mining*, chapter 5, pages 145–173. Wiley, New York, 2009.

- Andreas Maunz and Christoph Helma. Prediction of Chemical Toxicity with Local Support Vector Regression and Activity-Specific Kernels. *SAR and QSAR in Environmental Research*, 19(5-6):413–431, July 2008.

Peer-Reviewed Conferences (Full Proceedings)

- Andreas Maunz, David Vorgrimmler, and Christoph Helma. Out-of-Bag Discriminative Graph Mining. In *Proceedings of the 28th Symposium On Applied Computing, accepted*, 2013.

- Andreas Maunz, Christoph Helma, Tobias Cramer, and Stefan Kramer. Latent Structure Pattern Mining. In *Proceedings of ECML PKDD 2010*, pages 353–368. Springer Berlin / Heidelberg, 2010.

- Andreas Maunz, Christoph Helma, and Stefan Kramer. Large-Scale Graph Mining Using Backbone Refinement Classes. In *KDD '09: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 617–626, New York, NY, USA, 2009. ACM.

Professional Activities

Workshop Presentations:

- A. Maunz. Elaborate Graph Mining: Exploiting Structural Invariants and Latent Information in Graph Databases to predict REACH-relevant Endpoints, presented at the OpenTox InterAction Meeting, Summer 2011, Technical University Munich, Munich, Germany (9-12 August 2011)

- A. Maunz, C. Helma, T. Cramer, and S. Kramer. Latent Structure Pattern Mining, presented at the eCheminfo Community of Practice InterAction Meeting, Oxford University, Oxford, UK (01-05 August 2010)

- A. Maunz, C. Helma, and S. Kramer: Large Scale Graph Mining using Backbone Refinement Classes. In 7th International Workshop on Mining and Learning with Graphs, Leuven, Belgium (02-04 July 2009).

  The submitted abstract was one of eight (out of 21) that were selected for oral presentation at the conference. Following the conference, the work was selected for publication in the joint MLG/SRL/ILP workshop special issue of the Machine Learning Journal (among five other works out of 83).

- A. Maunz, C. Helma. New Lazar Developments, presented at the eCheminfo Community of Practice InterAction Meeting, Autumn 2008, Bryn Mawr College, Philadelphia, USA (13-17 October 2008).

- A. Maunz. Instance-based Regression Models for Quantitative Biological Activities using Support Vector Machines and Multilinear Models, presented at the Scarlet Workshop on in silico methods for carcinogenicity and mutagenicity, Milano, Italy (April 2008).

(Co-)Reviewing

- ECML/PKDD 2010
- ICDM 2010
- KDD 2011
- Journal of Intelligent Information Systems
- Journal of Chemical Information and Modelling.

Reviewing: ECML/PKDD 2010, ICDM 2010, KDD 2011, Journal of Chemical Information and Modelling, Journal of Intelligent Information Systems.