The Efficient Discovery of Interesting Closed Pattern Collections

Dissertation
zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt
von
Mario Boley
aus
Leverkusen

Bonn, 2010

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter: Prof. Dr. Stefan Wrobel
2. Gutachter: Prof. Dr. Rainer Manthey
Tag der Promotion: 26.9.2011
Erscheinungsjahr: 2011

Mario Boley

University of Bonn
Department of Computer Science III

and

Fraunhofer Institute for Intelligent Analysis
and Information Systems IAIS

Schloss Birlinghoven
53754 Sankt Augustin
Germany

`mario.boley@iais.fraunhofer.de`

Declaration

I, Mario Boley, confirm that this work is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g. ideas, equations, figures, text, tables, programs) are properly acknowledged at the point of their use. A full list of the references employed has been included.

Abstract

Enumerating closed sets that are frequent in a given database is a fundamental data mining technique that is used, e.g., in the context of market basket analysis, fraud detection, or Web personalization. There are two complementing reasons for the importance of closed sets—one semantical and one algorithmic: closed sets provide a condensed basis for non-redundant collections of interesting local patterns, and they can be enumerated efficiently. For many databases, however, even the closed set collection can be way too large for further usage and correspondingly its computation time can be infeasibly long. In such cases, it is inevitable to focus on smaller collections of closed sets, and it is essential that these collections retain both: controlled semantics reflecting some notion of interestingness as well as efficient enumerability. This thesis discusses three different approaches to achieve this: constraint-based closed set extraction, pruning by quantifying the degree or strength of closedness, and controlled random generation of closed sets *instead* of exhaustive enumeration.

For the original closed set family, efficient enumerability results from the fact that there is an inducing efficiently computable closure operator and that its fixpoints can be enumerated by an amortized polynomial number of closure computations. Perhaps surprisingly, it turns out that this connection does not generally hold for other constraint combinations, as the restricted domains induced by additional constraints can cause two things to happen: the fixpoints of the closure operator cannot be enumerated efficiently or an inducing closure operator does not even exist. This thesis gives, for the first time, a formal axiomatic characterization of constraint classes that allow to efficiently enumerate fixpoints of arbitrary closure operators as well as of constraint classes that guarantee the existence of a closure operator inducing the closed sets.

As a complementary approach, the thesis generalizes the notion of closedness by quantifying its strength, i.e., the difference in supporting database records between a closed set and all its supersets. This gives rise to a measure of interestingness that is able to select long and thus particularly informative closed sets that are robust against noise and dynamic changes. Moreover, this measure is algorithmically sound because all closed sets with a minimum strength again form a closure system that can be enumerated efficiently and that directly ties into the results on constraint-based closed sets. In fact both approaches can easily be combined.

In some applications, however, the resulting set of constrained closed sets is still intractably large or it is too difficult to find meaningful hard constraints at all (including values for their parameters). Therefore, the last part of this thesis presents an alternative algorithmic paradigm to the extraction of closed sets: instead of exhaustively listing a potentially exponential number of sets, randomly generate exactly the desired amount of them. By using the Markov chain Monte Carlo method, this generation can be performed according to any desired probability distribution that favors interesting patterns. This novel randomized approach complements traditional enumeration techniques (including those mentioned above): On the one hand, it is only applicable in scenarios that do not require deterministic guarantees for the output such as exploratory data analysis or global model construction. On the other hand, random closed set generation provides complete control over the number as well as the distribution of the produced sets.

Zusammenfassung

Das Aufzählen abgeschlossener Mengen (closed sets), die häufig in einer gegebenen Datenbank vorkommen, ist eine algorithmische Grundaufgabe im Data Mining, die z.B. in Warenkorbanalyse, Betrugserkennung oder Web-Personalisierung auftritt. Die Wichtigkeit abgeschlossener Mengen ist semantisch als auch algorithmisch begründet: Sie bilden eine nicht-redundante Basis zur Erzeugung von lokalen Mustern und können gleichzeitig effizient aufgezählt werden. Allerdings kann die Anzahl aller abgeschlossenen Mengen, und damit ihre Auflistungszeit, das Maß des effektiv handhabbaren oft deutlich übersteigen. In diesem Fall ist es unvermeidlich, kleinere Ausgabefamilien zu betrachten, und es ist essenziell, dass dabei beide o.g. Eigenschaften erhalten bleiben: eine kontrollierte Semantik im Sinne eines passenden Interessantheitsbegriffes sowie effiziente Aufzählbarkeit. Diese Arbeit stellt dazu drei Ansätze vor: das Einführen zusätzlicher Constraints, die Quantifizierung der Abgeschlossenheit und die kontrollierte zufällige Erzeugung einzelner Mengen *anstelle* von vollständiger Aufzählung.

Die effiziente Aufzählbarkeit der ursprünglichen Familie abgeschlossener Mengen rührt daher, dass sie durch einen effizient berechenbaren Abschlussoperator erzeugt wird und dass desweiteren dessen Fixpunkte durch eine amortisiert polynomiell beschränkte Anzahl von Abschlussberechnungen aufgezählt werden können. Wie sich herausstellt ist dieser Zusammenhang im Allgemeinen nicht mehr gegeben, wenn die Funktionsdomäne durch Constraints eingeschränkt wird, d.h., dass die effiziente Aufzählung der Fixpunkte nicht mehr möglich ist oder ein erzeugender Abschlussoperator unter Umständen gar nicht existiert. Diese Arbeit gibt erstmalig eine axiomatische Charakterisierung von Constraint-Klassen, die die effiziente Fixpunktaufzählung von beliebigen Abschlussoperatoren erlauben, sowie von Constraint-Klassen, die die Existenz eines erzeugenden Abschlussoperators garantieren.

Als ergänzenden Ansatz stellt die Dissertation eine Generalisierung bzw. Quantifizierung des Abgeschlossenheitsbegriffs vor, der auf der Differenz zwischen den Datenbankvorkommen einer Menge zu den Vorkommen all seiner Obermengen basiert. Mengen, die bezüglich dieses Begriffes *stark* abgeschlossen sind, weisen eine bestimmte Robustheit gegen Veränderungen der Eingabedaten auf. Desweiteren wird die gewünschte effiziente Aufzählbarkeit wiederum durch die Existenz eines effizient berechenbaren erzeugenden Abschlussoperators sichergestellt. Zusätzlich zu dieser algorithmischen Parallele zum Constraint-basierten Vorgehen, können beide Ansätze auch inhaltlich kombiniert werden.

In manchen Anwendungen ist die Familie der abgeschlossenen Mengen, zu denen die beiden oben genannten Ansätze führen, allerdings immer noch zu groß bzw. ist es nicht möglich, sinnvolle harte Constraints und zugehörige Parameterwerte zu finden. Daher diskutiert diese Arbeit schließlich noch ein völlig anderes Paradigma zur Erzeugung abgeschlossener Mengen als vollständige Auflistung, nämlich die randomisierte Generierung einer Anzahl von Mengen, die exakt den gewünschten Vorgaben entspricht. Durch den Einsatz der Markov-Ketten-Monte-Carlo-Methode ist es möglich die Verteilung dieser Zufallserzeugung so zu steuern, dass das Ziehen interessanter Mengen begünstigt wird. Dieser neue Ansatz bildet eine sinnvolle Ergänzung zu herkömmlichen Techniken (einschließlich der oben genannten): Er ist zwar nur anwendbar, wenn keine deterministischen Garantien erforderlich sind, erlaubt aber andererseits eine vollständige Kontrolle über Anzahl und Verteilung der produzierten Mengen.

Acknowledgements

This thesis was written at the Knowledge Discovery Department of Fraunhofer IAIS in Sankt Augustin under my thesis supervisor Prof. Dr. Stefan Wrobel and my additional advisor Dr. Tamas Horváth. I would like to express my gratitude not only for the thoughtful advice that both provided but also for the freedom, patience, and trust they granted me. The same holds true for Dr. Thomas Gärtner, the head of my research group CAML.

Among the many other people contributing to the motivating spirit at our institution I want to especially thank the individuals with whom I cooperated on research papers related to this thesis. In addition to the three aforementioned persons, these are Dr. Henrik Grosskreutz and Dr. Axel Poigné. Particularly, I really have to stress that many parts of this thesis would not exist if it were not for Henrik, our good mutual understanding, and of course his software craftsmanship.

Finally, I have to mention my linear algebra lecturer Prof. Dr. Karl Scherer, who recommended me to the German National Academic Foundation, and my friend Manuel Peelen, with whom I managed to laugh about such things as universal quantification, missing definitions, and logical reasoning in general. They both contributed to the not too natural development that led a moderately talented person away from the edge of dropping his undergraduate studies to finally submitting a doctoral thesis in a formal science. Thank you.

Ich ging ruhig aus dem Schatten ins Mondlicht, knöpfte den Überzieher auf und wärmte mich; dann ließ ich durch Erheben der Hände das Sausen der Nacht schweigen und fing zu überlegen an.

(Franz Kafka, Gespräch mit dem Betrunkenen)

Contents

List of Algorithms

1. Introduction

In the late 20th century, the advent of cheap mass storage media, powerful database technology, and the ubiquitous presence of the Internet initiated an enduring dynamic: digitally accessible data collections began to grow much faster than the computer-aided capability to analyze them. Driven by this development, "data mining" or "knowledge discovery from databases" constituted itself as an independent scientific area during the 1990s; its subject being "the nontrivial extraction of implicit, previously unknown, and potentially useful information from data" [Frawley et al., 1992]. Beside adapting methods from machine learning and statistics to large scale databases, the community brought up an original branch of research called "local pattern discovery" (see Hand [2002], Mannila [2002]). Local patterns indicate interesting phenomena within a given data collection that generally refer to only a subset of the data instead of fully describing it. There is a diverse set of problem domains where methods based on local patterns are successfully applied; examples include market basket analysis (see, e.g., Brijs et al. [1999]), fraud detection (see, e.g., Fawcett and Provost [1997]), and the design of effective recommender systems (see, e.g., Mobasher et al. [2001]).

A fundamental problem of local pattern discovery methods is, however, that they tend to produce an overwhelming mass of result patterns, many of which are redundant (see, e.g., Knobbe and Ho [2006]). This problem motivated the invention of algorithms that list *closed patterns*. Closed patterns are patterns that cannot be extended in a way that preserves their support, i.e., the fraction of the input database where the pattern is valid. This was an important step for two complementary reasons: closed patterns provide a condensed basis for non-redundant collections of interesting local patterns, and they can be enumerated efficiently. Unfortunately, for many databases even the closed pattern collection can be way too large for further usage and correspondingly its computation time can be infeasibly long. In such cases, it is inevitable to focus on smaller pattern families. The easiest way to produce such a family would be to use one of the efficient algorithms for enumerating all closed patterns and to stop it after some previously allocated computation time is exceeded. This approach, however, produces a collection of closed patterns that does not reflect any reasonable notion of interestingness but rather depends on the internal enumeration ordering of the particular algorithm.

Therefore, this thesis extends the research on closed pattern discovery—more precisely on discovering closed *sets* of attribute/value equality expressions—by introducing general methods that efficiently list meaningful closed set collections of feasible size. Three different approaches are investigated: (i) listing closed sets among all sets satisfying additional structural constraints that reflect application-dependent background knowledge, (ii) selecting closed sets possessing a higher degree or strength of closedness based on a suitable generalization of that notion, and (iii) generating closed sets at random according to some fixed and controlled distribution that reflects interestingness. The remainder of the introduction surveys the contributions of this thesis in greater detail after providing some more background on local pattern discovery.

name	type	region	date	death toll
1887 Huang He flood	river flood	Asia	before 1900	high
St. Lucia's flood	storm tide	Europe	before 1900	high
1974 Bangladesh monsoon	rain	Asia	after 1900	high
Burchardi flood	storm tide	Europe	before 1900	high
Great Dayton flood	river flood	N. America	after 1900	med
Morvi dam burst	dam break	Asia	after 1900	med
Great Sheffield flood	dam break	Europe	before 1900	med
1962 North Sea flood	storm tide	Europe	after 1900	med

Table 1.1.: Very small example database of historical flood catastrophes described by four categorical attributes. Given such a database, association rule discovery is concerned with extracting implications that are likely to hold between the attributes.

1.1. Background and Motivation

The data mining community has described a wide variety of local pattern discovery tasks and techniques. For this introduction we confine ourselves to a representative example and postpone a detailed formal introduction to Chapter 2. Namely we consider the discovery of "association rules" [Agrawal et al., 1996] from categorical data tables such as Table 1.1. Association rules are statements of the form "whenever X holds for a row (data record) it is also likely that Y holds" where X and Y are conjunctions of column/value equality expressions. For instance a rule holding for every row of Table 1.1 is

$$(\text{region} = \text{Europe}) \wedge (\text{death toll} = \text{high}) \rightarrow (\text{date} = \text{before 1900}) \ . \tag{1.1}$$

For discussing the usefulness of such a "unit of knowledge" it is necessary to take a closer look at the nature of data, which can be considered as being "collected by mapping entities in the domain of interest to symbolic representation by means of some *measurement procedure*" [Hand et al., 2001]. By discovering rules that hold in our data, we can form hypotheses about the underlying real-world domain and/or assess the quality of our measurement/data gathering procedure. This part of data analysis is called "exploratory data analysis" (EDA, see, e.g., Tukey [1980]). Furthermore, the combination of many local patterns can act as a representation of the complete database respectively of the underlying real-world domain. Building such a representation is known as descriptive respectively predictive "modeling" and has the purpose of either describing the data in a novel and insightful way or to infer missing values of future measurements.

No matter for what purpose local patterns are discovered, usually only a small fraction of all potential patterns is useful for subsequent analysis and modeling steps. There is a rich set of measures that can be used to rank the patterns according to their utility or interestingness (see Section 2.1). For instance one can rank according to the occurrence *frequency* $f(X \wedge Y)$ of data records that support the rule's antecedent as well as its consequent, i.e., the fraction of database records for which the rule is valid (e.g., 1/4 for rule (1.1)). Unfortunately, for many applications more expressive measures are required—such as a combination of antecedent frequency and deviation of local and global consequent frequency $f(X)(f(X \wedge Y)/f(X) - f(Y))$—but extracting only maximally interesting patterns with respect to many of these measures has no known efficient algorithm or is even known to be **NP**-hard (e.g., Wang et al. [2005b] or Jermaine [2005]). Therefore, pattern discovery methods usually resort to an exponential time two phase approach with an enumeration/listing step and a subsequent ranking/selection step (see Knobbe et al. [2008]). The computation time required for the listing step, naturally, is at least proportional to the number of patterns it produces. Thus, whenever the number of unfiltered patterns is too large the whole approach becomes infeasible. Due to the combinatorial explosion of the pattern space—there are 59,049 rules already for the example in Table 1.1 or even 3,486,784,401 if one also allows inequality expressions—one clearly needs algorithmically exploitable ways to reduce the number of enumerated patterns.

Two major approaches are used for that purpose: pruning based on a *minimum frequency threshold* and considering only *closed patterns* instead of all patterns. A pattern is closed if it cannot be extended in a way that preserves its frequency. For instance rule (1.1) is not closed because it can be extended to the rule

$$(\text{region} = \text{Europe}) \wedge (\text{death toll} = \text{high}) \to (\text{date} = \text{before 1900}) \wedge (\text{type} = \text{storm tide}) \quad (1.2)$$

having the same frequency of 1/4. Both concepts are indeed effective pattern reduction methods because it is possible to efficiently restrict pattern enumeration to closed patterns of high frequency (e.g., by the lcm algorithm [Uno et al., 2004], which enumerates them in an amortized polynomial time in the size of the input database). Equally important, both are semantically reasonable pruning criteria. Excluding patterns of low frequency can avoid reporting insignificant observations, and focusing on closed patterns enforces non-redundancy among the enumerated patterns. For example rule (1.1) can be considered redundant in the presence of the rule (1.2) because rule (1.1) is implied by rule (1.2).

Dataset	#rows	#attr.	#bin.	size
Election	801	56	227	0.4MB
Questionnaire	11.188	346	1893	19.7MB
Census (30k)	30.000	68	395	8.1MB

Table 1.2.: Summary of databases used in motivational experiment. Column *#attr.* gives the number of categorical attributes and *#bin.* the total number of binary attribute/value equality expressions.

While considering closed sets combined with frequency-based pruning can successfully solve a wide range of local pattern discovery tasks, this approach still is essentially an exponential time algorithm, and the number of patterns it produces can get out of hand easily. These scenarios are not mere theoretical constructs but can appear in practice already on small to mid-sized databases. As a motivational experiment consider the discovery of minimal non-redundant association rules, i.e., association rules based on closed sets, in three socio-economic databases that are summarized in Table 1.2. All datasets can be regarded as small to at most mid-sized datasets by the standards of the data mining community. Figure 1.1 shows the required computation time for several minimum frequency thresholds using JClose [Pasquier et al., 2005] as representative[1] rule discovery algorithm. Due to an exponential increase of the number of

[1]Although there are more recent and potentially faster rule discovery algorithms, all of them exhibit a similar

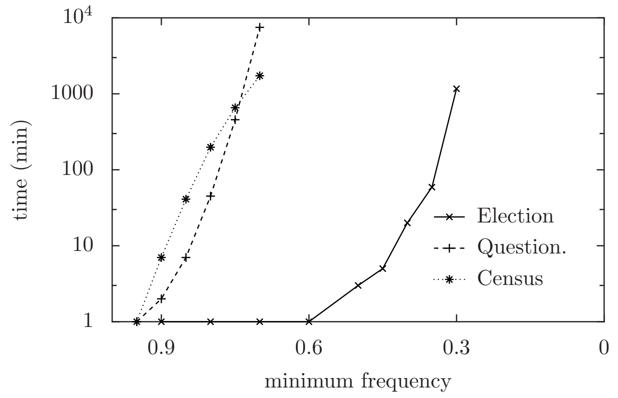


Figure 1.1.: Computation time of exhaustively listing all rules based on closed frequent sets.

closed frequent sets for decreasing thresholds one can observe an also exponentially increasing computation time. Note that this explosion occurs already very early on the frequency scale. For *Election* the method becomes problematic for thresholds lower than 0.35. The situation for the other two datasets is even worse. For *Questionnaire* the required time for a threshold as high as 0.7 is more than five days!

This demonstrates that even for moderately-sized datasets and moderately high minimum frequency thresholds, finding association rules based on all frequent closed sets can be impossible. With a frequency threshold of 0.7 or higher, however, local pattern discovery becomes in fact *global* pattern discovery which is not desired in many applications. While excluding patterns of very low frequency is useful for filtering out insignificant patterns, occurrence frequency alone is not a suitable ranking measure (see, e.g., Tatti [2008]): the most frequent patterns are usually not the most interesting but rather the most trivial. Hence, listing only very frequent sets (and subsequently applying a more expressive ranking measure) potentially misses the truly valuable patterns.

This thesis addresses the problem above by examining three complementary approaches that lead to sound algorithms with formal performance guarantees. Namely, it modifies the standard problem setting of exhaustively listing all frequent (anti-monotone constrained) closed sets (i) by allowing restrictions to the underlying pattern language, (ii) by introducing a stricter notion of closedness, and (iii) by replacing exhaustive listing with controlled random generation. More precisely, we introduce problem variants that give rise to collections of closed sets that

(R1) are defined according to well-specified selection criteria that are useful in application contexts (in particular that are *not* restricted to sets of very high frequency or depend on a black-box internal search order of some listing algorithm), and

(R2) can be listed by algorithms possessing efficient worst-case performance guarantees, i.e., they produce the collections within amortized polynomial time per closed set in the result collection (otherwise they would not be helpful towards meeting computation time limits).

qualitative behavior.

When considering algorithmic performance guarantees, it is important to view them in the context of their theoretically achievable limits. This allows to adequately assess the significance of results as well as to guide further research efforts. Therefore, beside developing useful algorithms, a complementary goal of this thesis is to explore the complexity landscape of the relevant computational problems.

1.2. Contributions

This section surveys the most important formal contributions of this thesis in detail. We start with a summary of the three major directions into which the main results can be clustered.

1. *Efficiently combining closed sets with the paradigm of constraint-based pattern discovery.* Instead of enumerating ordinary closed sets, the idea is to enumerate closed constrained sets, i.e., sets that cannot be augmented in a way that preserves their frequency *and* that satisfies additional structural constraints. This gives rise to a theoretical investigation of the relation of constrained closedness and partially defined closure operators as well as of the efficient enumerability of their fixpoints. This approach is discussed in Chapter 3.

2. *Developing a quantification of closedness that can be used for efficient pruning.* Just as frequency-based pruning uses the fact that sets can be compared in terms of their frequency, we can develop a quantification of closedness that allows us to state that one set is "more closed" than another: we call a set Δ-closed for a positive integer Δ if all its augmentations have a database support that is reduced by at least Δ. This notion can be related to a suitably defined efficiently computable closure operator. Consequently, despite not being anti-monotone, it is possible to perform efficient optimization with respect to this degree of closedness. This approach is discussed in Chapter 4.

3. *Replacing exhaustive listing of closed sets by controlled random generation.* More precisely, controlled random generation means a non-deterministic generation with respect to some probability distribution that can be chosen freely in order to reflect any application-dependent notion of interestingness. In order to realize this idea, the thesis applies Markov chain Monte Carlo sampling based on the Metropolis-Hastings algorithm. This involves the design of a stochastic process that operates on the family of closed sets and that can be simulated efficiently. This approach is discussed in Chapter 5.

In the following, the individual contributions for each of these three main directions are listed in detail. After that, some additional contributions are presented that arise in this thesis as a byproduct of the main results.

1.2.1. Closed constrained sets

A known approach for the fast generation of small pattern collections is the introduction of further constraints to the patterns that can be exploited during enumeration (see, e.g., Bonchi and Lucchese [2007]). Such constraints are usually related to background knowledge on the data attributes (see Chapter 3 for examples). As pointed out in Bonchi and Lucchese [2004], the definition of closedness has to be modified in the context of constraint-based set mining.

Let E denote the set of all attribute/value expressions of a database $\mathcal{D}$, $c : \mathcal{P}(E) \to \{0,1\}$ a constraint on subsets of E, and $\mathcal{F} = \{F \subseteq E\colon\ c(F) = 1\}$ the family of feasible sets. A closed constrained set is a set $F \in \mathcal{F}$ that has no augmentation with equal support in $\mathcal{D}$ *among the sets in* $\mathcal{F}$. This notion subsumes standard closedness with which it coincides for $\mathcal{F} = \mathcal{P}(E)$.

For the original closed set family, efficient enumerability results from the fact that for any input database there is an efficiently computable closure operator $\sigma : \mathcal{P}(E) \to \mathcal{P}(E)$ that is inducing the closed sets $\mathcal{C}$, i.e., $\mathcal{C} = \{\sigma(F)\colon\ F \subseteq E\}$. It turns out that this connection does not generally hold for closed constrained sets. Moreover, even if an inducing closure operator exists, the efficient enumerability of its fixpoints is not straightforward: while it is well known that the fixpoints of a *globally* defined closure operator can be listed with an amortized linear number of closure computations, for the closed constrained sets the fixpoints must only be computed for the *restricted domain* constituted by the feasible sets. As discussed below, efficient enumeration is not always possible in these scenarios.

This thesis gives, for the first time, a formal axiomatic characterization of constraint classes that allow to efficiently enumerate fixpoints of arbitrary closure operators ρ that are only defined on the feasible sets, i.e., $\rho\colon \mathcal{F} \to \mathcal{F}$, as well as of constraint classes that guarantee the existence of a closure operator inducing the closed constrained sets. For a set system $(E, \mathcal{F})$ with closure operator ρ let n denote the size of E and N the number of fixpoints, and let $T_\mathcal{F}$, $S_\mathcal{F}$, T_ρ, and S_ρ be the time and space complexity of evaluating the constraint inducing $\mathcal{F}$ and of computing ρ, respectively. The results are as follows.

(i) A simple divide-and-conquer algorithm correctly lists all closed sets of *strongly accessible* set systems in total time $O(Nn(T_\mathcal{F} + T_\rho + n))$, and space $O(n + S_\mathcal{F} + S_\rho)$. Strong accessibility means that every $Y \in \mathcal{F}$ can be reached from all $X \subset Y$ with $X \in \mathcal{F}$ via augmentations with single elements "inside $\mathcal{F}$". This is a strict relaxation of independence systems (set systems induced by anti-monotone constraints) as well as of greedoids [Korte and Lovász, 1985] and can be thought of as an abstract generalization of connectivity in the sense that the family of all connected vertex sets of a graph always forms a strongly accessible set system. The algorithm also provides an *algorithmic characterization* of strongly accessible set systems because it is correct for all closure operators of an input set system if and *only if* that set system is strongly accessible.

(ii) The problem becomes intractable for the class of *accessible* set systems (that are induced by weakly anti-monotone constraints). Specifically, there is a lower bound of $\Omega(2^{n/4})$ on the worst-case number of closure and constraint computations that has to be performed by any correct algorithm accessing the input only via these two operations. This bound holds even if the problem is restricted to instances with a constant number of closed sets. In particular this shows that no output polynomial time algorithm exists for that task.

(iii) The collection of closed constrained sets for all datasets is induced by a closure operator if and only if the set system induced by the constraints satisfies a certain *confluence* property, i.e., for all sets $X, Y \in \mathcal{F}$ with a non-empty intersection, $\mathcal{F}$ also contains the union $X \cup Y$. Moreover, a corresponding closure operator can be computed efficiently if its domain is strongly accessible. In conjunction with result (i) we have an $O\big(n^2\,(|\mathcal{D}| + nT_\mathcal{F})\big)$ delay and $O(n + S_\mathcal{F})$ space algorithm for listing the closed constrained sets of confluent and strongly accessible set systems $(E, \mathcal{F})$ with respect to a given dataset $\mathcal{D}$.

This constitutes a fairly general sufficiency criterion for the tractability of listing all closed constrained sets of a dataset. In contrast, if there is no corresponding closure operator, the problem turns out to be hard even for anti-monotone constraints.

1.2.2. Quantifying closedness

Sometimes there is no background knowledge based on which one can define reasonable constraints. Going back to our standard problem of exhaustive closed frequent set listing, this essentially leaves raising the frequency threshold as only way to reduce the output. However, as mentioned earlier, a particular problem with frequency-based (and also other anti-monotone) pruning is that it may exclude long, and thus particularly informative, patterns that are correspondingly of low frequency (see, e.g., Zhu et al. [2007]). In this thesis, an alternative, non-anti-monotone, pruning mechanism is introduced based on a quantification—and thus a generalization—of the notion of closedness. By quantifying closedness we mean to associate a numerical value with a set indicating “how much closed” it is. This allows to compare sets with each other based on their degree or strength of closedness and ultimately provides a second numerical selection parameter in addition to minimum frequency.

The quantification works as follows. For a positive integer Δ we call a set Δ-closed (or strongly closed) if it cannot be augmented without reducing its support by at least Δ. Thus, strongly closed sets are sets that are at the boundary of a sharp drop in frequency and generalize ordinary closed sets (which are 1-closed). Just as closed sets are used because of their *non-redundancy*, i.e., the property that no two distinct patterns have an identical support set, strongly closed sets can be used to strengthen this property: there are no two distinct Δ-closed sets with an *almost identical* support set, i.e., with a symmetric difference of less than Δ. Thus, they are realizing a *distinctiveness constraint* proposed by De Raedt and Zimmermann [2007]. Moreover, experimental results with different real-world databases indicate two desirable properties of strongly closed sets in applications: (a) they are indeed able to capture *long* patterns, even when this problem is difficult for frequency-based approaches and (b) they are *stable*, i.e., robust against noise and/or dynamic changes in the data.

Beside introducing the definition and discussing formal and semantic properties of strongly closed sets, this thesis gives the following algorithmic results.

(iv) The collection of strongly closed sets of a database always form a closure system just as the ordinary closed sets, i.e., for all input databases there is a closure operator σ_Δ such that a set is Δ-closed if and only if it is a fixpoint of σ_Δ. Let again E denote the set of all attribute/value expressions with $|E| = n$ and N the number of strongly closed sets. Building on the results for arbitrary closure operators, the operator σ_Δ can be used to design an algorithm that lists all strongly closed sets in time $O(nNT_{\sigma_\Delta})$ where T_{σ_Δ} is the time complexity of σ_Δ.

(v) Let m denote the size of the input database $\mathcal{D}$ measured by the number of 1-entries of its binary matrix representation, i.e., $m = \sum_{d \in \mathcal{D}} |\{e \in E \colon d \models e\}|$. While a trivial algorithmic implementation of σ_Δ performs a closure computation in time $O(nm)$, a more sophisticated algorithm for σ_Δ achieves linear time complexity with respect to the complement of the binary data matrix, i.e., the number of 0-entries. In particular, for non-sparse binary data matrices this means that all strongly closed sets can be listed with time $O(nm)$ per pattern. This is equal to the best known time bound for listing ordinary closed frequent sets [Uno et al., 2004]—a notable fact as the notion of Δ-closedness subsumes the latter one.

1.2.3. Closed set sampling

Finally, this thesis proposes randomized sampling as an alternative algorithmic paradigm for generating interesting collections of closed sets. This is fundamentally different to traditional exhaustive listing approaches. If a listing algorithm is prematurely terminated after some allocated computation time is exceeded then it produces a subset $\mathcal{C}' \subseteq \mathcal{C}$ of the closed sets $\mathcal{C}$, which is determined by the algorithm's internal search order. In contrast, sampling can efficiently generate closed set collections that are representative for some controlled target distribution $\pi : \mathcal{C} \to [0, 1]$. The distribution can be chosen freely according to some interestingness measure $q : \mathcal{C} \to \mathbb{R}$, e.g., $\pi(\cdot) = q(\cdot)/Z$ with a normalizing constant Z. This has the further advantage that it does not rely on a well-chosen value for a threshold parameter (like minimum support), which is often hard to determine in practice.

Sampling patterns is an emerging trend in the data mining community [Hasan and Zaki, 2009, Hasan et al., 2007, Schietgat et al., 2009]. However, all proposed algorithm are essentially simulating random walks on the family of frequent patterns. There is no straightforward way of adapting these processes to sampling *closed* sets. Using a (frequent) set sampler within a rejection sampling approach does not lead to an efficient algorithm because the number of sets can grow exponentially in the number of closed sets. Consequently, the probability of successfully drawing a closed set with rejection sampling can be very small; for a small database like "msweb" from the UCI machine learning repository [Asuncion and Newman, 2007] with 285 items and $32K$ transactions already as small as 10^{-80}. Similarly, it is not a viable option to map a drawn non-closed set to its smallest closed superset: the varying number of generators among the closed sets would introduce an uncontrollable bias.

Therefore, this thesis introduces a random walk algorithm that directly works with the closed sets. In particular,

(vi) For a given database, a Markov chain is defined that has the family of closed sets as its state space, allows an efficient computation of a single simulation step, and converges to any desired strictly positive target distribution. Since the worst-case mixing time of these chains can grow exponentially in the database size, a heuristic polynomially bounded function for the number of simulation steps is proposed. This heuristic results in sufficient closeness to the target distribution for a number of benchmark databases.

Moreover, it is shown how concept sampling can be used to build an approximation scheme for counting the number of all closed sets. This fact in conjunction with results from the computational complexity community for this counting problem (cf. contribution (viii) below) suggests that there exists no general worst-case polynomial time sampling algorithm for closed sets. Hence, using a heuristic method—as the one proposed here—seems inevitable.

1.2.4. Additional contributions

In addition to its main points, this thesis also contains two novel related results as a byproduct of the central contributions. Although these results do not directly contribute to the overall goal of this thesis, they are interesting in their own right.

The first result is a byproduct of discussing the basics of closed set listing. Closed sets are maximal, i.e., longest, representatives of their support equivalence class. In the context of supervised local pattern discovery, however, it is beneficial to instead consider shortest representatives. This gives rise to the question, whether efficient algorithms exist to extract those minimum patterns[2].

(vii) It turns out that the minimum cardinality representatives of each equivalence class can be found during the enumeration process of the standard closure system without additional cost, while finding a minimum representative of a single equivalence class is **NP**-hard.

This result is instructive, because it demonstrates that, in order to show the intractability of an enumeration problem P, it is insufficient to show that a solution to P consists of the solutions of a set of instances of an **NP**-hard optimization problem. This is caused by a changed underlying notion of efficiency between listing (amortized polynomial time) and optimization (polynomial time).

The second result considers the problem of approximately counting the number of frequent sets in a time that is polynomially bounded in the size of the input database. An algorithm achieving this could be used to either provide user guidance for setting an appropriate minimum support threshold or to tune it automatically. The investigation of sampling problems in local pattern discovery and the intrinsic connection between sampling and counting, gives rise to the following result.

(viii) Unless for all $\epsilon > 0$ and for all problems in **NP** there is a randomized algorithm (with suitably bounded error probability) that runs in time 2^{n^ϵ} for instances of size n, the following holds: there is a constant $c > 0$ such that there is no polynomial time randomized algorithm that, given a database of size n, correctly approximates the logarithm of the number of frequent sets within a multiplicative error bound of n^c.

The existence of algorithms for all problems in **NP** with arbitrary small exponential time complexity is generally considered as very unlikely. Thus, this negative results indicates that not even the order of magnitude of the number of frequent sets can be approximated reasonably well. For approximating the number of closed frequent sets the same hardness can be shown by a simple linear reduction.

1.3. Outline

The remainder of this thesis is structured as follows.

Chapter 2 starts off with a formal introduction to local pattern discovery with a strong focus on closed sets and support equivalence. In contrast to standard exposition of the topic, here, support equivalence is developed from a unified perspective of supervised as well as unsupervised local patterns. This motivates novel algorithms and results regarding the minimum, i.e., shortest, members of equivalence classes.

[2]Note that this does not refer to the problem of enumerating all *minimal* (with respect to set inclusion) representatives. For this task, i.e., generator extraction, efficient algorithms are well-known (see Calders et al. [2005]).

Chapter 3 presents the results regarding the combination of closedness with other, in particular structural, constraints. For this purpose, a theoretical analysis is performed on the efficient enumerability of fixpoints of partially defined closure operators as well as on the existence of closure operators inducing the closed constrained sets.

Chapter 4 presents strong closedness as novel interestingness measure that builds on the concept of closedness itself by quantifying its degree or strength. Along with theoretical properties and an efficient algorithm, experiments are presented that demonstrate the practical use of strong closedness.

Chapter 5 discusses a paradigm shift for local pattern discovery from exhaustive listing to controlled sampling. It introduces a Markov chain Monte Carlo method operating on the closed sets and investigates its theoretical and practical properties. In addition, the intrinsic connection between pattern sampling and approximate pattern counting is used to prove related computational complexity results.

Chapter 6 concludes with a brief high-level summary of all contributions and discusses their strengths and weaknesses. Finally, two selected directions for future research are presented.

1.4. Previously Published Work

Parts of this dissertation have been published in the following journal articles, each of which having one ore more preceding conference or workshop papers.

- M. Boley and H. Grosskreutz. Approximating the number of frequent sets in dense data. *Knowledge and Information Systems*, 21(1):65–89, Oct. 2009a
  - M. Boley and H. Grosskreutz. A randomized approach for approximating the number of frequent sets. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008)*, pages 43–52. IEEE Computer Society, 2008
  - M. Boley. Intelligent pattern mining via quick parameter evaluation. In *NSF Symposium on Next Generation of Data Mining and Cyber-Enabled Discovery for Innovation (NGDM '07)*, 2007b
- M. Boley, T. Horváth, A. Poigné, and S. Wrobel. Listing closed sets of strongly accessible set systems with applications to data mining. *Theoretical Computer Science*, 411(3):691–700, 2010b
  - M. Boley, T. Horváth, A. Poigné, and S. Wrobel. Efficient closed pattern mining in strongly accessible set systems (extended abstract). In *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2007)*, volume 4702 of *Lecture Notes in Computer Science*, pages 382–389. Springer, 2007a
  - M. Boley, T. Horváth, A. Poigne, and S. Wrobel. New results on listing closed sets of strongly accessible set systems. In *Proceedings of the 7th International Workshop on Mining and Learning with Graphs (MLG 2009)*, 2009a

  - M. Boley, T. Horváth, A. Poigné, and S. Wrobel. Efficient closed pattern mining in strongly accessible set systems. In *Proceedings of the 5th International Workshop on Mining and Learning with Graphs (MLG 2007)*, 2007b

- M. Boley, T. Horváth, and S. Wrobel. Efficient discovery of interesting patterns based on strong closedness. *Statistical Analysis and Data Mining*, 2(5–6):346–360, 2009c
  - M. Boley, T. Horváth, and S. Wrobel. Efficient discovery of interesting patterns based on strong closedness. In *Proceedings of the SIAM International Conference on Data Mining (SDM 2009)*, pages 1002–1013. SIAM, 2009b

Additional material appeared in the following conference papers.

- M. Boley. On approximating minimum infrequent and maximum frequent sets. In *Proceedings of the 10th International Discovery Science Conference (DS 2007)*, volume 4755 of *Lecture Notes in Computer Science*, pages 68–77. Springer, 2007a

- M. Boley and H. Grosskreutz. Non-redundant subgroup discovery using a closure system. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2009), Part I*, volume 5781 of *Lecture Notes in Computer Science*, pages 179–194. Springer, 2009b

- M. Boley and T. Gärtner. On the complexity of constraint-based theory extraction. In *Proceedings of the 12th International Discovery Science Conference (DS 2009)*, volume 5808 of *Lecture Notes in Computer Science*, pages 92–106. Springer, 2009

- M. Boley, H. Grosskreutz, and T. Gärtner. Formal concept sampling for counting and threshold-free local pattern mining. In *Proceedings of the SIAM International Conference on Data Mining (SDM 2010)*. SIAM, 2010a

Two of these papers received awards, namely a *distinguished paper award* of MLG 2007 (Boley et al. [2007b]) and the *best student paper award* of ICDM 2008 (Boley and Grosskreutz [2008]).

2. Efficient Local Pattern Discovery

This chapter lays the groundwork for the rest of this thesis by giving a formal introduction to the basic notions of local pattern discovery. In particular, it motivates and introduces our base problem of listing all closed frequent sets of a database as well as the corresponding notions of algorithmic efficiency. Starting from that point, the subsequent chapters introduce different efficient techniques that can be applied whenever the family of closed frequent sets is too large to be listed effectively.

The introduction given in this chapter aims to provide a formal view on the computational problems of local pattern discovery that, on the one hand, is as simple and as unified as possible, and, on the other hand, captures all aspects of subsequent technical discussions. Consequently, since the focus of this thesis is primarily on algorithmics and to a lesser degree also on computational complexity, the statistical side of local pattern discovery is not discussed here. Moreover, we do not cover techniques that have no effect on the asymptotic worst-case performance of pattern discovery algorithms.

This chapter also contains an original contribution motivated by global model construction from local patterns. Namely we give an efficient algorithm for listing all minimum, i.e., shortest, interesting generators of a dataset. Since minimum generator construction as an isolated task is hard, the existence of an efficient listing algorithm leads to an interesting observation: due to a modified underlying notion of algorithmic efficiency, it is possible to efficiently solve pattern listing problems that involve the computation of tasks that are considered intractable (with respect to standard efficiency notions).

The remainder of this chapter first develops the foundational terminology of local pattern discovery (Section 2.1) before it defines the related algorithmic concepts; in particular the output-sensitive efficiency notions (Section 2.2). It follows an introduction to closed sets and support equivalence that includes an investigation of its theoretical and practical properties (Section 2.3). Finally, the new algorithm for enumerating minimum generators is presented (Section 2.4), before the chapter concludes with a brief summary (Section 2.5).

2.1. Databases and Patterns

After clarifying some selected general terms and conventions, this section formally introduces the foundational notions of local pattern discovery, namely databases and patterns. Note that we strictly focus on formalizations that are necessary for the topics of this thesis and, thus, take on a rather limited view on pattern discovery. We refer to the data mining literature for advanced reading; in particular for the various kinds of structured patterns and the statistical motivation of several notions, which are both not covered here.

2.1.1. Basic Notions and Conventions

The reader is assumed to be familiar with standard mathematical notions and notation. Still, we briefly fix some terms and symbols that might otherwise be ambiguous and recall the notion of a "closure operator", which plays a central role in this thesis.

Throughout this work we denote "elementary objects" by non-capital letters (e.g., e), sets of elementary objects by capital letters (e.g, E), and families, i.e., sets of sets, by calligraphic letters (e.g., $\mathcal{E}$). In particular the **power set** of some set E, i.e., the family of all subsets of E, is denoted by $\mathcal{P}(E)$. The symbol "$\subset$" denotes the strict subset relation between sets, and the symbol "$\ominus$" denotes the **symmetric set difference**, i.e., $X \ominus Y = (X \setminus Y) \cup (Y \setminus X)$. For a family of sets $\mathcal{S}$ the terms **minimal** and **maximal** refer to the subset relation, i.e., a set $S \in \mathcal{S}$ is a minimal element of $\mathcal{S}$ if there no strict subset $S' \subset S$ that is also an element of $\mathcal{S}$. In contrast, the terms **minimum** and **maximum** is used with respect to the cardinality of a set, i.e., $S \in \mathcal{S}$ is called minimum element of $\mathcal{S}$ if there is no $S' \in \mathcal{S}$ with $|S'| < |S|$. For two functions $f, g \colon X \to \mathbb{R}$ we say that f is **polynomially bounded** by g, denoted by $f(\cdot) \leq \text{poly}(g(\cdot))$, if there is a polynomial p such that for all $x \in X$ it holds that $f(x) \leq p(g(x))$.

Let X be a set partially ordered by $\preceq$. A mapping $\rho \colon X \to X$ is called a **closure operator** if it satisfies for all $x, y \in X$: extensivity, i.e., $x \preceq \rho(x)$, monotonicity, i.e., $x \preceq y$ implies $\rho(x) \preceq \rho(y)$, and idempotence, i.e., $\rho(x) = \rho(\rho(x))$. In this thesis, we usually consider closure operators that are defined on families of sets with respect to the subset relation.

2.1.2. Databases

Among the formal terminology of knowledge discovery from databases and specifically local pattern discovery the most central notion is that of a "database". There are various advanced database concepts and technologies from different scientific disciplines (see Fox et al. [1994]). Throughout this dissertation, however, we content ourselves with a very simple formalization of databases, namely with binary datasets.

A binary **dataset** $\mathcal{D}$ over some finite **ground set** E is a bag (multiset) of sets, called **data records**, $D_1, \ldots, D_m$ each of which being a subset of E. Without loss of generality we always assume that the elements of the ground set are linearly ordered by indices $1, \ldots, n$, i.e., $E = \{e_1, \ldots, e_n\}$. Hence, $\max F$ and $\min F$ are well-defined expressions for all non-empty $F \subseteq E$. Moreover, we denote by E_i for $i \in \{1, \ldots, n\}$ the first i elements with respect to that order, i.e., $E_i = \{e_1, \ldots, e_i\}$.

Inspired by the application field of market basket analysis (see Russell and Petersen [2000]), data records are often referred to as "transactions" and the elements of E as "items" in the data mining literature. Yet another nomenclature is used in formal concept analysis [Ganter and Wille, 1999] where one considers finite binary relations between a set of "attributes" and a set of "objects" (in Chapter 5 we take on this view for its symmetric treatment of the ground set and data records). The binary dataset representation, albeit being simple, is sufficient to represent a huge class of possible data sources.

A categorical data table (e.g., the one given in Table 1.1) can easily be represented as a binary dataset by choosing the ground set as consisting of all attribute/value equality expressions that can be formed from the table. More precisely, a table (matrix) consisting of m data row vectors $d_1, \ldots, d_m$ with $d_i = (d_i(1), \ldots, d_i(n))$ can be represented by the dataset $\mathcal{D} = \{D_1, \ldots, D_m\}$

with $D_i = \{(j, v) : d_i(j) = v\}$ over ground set $E = \{(j, d_i(j)) : 1 \leq i \leq m, 1 \leq j \leq n\}$. As an example of this transformation consider the dataset given in Table 2.1, which is the binary representation of the table of historical flood catastrophes from the introductory chapter (Table 1.1). Moreover, using the same transformation, one can even represent numeric data granted that it can be suitably *discretized* or that one exhaustively creates items corresponding to numerical interval by techniques like *interordinal scaling* (see, e.g., Kaytoue-Uberall et al. [2009]).

	a	b	c	d	e	f	g	h	i	j	k
1	1	0	0	0	1	0	0	1	0	1	0
2	0	1	0	0	0	1	0	1	0	1	0
3	0	0	1	0	1	0	0	0	1	1	0
4	0	1	0	0	0	1	0	1	0	1	0
5	1	0	0	0	0	0	1	0	1	0	1
6	0	0	0	1	1	0	0	0	1	0	1
7	0	0	0	1	0	1	0	1	0	0	1
8	0	1	0	0	0	1	0	0	1	0	1

Table 2.1.: The binary dataset corresponding to the categorical data table given in Table 1.1. Each item corresponds to an attribute/value expression: a $\leftrightarrow$ "type=river flood", b $\leftrightarrow$ "type=storm tide", c $\leftrightarrow$ "type=rain", d $\leftrightarrow$ "type=dam break", e $\leftrightarrow$ "region=Asia", etc.

Finally, let us consider the physical representation of data. Despite the formalization as bag of sets, when *visualizing* a concrete dataset we usually switch to a **representation as binary matrix** $M_\mathcal{D}$, i.e., $M_\mathcal{D}$ is the $(|\mathcal{D}| \times |E|)$-matrix having a 1-entry in row i at column j if and only if the i-th data record contains the j-th item (for ease of reference and to avoid ambiguity, we usually use letters as column headings and number the rows/data records). However, for the purpose of *representing a dataset in a computer* we always assume lists, i.e., either data record incidence lists representing the individual data records or item incidence lists representing the support sets of the individual items. It is always assumed that individual list elements can be stored within one storage unit. Hence, we define the size of a data record or any other set as its cardinality. Correspondingly, as **size of a dataset** $\mathcal{D}$ over ground set we consider the sum of the sizes of its data records, i.e., we define

$$\text{size}(\mathcal{D}) = \sum_{D \in \mathcal{D}} |D| \quad .$$

This is usually much more economic than the corresponding binary matrix (or array) representation for which the size is $|E| \, |\mathcal{D}|$; especially for transformed categorical data tables with columns that take on many different values.

2.1.3. Patterns and Interestingness

Similarly to database notions, there are many and diverse possibilities to formalize local patterns. This holds for both: their form, i.e., the specification of objects that are potential patterns, as well as their quality, i.e., the specification of why we regard some of these objects as factual patterns, i.e., as interesting, while rejecting others. Again, for the topics of this thesis, it is sufficient to consider only very few and rather basic notions of patterns and interestingness. The reader is referred to Hand [2002] and references therein for a foundational development of the term “pattern” and to Tan et al. [2002] for a survey on interestingness measures that are used in the data mining literature.

Let $\mathcal{D}$ be a dataset over ground set E. For a specific pattern discovery task, we refer to the set of all potential local patterns as **pattern space** over $\mathcal{D}$ and denote it by $\mathcal{L}(\mathcal{D})$. The pattern space is sometimes also called “pattern language” [Gunopulos et al., 2003]. For the most part we consider only simple **(item-)sets** $F \subseteq E$ as potential patterns, i.e., we set $\mathcal{L}(\mathcal{D}) = \mathcal{P}(E)$. Sometimes, we are also interested in **(association) rules**, i.e., ordered pairs of sets $(X, Y) \in \mathcal{P}(E) \times \mathcal{P}(E)$ with $X \cap Y = \emptyset$. Such rules are denoted $X \to Y$ where X is called the rule’s **antecedent** and Y its **consequent**. We interpret sets conjunctively. That is, the *local* data portion described by a set F, called the **support (set)** of F in $\mathcal{D}$ and denoted $\mathcal{D}[F]$, is defined as the multiset of all data records from $\mathcal{D}$ that contain *all* elements of F, i.e., $\mathcal{D}[F] = \{D \in \mathcal{D} \colon D \supseteq F\}$. The support set is sometimes alternatively called “extension” or “extent” (the latter term is used in formal concept analysis where the related pattern is called “intent” correspondingly). A simple but important observation is that support sets are anti-monotone with respect to the subset relation, i.e., for $F, F' \subseteq E$ it holds that

$$F \subseteq F' \Rightarrow \mathcal{D}[F] \supseteq \mathcal{D}\big[F'\big] \quad . \tag{2.1}$$

Now we turn to specifications of interestingness, i.e., formal criteria for deciding whether a potential pattern is considered as a factual pattern. Generally the answer to this question is affirmative whenever some interestingness value associated with the pattern is high enough, i.e., we rely on being able to quantify the interestingness of a pattern with respect to the observed data. An **interestingness measure** for a pattern language $\mathcal{L}(\cdot)$ is a function

$$q \colon \{(\mathcal{D}, x) \colon \mathcal{D} \text{ a binary dataset, } x \in \mathcal{L}(\mathcal{D})\} \to \mathbb{R} \ .$$

However, often there is a fixed dataset that is clear from the context. In such cases—and if we want to simplify the notation—we just write q as an unary function $q(\cdot) = q(\mathcal{D}, \cdot)$ and omit the first argument.

The most basic measures for set patterns are the **support (count)**, i.e., the size of its support set $q_{\text{supp}}(\mathcal{D}, F) = |\mathcal{D}[F]|$ and the **frequency**, i.e., the relative size of its support with respect to the total number of data records $q_{\text{freq}}(\mathcal{D}, F) = |\mathcal{D}[F]| \,/\, |\mathcal{D}|$. A further fundamental measure is the **area function** [Geerts et al., 2004] $q_{\text{area}}(\mathcal{D}, F) = |F|\,|\mathcal{D}[F]|$. Intuitively, the area of a set corresponds to the number of 1 entries in the submatrix of $M_{\mathcal{D}}$ consisting of the columns corresponding to F and the rows corresponding to $\mathcal{D}[F]$.

All set measures can be extended to rules by applying them to the union of the rule’s antecedent and consequent. That is, the **support and the frequency of a rule** $X \xrightarrow{c} Y$ are defined as the support and the frequency of the union $X \cup Y$, respectively. A measure exclusive to rules is the **confidence**, which is defined as $q_{\text{conf}}(\mathcal{D}, X \to Y) = |\mathcal{D}[X \cup Y]| \,/\, |\mathcal{D}[X]|$. We sometimes denote the confidence of a rule above its $\to$-symbol, i.e., $X \xrightarrow{c} Y$. Rules with a confidence of 1 are called implications or **exact rules**.

All measures defined so far are unsupervised measures in the sense that they rely on no further information but the dataset itself. In contrast, there are so-called supervised descriptive rule induction techniques that rely on additional information in the form of binary[1] labels $l(D) \in \{+, -\}$ associated to each data record $D \in \mathcal{D}$. Examples for this setting are emerging pattern mining [Dong and Li, 1999] and contrast set mining [Bay and Pazzani, 2001], where one is interested in patterns having a high support difference between the positive and the negative portion of the data records, or subgroup discovery [Wrobel, 1997], where one searches for patterns with a high distributional unusualness of these labels on their support set. A representative interestingness measure used in subgroup discovery is the **binomial test quality function**

$$q_{\mathrm{bino}}(\mathcal{D}, F) = \sqrt{|\mathcal{D}[F]|} \left(\frac{|\mathcal{D}^{+}[F]|}{|\mathcal{D}[F]|} - \frac{|\mathcal{D}^{+}|}{|\mathcal{D}|} \right) \tag{2.2}$$

where $\mathcal{D}^{+}$ denotes the sub-dataset of $\mathcal{D}$ containing only the data records with positive labels, i.e., $\mathcal{D}^{+} = \{D \in \mathcal{D} \colon l(D) = +\}$.

2.2. Pattern Listing Algorithms

Having introduced basic definitions we are now ready to consider the computational side of local pattern discovery. The standard computational problems considered in this thesis are of the form: given a dataset and some threshold, list all patterns with an interestingness value no less than the threshold. For this purpose, this section reviews depth-first subset search, a very simple yet important base algorithm that correctly lists all interesting sets with respect to an anti-monotone quality measure. Motivated by this algorithm, we then introduce output-sensitive notions of algorithmic efficiency. These notions mainly abound from algorithmic combinatorics (see, e.g., Goldberg [1993], Johnson et al. [1988], Sawada [2001]). Finally, we discuss the extension of depth-first subset search to non-anti-monotone quality measures based on optimistic estimators.

2.2.1. Depth-First Subset Search

One of the most fundamental computational problems in local pattern discovery is the problem of listing all **frequent sets**, i.e., sets having a frequency with respect to a given input database $\mathcal{D}$ that exceeds a user-defined threshold t (hence, family is also referred to as "t-frequent sets"). Based on the frequent sets one can for instance compute all frequent high confidence association rules. A formal problem statement is:

Problem 2.1 (LIST-FREQUENT-SETS). *Given* a dataset $\mathcal{D}$ over ground set E, and a minimum frequency threshold $t \in [0, 1]$, *list* the family $\mathcal{R}$ of all frequent sets of $\mathcal{D}$, i.e., $\mathcal{R} = \{F \subseteq E \colon q_{\mathrm{freq}}(F) \geq t\}$. $\Diamond$

Suppose we have a linear time algorithm NEXT(F) that generates a successor of a set $F \subseteq E$ with respect to the **lexicographical order** $\prec_L$ on $\mathcal{P}(E)$ defined by $F \prec_L F' \iff \max(F \ominus F') \in F'$ and another linear time algorithm COUNT-FREQUENCY$(F, \mathcal{D})$ that computes

[1]We focus here on binary labels. In fact, arbitrary labels are considered in the supervised descriptive rule induction literature.

$q_{\text{freq}}(\mathcal{D}, F)$. Based on these algorithms—which are easy to design—we can give a straightforward algorithmic solution to Problem 2.1:

```
$F \leftarrow \emptyset$
**repeat**
   **if** COUNT-FREQUENCY$(F, \mathcal{D}) \geq t$ **then print** $F$
   $F \leftarrow$NEXT$(F)$
**until** $F = E$
```

While this algorithm is obviously correct, it has exponential time complexity and, thus, is not considered efficient with respect to the standard notion of algorithmic efficiency, which is polynomial time. That said, we can observe for problem LIST-FREQUENT-SETS that there can be no correct polynomial time algorithm at all because its input/output relation is not polynomially balanced: for each positive number n we can consider the input dataset $\mathcal{D} = \{E\}$ over $E = \{1, \ldots, n\}$ and frequency threshold 1. For these problem instances, the input is of size n while the output $\mathcal{F} = \mathcal{P}(E)$ is of size 2^n. Hence, even printing the output takes a time that can grow exponentially in the input size.

Algorithm 1 Depth-First Subset Search

```
Input  : dataset $\mathcal{D}$ over ground set $E$, anti-monotone quality measure $q : \mathcal{P}(E) \to \mathbb{R}$
         quality threshold $t \in \mathbb{R}$
Output: result family $\mathcal{R} = \{F \subseteq E \colon q(F) \geq t\}$ of all interesting sets

**main:**
1. **if** $q(\emptyset) \geq t$ **then** dfs$(\emptyset)$

**dfs**$(F)$**:**
1. **print** $F$
2. **for all** $e \in E$ with $e > \max F$ **do**
3.    **if** $q(F \cup \{e\}) \geq t$ **then** dfs$(F \cup \{e\})$
```

Although the standard notion of algorithmic efficiency, i.e., “polynomial time”, cannot be used to differentiate between any two correct algorithms for Problem 2.1, there are listing strategies that one naturally considers as superior to the lexicographic brute-force approach above. For instance consider the following idea: instead of visiting the sets in lexicographical order, visit them in a bottom-up fashion (with respect to the subset relation) by a branching out refinement process that is stopped wherever an infrequent set is encountered. Albeit not visiting all subsets of the ground set, this approach cannot miss a frequent set because of the anti-monotonicity of support sets (Equation (2.1)). Thus, it is not only correct for listing frequent sets but for listing the interesting sets with respect to any anti-monotone measure. Moreover, we can use any visiting strategy that is compatible with the subset relation. The precise pseudocode realization given in Algorithm 1 uses depth-first visiting because this has the advantage of being simple to express and space efficient. Hence we call the algorithm “depth-first subset search”.

The key insight that makes us considering Algorithm 1 as superior to the naive lexicographical approach above is the following: although Algorithm 1 has exponential time worst-case complexity, its computation time is guaranteed to be proportional to the amount of output it produces. Since the algorithm is correct, the procedure **dfs** can only be called once for each set $F \in \mathcal{R}$. Moreover, every call of **dfs** has an exclusive time complexity of $O(|E| \text{ size}(\mathcal{D}))$, i.e., a time complexity that we have not already accounted for by the time complexity of other **dfs** calls.

Proposition 2.1. *Algorithm 1 solves Problem 2.1, i.e., lists the family $\mathcal{R}$ of frequent sets, in time $O(|E| \text{ size}(\mathcal{D}) \ |\mathcal{R}|)$.*

In contrast, the naive algorithm based on a complete enumeration of $\mathcal{P}(E)$ does not have a similar guarantee—its computation time is always proportional to 2^n independent of the size of $\mathcal{R}$. Formalizing this difference gives rise to the output sensitive notions of efficiency introduced below.

2.2.2. Listing Algorithms and Efficiency Notions

All listing problems considered in this thesis are of the following form: given an instance $x \in X$ from some specific domain X specifying legal problem instances, list a finite solution family of finite sets $\mathcal{S} \in \boldsymbol{S}(x)$ where $\boldsymbol{S}(x)$ is the set of all feasible solutions to instance x (note that, in this thesis, $\boldsymbol{S}(x)$ contains only one uniquely defined solution for all problems except for Problem 2.4). That is, formally we can consider a **listing problem** as an ordered pair $P = (X, \{\boldsymbol{S}(x)\}_{x \in X})$. Generally we assume for all such problems P that they are **polynomially balanced** in the sense that $\max\{|S| : S \in \mathcal{S}, \mathcal{S} \in \boldsymbol{S}(x)\} \leq \text{poly}(\text{size}(x))$.

A **listing algorithm** $\mathfrak{A}$ for problem P is a computer program[2] that given an input x from X successively prints an output sequence of finite sets $\mathfrak{A}(x) = O_1(x), \ldots, O_{k(x)}(x)$. We say that $\mathfrak{A}$ solves P **exactly** if for all $x \in X$ there is a solution $\mathcal{S} \in \boldsymbol{S}(x)$ such that $\{O_1, \ldots, O_{k(x)}\} = \mathcal{S}$. Sometimes this is further differentiated into "soundness", i.e., $\{O_1(x), \ldots, O_{k(x)}(x)\} \subseteq \mathcal{S}$, and "completeness", i.e., $\{O_1, \ldots, O_{k(x)}\} \supseteq \mathcal{S}$). Moreover, we say that $\mathfrak{A}$ solves P **non-redundantly** if no set of the output sequence is printed more than once, and that $\mathfrak{A}$ solves P **correctly** if it does so exactly and non-redundantly.

For quantifying the performance of a listing algorithm $\mathfrak{A}$ one can use the standard notions of **time and space complexity**, which we denote by $\text{time}_{\mathfrak{A}}(\cdot)$ and $\text{spc}_{\mathfrak{A}}(\cdot)$, respectively. In addition, there is a further important complexity measure that is exclusive to listing algorithms. Let $O_1(x), \ldots, O_{k(x)}(x)$ be the output sequence $\mathfrak{A}(x)$. Then the **delay** of $\mathfrak{A}$ on input x, denoted $\text{del}_{\mathfrak{A}}(x)$, is defined by

$$\text{del}_{\mathfrak{A}}(x) = \max\{\text{del}_{\mathfrak{A}}(x, i) \colon 1 \leq i \leq k(x) + 1\}$$

where $\text{del}_{\mathfrak{A}}(x, i)$ denotes the **individual delay** of element i, i.e., the number of computational steps that $\mathfrak{A}$ performs before printing $O_1(x)$ for $i = 1$, after printing $O_{k(x)}(x)$ for $i = k(x) + 1$, and between printing two consecutive elements $O_{i-1}(x)$ and $O_i(x)$ for $i \in \{2, \ldots, k(x)\}$. That

[2] We do not specify an explicit machine model for listing algorithms here. Note, however, that if one is inclined to do so, it is advisable to choose a random access based model over tape-based models like Turing machines. Within the latter it is problematic to efficiently simulate exponential size search structures that may arise in pattern listing problems.

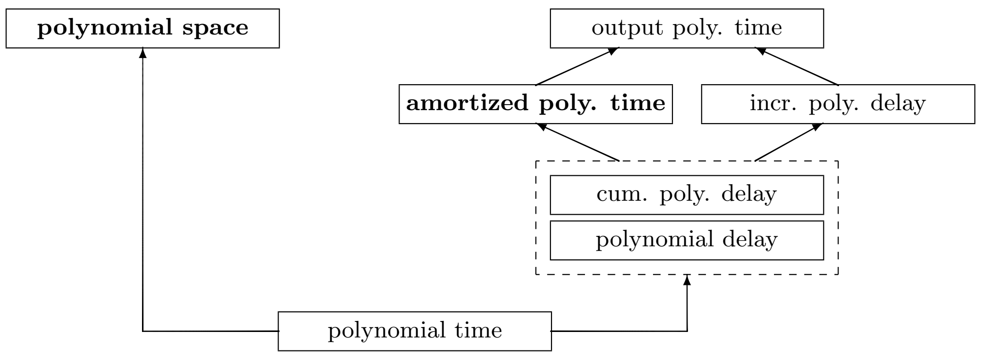


Figure 2.1.: Relations among the different notions of algorithmic efficiency. Polynomial space is not implied by any of the output-sensitive efficiency notions. Cumulative polynomial delay and polynomial delay are equivalent in the sense that polynomial delay implies cumulative polynomial delay and cumulative polynomial delay implies that there is an input/output equivalent reformulation of the algorithm having polynomial delay.

is, the delay is the maximum time that passes between the production of any two successive sets of the output sequence. An important modification of this concept due to Goldberg [1993] is to consider the maximum average delay that one can observe for the computation of any prefix of the output sequence. This quantity is called the **cumulative delay** of $\mathfrak{A}$ on input x, denoted $\mathrm{cdel}_{\mathfrak{A}}(x)$, and formally defined by

$$\mathrm{cdel}_{\mathfrak{A}}(x) = \max \left\{ \sum_{j=1}^{i} \mathrm{del}_{\mathfrak{A}}(x,j)/i \colon\ 1 \leq i \leq k(x)+1 \right\} \ .$$

Clearly, we have $\mathrm{cdel}_{\mathfrak{A}}(x) \leq \mathrm{del}_{\mathfrak{A}}(x)$ for any algorithm $\mathfrak{A}$ and input x. Goldberg [1993] observed that also the converse inequality holds in a certain sense. Namely, for all listing algorithms $\mathfrak{A}$ one can give an input/output equivalent reformulation $\mathfrak{A}'$ such that $\mathrm{del}_{\mathfrak{A}'}(x) \leq \mathrm{cdel}_{\mathfrak{A}}(x)$ for all inputs x. Intuitively, $\mathfrak{A}'$ exactly simulates $\mathfrak{A}$ except that it "holds back" some of the output for a certain time (see Section 3.3.2).

The idea of output-sensitive efficiency notions for listing algorithms is then either to give (input size dependent) performance bounds on the different forms of delay or to give performance bounds on the total time complexity that depend not only on the size of the input but also on the length of the output sequence. We say that an algorithm $\mathfrak{A}$ has **output polynomial time (complexity)** if $\mathrm{time}_{\mathfrak{A}}(x) \leq \mathrm{poly}(\mathrm{size}(x) + |\mathfrak{A}(x)|)$. A stronger notion is **incremental polynomial time** that demands output polynomial time for each prefix of the output sequence $\mathfrak{A}(x) = O_1(x), \ldots, O_{k(x)}(x)$, i.e., $\sum_{j=1}^{i} \mathrm{del}_{\mathfrak{A}}(x,j) \leq \mathrm{poly}(\mathrm{size}(x)+i)$ for all $i \in \{1, \ldots, k(x)+1\}$. While these conditions guarantee a good, i.e., polynomial, relation between computation time and input/output size, for practical applications a superlinear dependency on the output size can be infeasible. Therefore, the following notions all require a linear dependency. We say that $\mathfrak{A}$ has **amortized polynomial time** if the average computation time of $\mathfrak{A}$ per output element is polynomially bounded, i.e., $\mathrm{time}_{\mathfrak{A}}(x)/\left|\mathfrak{A}(x)\right| \leq \mathrm{poly}(\mathrm{size}(x)))$. Moreover, we say that $\mathfrak{A}$ has **cumulative polynomial delay** if $\mathrm{cdel}_{\mathfrak{A}}(x) \leq \mathrm{poly}(\mathrm{size}(x))$, and accordingly that it has **polynomial delay** if $\mathrm{del}_{\mathcal{A}}(x) \leq \mathrm{poly}(\mathrm{size}(x))$.

All efficiency notions and the implications among them are depicted in Figure 2.1. It is an important observation that none of the output-sensitive efficiency notions implies space efficiency. Thus, investigating whether an algorithm has polynomial space is an additional relevant analysis step after considering the time/delay performance. A final remark is related to the input size: Usually the problem instances contain a binary dataset and some additional numeric parameters, and the size of the input is dominated by the size of the dataset. However, sometimes we consider more abstract problems that involve oracles, i.e., subprograms, as part of their input. For such problems we either assume that the time complexity of the oracles is linear in the size of additional input or we make their time complexity an explicit parameter of all performance statements.

We can check that the introduced efficiency notions successfully capture our desired intuition of "good" and "bad" algorithms for listing problems with a potentially superpolynomial number of output elements: Proposition 2.1 guarantees that Algorithm 1 solves Problem 2.1 with amortized polynomial time (it is also straightforward to show that the algorithm has in fact also polynomial delay and space). In contrast, the naive algorithm from Section 2.2.1 does not have output polynomial time.

Note that all efficient listing algorithms discussed in this thesis are in fact polynomial delay algorithms and, conversely, that all negative results rule out even output polynomial time algorithms. Still for high-level discussion and in particular for the remainder of this chapter we confine ourselves to amortized polynomial time and polynomial space as representative efficiency notions. The reason for this is that the total time and the space usage of an algorithm are usually easiest to see and that both are sufficient given the application contexts we have in mind.

2.2.3. Subset Search for Non-Anti-Monotone Measures

The depth-first subset search approach is an efficient algorithm for listing all interesting sets of a dataset with respect to an anti-monotone interestingness measure. Unfortunately, most measures that have really proven to express meaningful notions of interestingness in applications are not anti-monotone. As an exemplary non anti-monotone interestingness measure consider the binomial test quality function (Eq. (2.2)) used in subgroup discovery. The corresponding pattern discovery problem can be defined as follows.

Problem 2.2 (SUBGROUP-DISCOVERY)**.** *Given* a binary labeled dataset $\mathcal{D}$ over ground set E, and an interestingness threshold t, *list* the family of all sets $F \subseteq E$ having a binomial quality q_{bino} for $\mathcal{D}$ of at least t. $\Diamond$

The fact that q_{bino} is not anti-monotone can be observed in the example dataset given in Table 2.1 (that corresponds to Table 1.1). If we regard the information encoded in items j and k (corresponding to column "death toll") as binary labels, i.e., $l(D) = + \iff j \in D$ (corresponding to "death toll=high"), then we have $q_{\text{bino}}(\{b, h\}) = 1/\sqrt{2}$ but $q_{\text{bino}}(\emptyset) = 0$. While $\mathcal{R} \neq \emptyset$ for threshold $t = 1/\sqrt{2}$, Algorithm 1 does not print any result set for this input. Hence, it is incorrect for Problem 2.2.

Using the following idea of Wrobel [1997], however, one can construct a modified version of depth-first subset search that correctly solves SUBGROUP-DISCOVERY: instead of directly enumerating the result family $\mathcal{R}$, enumerate an anti-monotone hull of $\mathcal{R}$ that is defined by an optimistic estimator of the interestingness measures. An **optimistic estimator** for a measure $q \colon \mathcal{P}(E) \to \mathbb{R}$ is a function $\hat{q} \colon \mathcal{P}(E) \to \mathbb{R}$ that upper bounds the quality of a set and *all its supersets*, i.e., it holds for all $F \subseteq F' \subseteq E$ that $\hat{q}(F) \geq q(F')$. For instance, a simple yet effective optimistic estimator for the binomial test quality function given by Wrobel [1997] is

$$\hat{q}_{\text{bino}}(F) = \sqrt{|\mathcal{D}[F]|}\left(1 - \left|\mathcal{D}^+\right| / |\mathcal{D}|\right) \quad . \tag{2.3}$$

As this function is anti-monotone (in fact, it is order-equivalent to the frequency measure), Algorithm 1 can be used to enumerate the space $\mathcal{S} = \{F \subseteq E \colon \hat{q}_{\text{bino}}(F) \geq t\}$ and only list the elements of $\mathcal{R} = \{F \subseteq E \colon q_{\text{bino}}(F) \geq t\}$ which is guaranteed to be a subfamily of $\mathcal{S}$ due to the optimistic estimator property. Listing only elements of $\mathcal{R}$ can be achieved by simply changing line 1 of **dfs** appropriately.

Note that, the anti-monotonicity of the involved optimistic estimator is in fact not required for the correctness of this approach: the optimistic estimator property guarantees that $\mathcal{R}$ already lies in some anti-monotone subfamily of $\mathcal{S}$. Hence, it follows that, for all interestingness measures q with an optimistic estimator $\hat{q}$, the depth-first subset search algorithm can be modified to list the family of interesting sets $\mathcal{R} = \{F \subseteq E \colon q(F) \geq t\}$ in time $O(|E| \text{ size}(\mathcal{D}) \, |\mathcal{S}|)$ where $\mathcal{S} = \{F \subseteq E \colon \hat{q}(F) \geq t\}$ is the **enumeration space** defined by $\hat{q}$—assuming that the evaluation of q and $\hat{q}$ can both be done in time $O(\text{size}(\mathcal{D}))$. In particular for Problem 2.2 we can conclude:

Proposition 2.2. *Algorithm 1 can be modified to solve Problem 2.2 in time $O(|E| \text{ size}(\mathcal{D}) \, |\mathcal{S}|)$ where $\mathcal{S} = \{F \subseteq E \colon \hat{q}_{\text{bino}}(F) \geq t\}$ is the enumeration space of all potentially interesting sets.*

This is not an efficient algorithm for Problem 2.2 because the size of the enumeration space induced by $\hat{q}_{\text{bino}}$ can grow exponentially in the size of the result family induced by q_{bino}. As, on the other hand, $\mathcal{S}$ is usually much smaller than $\mathcal{P}(E)$, it is much better than, e.g., the naive lexicographical approach from Section 2.2.1. Moreover, for many local pattern listing problems including Problem 2.2, no output polynomial time algorithm is known, and for some (e.g., for emerging pattern discovery [Wang et al., 2005b]) one can even show that none exists (unless **P** = **NP**). Thus, the modified subset search approach based on optimistic estimators is a practically very relevant algorithm (see also Grosskreutz et al. [2008] for further refinements of the idea).

2.3. Closed Sets and Non-Redundancy

In this section, we introduce the central notions of support equivalence [Bastide et al., 2000b] and closed sets [Pasquier et al., 1999]. Closed sets are the unique maximal elements of their respective support equivalence class. They are also known as maximal tiles [Geerts et al., 2004] respectively as the intents of formal concepts [Ganter and Wille, 1999]. We review three fundamental facts that underline their importance.

1. Many pattern discovery tasks can be solved equally well or better if one only lists a non-redundant result family in the sense that no two sets of the result have a common support equivalence class.

2. The compression ratio of interesting sets compared to a (non-redundant) set of representatives of their support equivalence classes can grow exponentially in theory and also be huge in practice.

3. The depth-first subset search algorithm can be modified to list a set of representatives, namely the closed sets, of all interesting equivalence classes with respect to an anti-monotone interestingness measure with amortized polynomial time and no space overhead.

The modified depth-first subset search algorithm is the seminal lcm algorithm [Uno et al., 2004]. Due to points 2 and 3 above, lcm can substantially outperform approaches that enumerate all interesting sets exhaustively instead of only considering a set of representatives.

2.3.1. Support Equivalence, Closed Sets, and Generators

Let $\mathcal{D}$ be a dataset over ground set E. Two sets $F, F' \subseteq E$ are **support equivalent** (with respect to $\mathcal{D}$), denoted by $F \equiv F'$, if they have an identical support in $\mathcal{D}$, i.e., $\mathcal{D}[F] = \mathcal{D}[F']$. Clearly, support equivalence is an equivalence relation on $\mathcal{P}(E)$. For an $F \subseteq E$ we denote by $[F]$ its **support equivalence class**, i.e., the family $[F] = \{F' \in \mathcal{P}(E) : F \equiv F'\}$. In this context F is called a **representative** of $[F]$. For instance for the example dataset of Table 2.1 we have

$$[\{b,f,j,h\}] = \{\{b,f,j,h\},\{b,f,j\},\{b,f,h\},\{b,j,h\},\{f,j,h\},\{b,f\},\{j,h\}\} \tag{2.4}$$

all of which are sharing the support set $\{2,4\}$. For a family $\mathcal{F} \subseteq \mathcal{P}(E)$ we denote by $\mathcal{F}^{\equiv}$ the equivalence classes it contains, i.e., $\mathcal{F}^{\equiv} = \{[F] : F \in \mathcal{F}\}$. In the following definition we introduce an important special term that is used for the maximal elements of each equivalence class.

Definition 2.1 (Closed Set)**.** Let $\mathcal{D}$ be a dataset over ground set E. A set $F \subseteq E$ is called **closed** if it is a maximal element of its support equivalence class $[F]$, i.e., if for all $F' \supset F$ it holds that $\mathcal{D}[F'] \subset \mathcal{D}[F]$. $\Diamond$

By $\mathcal{C}(\mathcal{D})$ we denote the family of all closed sets of a dataset $\mathcal{D}$. In cases where the dataset is clear from the context we sometimes only use $\mathcal{C}$ instead. The closed sets/maximal members of their support equivalence class, exhibit two important properties: they are unique and can be efficiently computed as the intersection of all data records of the corresponding support set. Moreover, the thus defined operation satisfies the closure operator properties. In summary we have the following result.

Proposition 2.3 (Pasquier et al. [1999])**.** *Let $\mathcal{D}$ be a dataset over ground set E and define the map $\sigma \colon \mathcal{P}(E) \to \mathcal{P}(E)$ as the operation given by*

$$\sigma(F) = \{e \in E \colon \forall D \in \mathcal{D}[F]\,,\, e \in D\} = \bigcap \mathcal{D}[F] \quad .$$

Then the following statements hold.

(i) For all sets $F \subseteq E$ there is a unique maximal set in $[F]$, which is given by $\sigma(F)$.

(ii) The operation σ is a closure operator on $\mathcal{P}(E)$ with respect to the subset relation.

We call the mapping σ the **(standard) support closure operator** with respect to $\mathcal{D}$. Due to property (i) of Proposition 2.3, the family of closed sets $\mathcal{C}$ corresponding to the fixpoints of σ is a set of representatives of $\mathcal{P}(E)^{\equiv}$, i.e., it contains exactly one class representative for each support equivalence class. Moreover, for any set $F \subseteq E$ the corresponding class representative $C \in \mathcal{C}$ with $C \equiv F$ is given by $\sigma(F)$. In this sense F can be used to generate C using σ, and, hence, we call F a **generator** of C. The minimal members of an equivalence class, i.e., the **minimal generators**, are sometimes also referred to as "key patterns" [Bastide et al., 2000b] or "free sets" [Boulicaut et al., 2003]. The latter term is derived from their property of not containing certain rules (see Section 4.5.2).

In contrast to the closed sets, there can be more than one minimal generator per equivalence class. This can be observed in the exemplary equivalence class given in Equation (2.4) where $\{b, h\}$ and $\{f, j\}$ both are minimal in $[\{b, f, h, j\}]$. The family of all minimal generators satisfies another important property, though, namely anti-monotonicity.

Proposition 2.4 (Boulicaut et al. [2003])**.** *Let $\mathcal{D}$ be a dataset over ground set E. The property of being a minimal generator is anti-monotone. That is, for all $F \subset F' \subseteq E$ it holds that if F' is a minimal member of $[F']$ then F is a minimal member of $[F]$.*

Thus, all minimal generators of a dataset can be listed efficiently; for instance using Algorithm 1.

2.3.2. Representativity and Non-Redundancy

After this formal introduction we can now discuss the value of closed sets and other sets of representatives of the support equivalence relation. The key insight here is that many interestingness measures are either a function of a pattern's support set (e.g., frequency, binomial test quality, or the optimistic estimator of binomial test quality) or a compound function of several support sets (e.g., confidence), hence, in these cases all members of an equivalence class induce patterns with the same quality value. As many sets can have an identical support set on the given data, listing all of them, e.g., with depth-first subset listing may lead to many redundant evaluations of the quality function and to a result family that contains multiple descriptions of the same subportion of the data.

By listing only one set per equivalence class such redundancies can not occur. The closed sets in particular are a set of representatives that fully represents the support sets of all other sets without further information. That is, the support set in a dataset $\mathcal{D}$ over E of an arbitrary set $F \subseteq E$ can be deduced from the family of closed sets $\mathcal{C}(\mathcal{D})$ annotated by their corresponding support sets without further information from the dataset: the support set of F is equal to the support set of the unique smallest superset of F in $\mathcal{C}$. For this feature, the family of closed frequent sets is often referred to as a "condensed representation" of the family of frequent sets. Together with the minimal generators, the closed sets even represent all **minimal non-redundant association rules**, i.e., rules that are defined as follows: a rule $X \overset{c}{\to} Y$ is called minimal non-redundant if there is no rule $X' \overset{c}{\to} Y'$ with $X' \subseteq X$, $Y' \supseteq Y$ and the same support. In Bastide et al. [2000a] minimal non-redundant association rules are related to closed sets and generators. For simplicity, here we only quote the result on exact rules.

dataset	credi.	lung-.	lymph	mush.	nurse.	sick	splice	soybe.	tic-t.	vote
$\lvert E\rvert$	58	159	50	117	27	66	133	287	27	48
$\lvert \mathcal{D}\rvert$	1000	32	148	8124	12960	3772	3190	638	958	435
label	bad	1	maln.	pois.	recm.	sick	EI	bspot.	pos.	repub.
sgd	0.628	0.708	0.757	0.890	0.813	0.908	0.987	0.792	0.999	0.972
min-repr	**0.633**	**0.731**	**0.831**	**0.946**	0.813	**0.928**	0.987	**0.856**	0.999	0.972

Table 2.2.: Base statistics and label attribute of evaluation datasets and results of prediction experiments: average AUC using 5-fold cross-validation for models based on 20 sets of greatest binomial test quality (sgd) compared to models based on minimum generators of 20 support equivalence classes of greatest binomial test quality.

Proposition 2.5 (Bastide et al. [2000a]). *All exact minimal non-redundant rules $X \to Y$ of a dataset $\mathcal{D}$ are of the form $X \cup Y \in \mathcal{C}$ with X being a minimal generator of $X \cup Y$.*

The value of a condensed representation fully representing some larger family of patterns is obvious: whenever the complete family is intractably large we can just consider the equally informative condensed family. However, even if we do not care about completely representing some larger pattern family, the non-redundancy provided by considering only one representative per equivalence class is useful in its own right. In order to support this claim, we review experimental results of using local patterns for global model construction—a central task of supervised machine learning. This is an important application branch of local pattern discovery known as Lego [Knobbe et al., 2008]. The basic idea is that pattern families discovered from labeled datasets over some ground set E can be converted to global models $h\colon \mathcal{P}(E) \to \{+,-\}$ for predicting the label of previously unseen data records $D \subseteq E$, which are assumed to be generated by the same process as the individual data records of the input dataset.

For interesting pattern collections $\mathcal{R}$ discovered based on the binomial quality function there is a simple but effective way of interpreting them as a set of models (see Lavrac et al. [2004]). Let $\mathcal{R} = \{F_1, \dots, F_k\}$ be ordered in decreasing order with respect to the binomial test quality. Then we can for each $i \in \{1, \dots, k\}$ consider the prediction model h_i defined by

$$h_i(D) = \begin{cases} +, & \text{if } D \supseteq F_j \text{ for at least one } j \in \{1,\dots,i\} \\ -, & \text{otherwise} \end{cases} .$$

That is, h_i classifies a given data point $D \subseteq E$ as positive if it supports any of the i highest quality sets from $\mathcal{R}$. The rational behind this choice is that if D was added to the dataset it would belong to at least one group of data records exhibiting an unusually high fraction of positive labels. We can evaluate the complete model set $\{h_1, \dots, h_k\}$ at once by considering its **area under the ROC curve (AUC)**, i.e., the curve given by the convex hull of the points (x_i, y_i) corresponding to the false positive rate and the true positive rate of model h_i, respectively. That is, we have $x_i = \lvert\{D \in \mathcal{D}^- \colon h_i(D) = +\}\rvert \,/\, \lvert\mathcal{D}^-\rvert$ and $y_i = \lvert\{D \in \mathcal{D}^+ \colon h_i(D) = +\}\rvert \,/\, \lvert\mathcal{D}^+\rvert$ (see also Hastie et al. [2009, pp. 277-278]).

Using the approaches above for model construction and evaluation, we now consider the following experimental setting. For each of 10 real-world datasets from the UCI machine learning repository [Asuncion and Newman, 2007] we investigate the predictive performance of

- a family $\mathcal{R}$ of 20 sets of greatest binomial test quality and
- a family $\mathcal{M}$ of *minimum* representatives of the 20 support equivalence classes of greatest binomial test quality.

We are interested in the minimum representatives as they lead to small models, which are easier to interpret and tend less to overfitting than large models. The predictive performances are measured by the average AUC achieved over a 5-fold cross-validation[3] by the model sets h_i described above. Table 2.2 summarizes the ten datasets and gives the corresponding AUC results. Even though the non-redundant families $\mathcal{M}$ do not necessarily fully represent all the information of the other families $\mathcal{R}$, the prediction models based on $\mathcal{M}$ perform equally good or better than the corresponding models based on $\mathcal{R}$. Intuitively, the reason for this behavior is that redundant sets do not contribute any new information to the model in the sense that data records D generated by the same process as $\mathcal{D}$ are likely to equally support equivalent sets (for $|\mathcal{D}|$ large enough).

2.3.3. Compression

Having seen the value of non-redundant pattern families, we now investigate the potential reduction of output as well as of computation time that can be achieved by considering class representatives instead of individual sets. For that purpose we reconsider Problem 2.2 of listing interesting subgroups with respect to the binomial quality function. Recall that, for this problem, the modified depth-first subset search lists all sets in $\mathcal{R} = \{F \subseteq E \colon q_{\text{bino}}(F) \geq t\}$ by systematically enumerating the family $\mathcal{S} = \{F \subseteq E \colon \hat{q}_{\text{bino}}(F) \geq t\}$.

	e_1	e_2	e_3	e_4	e_5	e_6	l
1	0	0	1	1	1	1	$-$
2	1	1	0	0	1	1	$-$
3	1	1	1	1	0	0	$-$
4	1	1	1	1	1	1	$+$

Table 2.3.: Dataset $\mathcal{D}_6$ emerging from Construction 2.1.

By definition all families of subsets can contain no more equivalence classes than sets, i.e., it trivially holds that $|\mathcal{R}^{\equiv}| \leq |\mathcal{R}|$ and $|\mathcal{S}^{\equiv}| \leq |\mathcal{S}|$. In order to investigate the theoretical potential of enumeration space and output reduction, we now give a general dataset construction that intuitively reflects worst-case situations for solving Problem 2.2 via the depth-first subset search algorithm.

[3]That is, the datasets are divided randomly into five folds of equal size, each of which is used as validation data during one round where the remaining data is used for pattern discovery.

Construction 2.1. For even positive integers $n \in \mathbb{N}$ let the dataset $\mathcal{D}_n$ over ground set $E = \{e_1, \ldots, e_n\}$ with binary labels l be defined by $\mathcal{D}_n = \{D_1, \ldots, D_{n/2+1}\}$ such that $D_i = (E \setminus \{e_{2i-1}, e_{2i}\})$ with $l(D_i) = -$ for $i = 1, \ldots, n/2$, and $D_{n/2+1} = E$ with $l(D_{n/2+1}) = +$. $\Diamond$

Table 2.3 illustrates this construction for $n = 6$ annotated with binary labels. For threshold $t = 3/4$ only one equivalence class is interesting with respect to the binomial test quality function; namely the one describing the support set $\{D_4\}$, which is done, e.g., by the representative set $\{e_1, e_3, e_5\}$. In total, however, the class consists of all $3^3 = 27$ sets that contain either one or both items of each of the pairs $\{e_1, e_2\}$, $\{e_3, e_4\}$, and $\{e_5, e_6\}$. Generally, for $\mathcal{D}_n$ and threshold $t = n/(n+2)$ one can check that the only interesting equivalence class is

$$[E] = \{F \subseteq E \colon (F \cap \{e_1, e_2\}) \neq \emptyset, \ldots, (F \cap \{e_{n-1}, e_n\}) \neq \emptyset\}$$

containing $3^{n/2}$ members, which is equal to the compression rate $|\mathcal{R}| / |\mathcal{R}^{\equiv}|$ as $\mathcal{R}$ contains only one element. For the enumeration space we have the situation that $\mathcal{S} = \mathcal{P}(E)$. On the other hand, only sets F can be closed that contain each pair $\{e_{2i-1}, e_{2i}\}$ for $i = 1, \ldots, n/2$ either fully or not at all. Hence, $|\mathcal{S}^{\equiv}| = |\mathcal{C}| = 2^{n/2}$, and the compression rate $|\mathcal{S}| / |\mathcal{S}^{\equiv}| = 2^{n/2}$. Altogether, the datasets $\{\mathcal{D}_n : n \in \mathbb{N}\}$ witness that the compression rate achieved by considering equivalence classes instead of individual sets can grow exponentially in the number of items (and data records).

Theorem 2.6. *For all positive integers $n \in \mathbb{N}$ there is a dataset $\mathcal{D}$ of cardinality $n/2 + 1$ over a ground set E of size n and a quality threshold t such that the compression rates $|\mathcal{R}| / |\mathcal{R}^{\equiv}|$ and $|\mathcal{S}| / |\mathcal{S}^{\equiv}|$ are in $\Omega(\exp(n/2))$.*

Now we again consider the ten real-world datasets summarized in Table 2.2, in order to investigate the extend of compression that can be achieved in practice. Table 2.4 shows sizes of result families $\mathcal{R}$, compressed result families $\mathcal{R}^{\equiv}$, enumeration spaces $\mathcal{S}$, and compressed enumeration spaces $\mathcal{S}^{\equiv}$ for two quality thresholds $t = \epsilon$ and $t = t100$ per dataset. The threshold $t100$ varies among the datasets: it is equal to the quality of the 100th highest binomial test quality among all sets (note that because of ties in the quality $|\mathcal{R}_{t100}|$ can still be greater than 100). The threshold ϵ is equal to the smallest positive number distinguishable from zero in double precision. The results for $t = t100$ give a differentiated impression: ranging from tremendous compression rates of 749.000 (soybean) to no compression (e.g., vote). For decreasing thresholds, however, a significant compression arises for all datasets: while (with one exception) it is tractable to search through all equivalence classes with a potentially positive quality ($t = \epsilon$), this is infeasible on most datasets for exhaustive enumeration due to the large number of equivalent sets.

2.3.4. The LCM Algorithm for Efficient Closed Set Listing

In Section 2.3.2 we discussed that the family of closed interesting sets often serves our purposes equally well or even better than the family of all interesting sets. Moreover, Theorem 2.6 states that an algorithm that efficiently enumerates only closed interesting sets can potentially outperform an algorithm that enumerates all interesting sets by a margin that grows exponentially in the input size. This section presents the efficient closed set enumeration algorithm of Uno et al. [2004] called "lcm". It can be regarded as an extension of the depth-first subset search algorithm and solves the following problem for anti-monotone interestingness measures.

	credi.	lung-.	lymph	mush.	nurse.	sick	soybe.	splice	tic-t.	vote
$\lvert\mathcal{R}_{t100}\rvert$	100	38K	124	168	100	128	749K	100	113	101
$\lvert\mathcal{R}^{\equiv}_{t100}\rvert$	83	1	17	12	100	1	1	99	113	101
$\lvert\mathcal{S}_{t100}\rvert$	148K	456M	12K	3458	103K	>100M	>100M	398K	6067	3505
$\lvert\mathcal{S}^{\equiv}_{t100}\rvert$	87K	159K	4176	890	69K	8M	1M	395K	5824	3465
$\lvert\mathcal{R}_{\epsilon}\rvert$	6119K	>100M	1078K	>100M	11K	>100M	>100M	>100M	65K	3610K
$\lvert\mathcal{R}^{\equiv}_{\epsilon}\rvert$	175K	103K	19K	105K	11K	2M	2M	>100M	23K	82K
$\lvert\mathcal{S}_{\epsilon}\rvert$	17M	>100M	26M	>100M	192K	>100M	>100M	>100M	129K	11M
$\lvert\mathcal{S}^{\equiv}_{\epsilon}\rvert$	385K	183K	45K	228K	115K	9M	3M	>100M	43K	227K

Table 2.4.: Uncompressed and compressed result families and enumeration spaces of depth-first subset search for the binomial test quality and two thresholds.

Problem 2.3 (LIST-INTERESTING-CLOSED-SETS). *Given* a dataset $\mathcal{D}$ over ground set E, an interestingness measure q, and a threshold $t \in \mathbb{R}$, *list* the family $\mathcal{R}$ of all interesting closed sets of $\mathcal{D}$, i.e., $\mathcal{R} = \{C \in \mathcal{C} \colon q(\mathcal{D}, C) \geq t\}$. $\Diamond$

For the purpose of efficiently enumerating all closed sets of a binary dataset, one can use the fact that the closed sets are induced (see Prop. 2.3) by a closure operator σ, which can be efficiently computed in time $O(\text{size}(\mathcal{D}))$. By the closure properties of σ, it follows that every closed set $C \in \mathcal{C}$ with $C \neq \sigma(\emptyset)$ must have a generator that consists of a closed subset of C augmented by a single element, i.e., there must be a closed set $C' \subset C$ with $\sigma(C' \cup \{e\}) = C$ for some $e \in E$. Hence, an adapted version of depth-first subset search that iterates over all possible augmentation elements $e \in E$ (replacing line 2 of Alg. 1) and considers the closure $\sigma(F \cup \{e\})$ instead of $F \cup \{e\}$ would completely list all closed sets of a dataset (and only closed sets). A problem of this approach is, however, that it may redundantly list the same closed set more than once. As straightforward solution to this is to store all printed closed sets in a table and check for each generated closed set whether it has already been visited. One can implement the table such that a look-up is possible in time $O(|E|)$. Hence, the resulting listing algorithm has polynomial delay. However, as the table grows proportional to the size of the closed set family, this modification of subset search has an exponential space overhead.

A more efficient, alternative solution is enabled by the following theorem of Uno et al. [2004]. It states that each closed set other than the closure of the empty set has a unique generator that preserves its prefix up to a certain critical index (recall that by E_i we denote the first i elements of the ground set $\{e_1, \ldots, e_i\}$ with respect to some fixed underlying order).

Theorem 2.7 (Uno et al. [2004]). *Let $\mathcal{D}$ be a dataset over ground set E. For all closed sets $C' \in \mathcal{C}$ with $C' \neq \sigma(\emptyset)$ there is a unique closed set $C \in \mathcal{C}(\mathcal{D})$ with $C \subset C'$ such that there is an index $i \in \{1, \ldots, n\}$ satisfying $\sigma(C' \cap E_i) = C'$, $C' = \sigma(C \cup \{e_i\})$, and $C' \cap E_i = C \cap E_i$.*

The three conditions of the theorem allow an efficient redundancy check that does not rely on storing previously listed closed sets and, hence, uses only linear space as desired. The corresponding algorithm is called “linear time closed pattern miner (lcm)” because it has a total time complexity linear in the number of closed sets $|\mathcal{C}|$ and polynomial in the input size, i.e., $O(\text{size}(\mathcal{D})\ |E|)$. In our nomenclature, this corresponds to an amortized polynomial time algorithm (it fact it has also polynomial delay). Furthermore, just as the basic subset search algorithm, lcm can be combined with any anti-monotone pruning conditions. A summarizing pseudocode is given in Algorithm 2. We can conclude:

Proposition 2.8. *For anti-monotone interestingness measures q, Algorithm 2 solves Problem 2.3, i.e., lists the family $\mathcal{R}$ of interesting closed sets, in time $O(|E| \operatorname{size}(\mathcal{D}) |\mathcal{R}|)$.*

Algorithm 2 LCM Closed Set Listing

```
Input  : dataset $\mathcal{D}$ over ground set $E$, anti-monotone quality measure $q\colon \mathcal{P}(E) \to \mathbb{R}$
         quality threshold $t \in \mathbb{R}$
Output: family of interesting closed sets $\{C \in \mathcal{C}(\mathcal{D})\colon q(C) \geq t\}$

main:
  1. if $q(\sigma(\emptyset)) \geq t$ then dfs($\sigma(\emptyset)$)

dfs($C$):
  1. print $C$
  2. for all $e_i \in E \setminus C$ with $\sigma(C \cap E_i) = C$ do
  3.     $C' \leftarrow \sigma(C \cup \{e_i\})$
  4.     if $q(C') \geq t$ and $(C \cap E_i) = (C' \cap E_i)$ then dfs($C'$)
```

2.4. Minimum Generators

With the lcm algorithm we have an efficient solution to the closed set listing problem. However, sometimes we are interested in other sets of representatives. For instance, in Section 2.3.2 we used *minimum generators* of the most interesting equivalence classes for building a global prediction model. It is unclear whether the positive result for closed set listing transfers to finding such a family of representatives. Therefore, in this section we directly investigate the following modification of SUBGROUP-DISCOVERY .

Problem 2.4 (NON-REDUNDANT-SUBGROUP-DISCOVERY)**.** *Given* a dataset $\mathcal{D}$ over E and a quality threshold t, *list* a result family $\mathcal{R} \subseteq \mathcal{P}(E)$ of interesting sets with respect to the binomial test quality that satisfies,

(i) representativity, i.e., for all sets $F \subseteq E$ with $q_{\text{bino}}(F) \geq t$ there is an $F' \in \mathcal{R}$ that is support-equivalent to F,

(ii) non-redundancy, i.e., for all pairs of distinct sets $F, F' \in \mathcal{R}$ the support set of F and F' in $\mathcal{D}$ are distinct, and

(iii) *representative minimality*, i.e., for all listed sets $F \in \mathcal{R}$ there is no smaller set F' having the same support set, i.e., for all $F' \in [F]$ it holds that $|F'| \geq |F|$. $\Diamond$

Note that in this problem definition q_{bino} is used for the sake of concreteness. It could be replaced by any other interestingness measure that is a function of a sets support set. Below we describe two alternative algorithmic approaches to Problem 2.4: one based on closed set enumeration and one based on breadth-first traversal of all equivalence classes directly via the minimum generators.

2.4.1. Reduction to Closed Set Listing

Having an efficient algorithm for closed set enumeration, the following 2-component approach to Problem 2.4 suggests itself: (i) enumerate all closed sets C with $\hat{q}_{\text{bino}}(C) \geq t$ (where $\hat{q}_{\text{bino}}$ again denotes an optimistic estimator of q_{bino}; see Eq. (2.3)), and (ii) compute and print a minimum generator of C in case $q_{\text{bino}}(C) \geq t$. While this strategy obviously lists exactly and non-redundantly one minimum generator per interesting equivalence class, it involves solving the following computational task repeatedly.

Problem 2.5 (MINIMUM-GENERATOR)**.** *Given* a dataset $\mathcal{D}$ over E and set $F \subseteq E$, *compute* a support equivalent minimum generator of F. $\Diamond$

As we show below, this problem is equivalent to the well-known **NP**-hard problem of computing a minimum set cover, which can be defined as follows.

Problem 2.6 (MIN-SET-COVER)**.** *Given* a family of subsets $\mathcal{S} \subseteq \mathcal{P}(U)$ over some finite universe U with $\bigcup \mathcal{S} = U$, *compute* a minimum subfamily $\mathcal{S}' \subseteq \mathcal{S}$ with $\bigcup \mathcal{F}' = U$. $\Diamond$

Algorithms and complexity of the set cover problem are well investigated. Namely, we know due to Slavík [1997] that an approximate solution with a logarithmic accuracy guarantee can be computed in time $(|U|\,|\mathcal{S}|)$ by a simple greedy algorithm and, moreover, due to Feige [1998] that this guarantee is optimal up to low order terms.

Before we state the consequences of these results formally, we verify the equivalence of the two problems. An instance of the set cover problem can be transformed to a binary dataset as follows: introduce an item for all sets in $\mathcal{S}$ and a data record for all elements in U. Moreover, let $\mathcal{D}$ model the negated element/set incidence relation of the set cover instance. That is, apply the following construction.

Construction 2.2. Let $(U, \mathcal{S})$ be a set cover instance with universe $U = \{u_1, \ldots, u_m\}$ and set family $\mathcal{S} = \{S_1, \ldots, S_n\}$. Define the dataset $\mathcal{D}_{(U,\mathcal{S})} = \{D_1, \ldots, D_m\}$ over $E_{(U,\mathcal{S})} = \{e_1, \ldots, e_n\}$ by setting D_i to be the set of all elements corresponding to sets in $\mathcal{S}$ *not* covering universe element u_i, i.e., $D_i = \{e_j \in E \colon u_i \notin S_j\}$ for $i = \{1, \ldots, m\}$. $\Diamond$

With this construction we have that if a data record $D \in \mathcal{D}_{(U,\mathcal{S})}$ is in the support set of a set $F \subseteq E_{(U,\mathcal{S})}$ then all the sets of the set cover instance corresponding to F do *not* cover the universe element corresponding to D. It follows that $\mathcal{D}_{(U,\mathcal{S})}[E] = \{\}$, because $(U, \mathcal{S})$ is a valid set cover instance (having at least one solution), and, moreover, that $F \equiv E$ if and only if $\mathcal{S}_F = \{S_i \in \mathcal{S} : \ e_i \in F\}$ is a cover of U. Consequently, a minimum generator of E corresponds to a minimum cover of U. On the other hand, by applying the inverse Construction 2.2 to the dataset $\mathcal{D} \setminus \mathcal{D}[F]$ restricted to the items $\sigma(F)$, we get a set cover instance $(U, \mathcal{S})$ such that a minimum set cover of U corresponds to a minimum generator of

$\sigma(F)$. Finally, note that our transformation preserves solution sizes, i.e., $|F| = |\mathcal{S}_F|$, and, hence, positive as well as negative results regarding approximation guarantees for MIN-SET-COVER also apply to MINIMUM-GENERATOR.

Algorithm 3 Greedy Minimum Generator Approximation

```
Input  : dataset $\mathcal{D}$ over $E$, set $F \subseteq E$
Output : approximation $G \subseteq E$ to a minimum generator of $\sigma(F)$

1. set $\bar{\mathcal{D}} \leftarrow \mathcal{D} \setminus \mathcal{D}[F]$, $G \leftarrow \emptyset$
2. while $\bar{\mathcal{D}}[G] \neq \emptyset$ set $G \leftarrow G \cup \{\arg\min_{e \in \sigma(F)} |\bar{\mathcal{D}}[G \cup \{e\}]|\}$
3. return $G$
```

The greedy algorithm for MIN-SET-COVER [Johnson, 1973] iteratively selects a set covering a maximum number of universe elements that have not been covered during previous iterations. Slavík [1997] found that this approach has a multiplicative approximation guarantee of $g(m) = \ln m - \ln \ln m + 0.78$ for instances with $|U| = m$. We refer to the number $g(m)$ as **greedy approximation factor** for universes with m elements. Algorithm 3 gives a version of the greedy algorithm that directly incorporates the inverse of Construction 2.2, hence it approximates a minimum generator within $g(|\mathcal{D} \setminus \mathcal{D}[H]|)$. Combining this result with the inapproximability result of Feige [1998], we have a quite precise statement on the computational complexity of MINIMUM-GENERATOR .

Proposition 2.9. *Let a set $F \subseteq E$ and a dataset $\mathcal{D}$ over E be a problem instance of* MINIMUM-GENERATOR *and $m = |\mathcal{D} \setminus \mathcal{D}[F]|$ be the number of data records not in the support set of F. Then the following statements hold.*

- *(a) Algorithm 3 approximates within the greedy approximation factor $g(m)$ a minimum generator of $\sigma(F)$ in time $O(|\sigma(F)|\, m)$, i.e., it computes a set $G' \equiv F$ such that $|G'| \leq |G|\, g(m)$ where G is a minimum generator of $\sigma(F)$.*
- *(b) Unless for all problems in **NP** there is an $n^{O(\log \log n)}$-time algorithm for instances of size n, for all $\epsilon > 0$ there is no polynomial time algorithm computing an approximation $G' \in [F]$ that satisfies $|G'| \leq |G|\,(1-\epsilon) \ln m$.*

Thus, we can combine the greedy algorithm with the lcm algorithm in order to solve NON-REDUNDANT-SUBGROUP-DISCOVERY approximately. In particular we have to use $q = \hat{q}_{\text{bino}}$ as parameter and to replace line one of **dfs** by

1.' **if** $q_{\text{bino}}(C) \geq t$ **then print** GREEDY MINIMUM GENERATOR(C) .

The time complexity of the greedy algorithm is asymptotically dominated by the cost of computing the next closed set. Hence, we end up with the following performance statement.

Theorem 2.10. *Problem 2.4 can be solved in time $O(|\mathcal{S}^{\equiv}|\,|E|\,\text{size}(\mathcal{D}))$ and linear space if the representative minimality condition is relaxed to: for all listed sets $F \in \mathcal{R}$ there is no description F' having the same support set with $|F| > g(|\mathcal{D} \setminus \mathcal{D}[F]|)\,|F'|$.*

	credi.	lung-.	lymph	mush.	nurse.	sick	soybe.	splice	tic-t.	vote
avg. min.	5.59	4.91	4.53	5.78	5.33	?	?	?	5.01	5.99
avg. apx.	5.6	5.03	4.56	5.86	5.33	9.06	6.59	?	5.04	6.04
max. dif.	5 - 3	7 - 4	5 - 3	6 - 4	no diff	?	?	?	6 - 4	8 - 5
time frac.	0.9	0.07	0.54	0.99	0.97	0.96	0.78	?	0.93	0.59

Table 2.5.: Greedy performance for threshold $t = \epsilon$ (see Tab. 2.2): average length of minimum generators of all interesting equiv. classes (avg. min.), average length of approximations (avg. apx.), maximum difference (max. diff), and relative computation time consumed by greedy algorithm (time frac.); "?"-entries indicate a corresponding computation runs out of resources (see Tab. 2.6).

The theoretical approximation guarantee of the greedy algorithm, although almost optimal with regard to Proposition 2.9, may appear weak for practical use. However, the worst-case bound is rarely attained in practice, and the greedy result is usually close to optimum even for instances that are considered hard (see, e.g., Feo and Resende [1989] for practical experiments on the greedy performance for set cover instances resulting from Steiner triple systems[4]). Empirical evidence for this statement in the specific context of our problem NON-REDUNDANT-SUBGROUP-DISCOVERY is presented in Table 2.5, were solutions of Problem 2.2 computed by the lcm/greedy approach are compared to optimal solutions. For all datasets the average cardinality of the sets obtained using the greedy algorithm is only marginally greater than the average cardinality of the corresponding minimum generators. The maximum difference per dataset shows that the greedy algorithm approximates tighter than a factor of 2 on these examples. In addition to the approximation performance, the table also shows the relative amount of the total computation time that was consumed by the greedy algorithm. The asymptotic worst-case computation time of the lcm algorithm is not altered by the greedy approximations. However, for low thresholds and correspondingly many greedy calls, we see that the practical computation time can be dominated by them.

2.4.2. Breadth-First Minimum Generator Construction

In this section we present an alternative algorithmic approach to Problem 2.2 that directly (and efficiently) enumerates minimum generators of all potentially interesting equivalence classes. Besides producing exact solutions this approach has the advantage of avoiding potentially expensive greedy computations. The existence of such an algorithm may perhaps be surprising in the light of the **NP**-hardness of minimum generator construction. However, this hardness result refers to the construction of a *single* minimum generator and has consequences only for corresponding algorithms with (input) polynomial time complexity; namely their existence implying $\mathbf{P} = \mathbf{NP}$. In contrast, here we are interested in constructing *all* minimum generators of a dataset and consider a notion of efficiency that accepts superpolynomial algorithms. In

[4]A Steiner triple system of a finite ground set X is a family $\mathcal{B}$ of 3-subsets of X such that for all 2-subsets X' of X it holds that there is exactly one $B \in \mathcal{B}$ with $B \supset X'$. These set systems, albeit being very sparse, are known to result in practically difficult set cover problems.

particular, we can make use of the fact that larger minimum generators can be constructed inductively from shorter ones and, thus, computation time invested to compute short minimum generators can be re-used for subsequent computations of longer ones.

Intuitively, the approach is based on a breadth-first traversal of the directed graph containing as vertices all potentially interesting equivalence classes and edges $([F],[F'])$ between two classes $[F]$ and $[F']$ for each item $e \in E$ with $(F \cup \{e\}) \in [F']$. The existence of such an item is independent of the chosen representatives F, F' (for the closed sets as representatives the graph is also used in subsequent chapters where it is referred to as **generator graph**—see Sections 3.6 and 5.3). It turns out that minimum representatives of an equivalence class $[F]$ correspond to shortest paths from $[\emptyset]$ to $[F]$ in this generator graph.

However, an equivalence class can have a number of minimum generators that grows exponentially in the cardinality of its closed set. This statement is witnessed again by the datasets $\mathcal{D}_n$ over $E = \{e_1, \dots, e_n\}$ from Construction 2.1: all sets $F \subseteq E$ satisfying

$$(|F \cap \{e_1, e_2\}| = 1) \wedge \ldots \wedge (|F \cap \{e_{n-1}, e_n\}| = 1)$$

are a minimum set with support set $\{D_{n/2+1}\}$ and there are $2^{n/2}$ such sets. In addition each of these minimum generators G can be reached via $|G|!$ different shortest paths in the generator graph—one for each ordering of G. Hence, simply enumerating all sets corresponding to shortest paths would not yield an efficient algorithm for Problem 2.2.

A solution to these problems is to specify a single minimum generator to be enumerated per equivalence class such that this generator can be reached via *decreasing paths* in the generator graph, i.e., paths consisting of edges that correspond to decreasing item indices (recall that every edge $([F],[F'])$ in the generator graph corresponds to an item $e \in E$ with $(F \cup \{e\}) \in [F']$). This can be done by a **modified lexicographical order** on $\mathcal{P}(E)$, denoted by "$\prec_M$", that is defined by

$$F \prec_M F' \Leftrightarrow |F| < |F'| \vee (|F| = |F'| \wedge \max(F \ominus F') \in F') \enspace .$$

That is, $\prec_M$ orders the sets primarily by cardinality and secondarily by the lexicographical order $\prec_L$. Using this strict linear order we specify a unique minimum generator per class to be enumerated as follows.

Definition 2.2 (Canonical Minimum Generator)**.** Let $\mathcal{D}$ be a dataset over E and $F \subseteq E$. The **canonical minimum generator** of an equivalence class $[F]$, denoted by $\mu(F)$, is the unique minimum representative of $[F]$ that is minimal with respect to $\prec_M$.

While there can clearly be only one canonical minimum generator per equivalence class, they retain the important property that they can be built from one another. More precisely, for a set $F \subseteq E$ and $i \in \{1, \dots, n\}$ let us call $F \setminus E_i$ a "suffix" of F. Then every suffix of a canonical minimum generator is again a canonical minimum generator (hence, they correspond to a "convertible" constraint; see Sec. 3.1). In particular, the following lemma holds.

Lemma 2.11. *Let $G \neq \emptyset$ be a non-empty canonical minimum generator of its equivalence class $[G]$, i.e., $G = \mu(G)$. Then $G' = G \setminus \{\min G\}$ is the canonical minimum generator of $[G']$.*

Proof. Assume there is a $G'' \in [G']$ with $G'' \prec_M G'$. Then $G'' \cup \{\min G\} \prec_M G' \cup \{\min G\} = G$. But as

$$\begin{aligned}\mathcal{D}\big[G'' \cup \{\min G\}\big] &= \mathcal{D}\big[G''\big] \cap \mathcal{D}[\{\min G\}] \\ &= \mathcal{D}\big[G'\big] \cap \mathcal{D}[\{\min G\}] \\ &= \mathcal{D}\big[G' \cup \{\min G\}\big] = \mathcal{D}[G] \quad ,\end{aligned}$$

i.e., $(G'' \cup \{\min G\}) \in [G]$, this contradicts $G = \mu(G)$. □

Algorithm 4 Breadth-First Minimum Generator Listing

```
Input  : dataset $\mathcal{D}$ over $E$, interestingness measure $q$ with optimistic estimator $\hat{q}$,
         interestingness threshold $t$,
Output: family $\{\mu(F)\colon F \subseteq E,\ q(\mu(F)) \geq t\}$ in modified lexicographical order

1.  init $\mathcal{Q}$ as empty queue and $\mathcal{V}$ as empty prefix tree
2.  enqueue $(\emptyset, \sigma(\emptyset), E)$ on $\mathcal{Q}$
3.  while $\mathcal{Q} \neq \emptyset$ do
4.    dequeue front element $(G, C, A)$ of $\mathcal{Q}$
5.    if $q(C) \geq t$ then print $G$
6.    $A' \leftarrow \{e \in A \setminus C\colon e < \min G,\ \hat{q}(G \cup \{e\}) \geq t\}$
7.    for all $e \in A'$ in ascending order do
8.      $G' \leftarrow G \cup \{e\}$
9.      $C' \leftarrow \sigma(G')$
10.     if $C' \notin \mathcal{V}$ then
11.       add $C'$ to $\mathcal{V}$
12.       enqueue $(G', C', A')$ on $\mathcal{Q}$
```

Hence, a breadth-first traversal of the generator graph using only paths of decreasing order can be used to efficiently enumerate all canonical minimum generators that lie within an anti-monotone enumeration space. Algorithm 4 realizes this strategy by storing representatives of the visited equivalence classes in a queue (FIFO). The following theorem states its correctness and efficiency (with respect to the enumeration space size). A proof can be found in Appendix A.1.

Theorem 2.12. *Algorithm 4 correctly solves Problem 2.4 in time* $O(|\mathcal{S}^{\equiv}|\,|E|\,\mathrm{size}(\mathcal{D}))$ *and space* $O(|\mathcal{S}^{\equiv}|\,|E|)$ *where* $\mathcal{S}^{\equiv} = \{[F] \in \mathcal{P}(E)^{\equiv} : \hat{q}_{\mathrm{bino}}(F) \geq t\}$ *is the enumeration space of all potentially interesting equivalence classes. In particular, the algorithm exactly and non-redundantly lists* $\mu(F)$ *for all* $[F] \in \mathcal{R}^{\equiv} = \{[F] \in \mathcal{P}(E)^{\equiv} : q_{\mathrm{bino}}(F) \geq t\}$ *in modified lexicographical order.*

We close this section with a comparison of the practical performances[5] of the presented algorithms when applied to Problem 2.4. Table 2.6 contains the computation times that

[5]All experiments were performed on a Core 2 Duo E8400 @ 3Ghz running a Sun SE 6u10 Java Virtual Machine with 1.5 GB of Java heap space under Windows XP.

correspond to the compression experiments from Section 2.3 for depth-first subset search (dfs), lcm/greedy (lcm/gr), and breadth-first minimum generator listing (Algorithm 4, bfmin). The threshold $t100$, i.e., the quality of the 100th best subgroup description, is explicitly stated. The results essentially reflect the already observed enumeration space reduction. A further

dataset	credi.	lung-.	lymph	mush.	nurse.	sick	soybe.	splice	tic-t.	vote
t	0.094	0.336	0.244	0.267	0.029	0.177	0.223	0.190	0.061	0.306
dfs	**2**	84.4m	0.5	**0.6**	**1.2**	4.3h	10.6h	23	0.3	0.4
lcm/gr	3.2	**23**	0.3	1.0	2.3	**18.3m**	123	38	**0.2**	**0.2**
bfmin	3.6	**23**	**0.2**	0.9	2.4	oom	**115**	**20**	**0.2**	0.3
t					ϵ					
dfs	242	>12h	457	>12h	**2**	>12h	>12h	>12h	**1**	127
lcm/gr	184	95	6.5	53m	85.5	**7h**	**41m**	>12h	15	59
bfmin	**26**	**60**	**4**	**39**	5	oom	oom	oom	2	**19**

Table 2.6.: Computation times (in seconds unless stated differently) for quality thresholds t equal to the 100-th highest quality value (upper half) respectively equal to the smallest positive number distinguishable from zero in double precision (lower half); "oom" and ">12h" indicate computations that run out of memory or out of time, respectively.

observation for the more challenging configurations is that unless Algorithm 4 runs out of memory (the oom entries) it outperforms lcm/greedy.

2.5. Summary and Discussion

This chapter formally introduced all aspects of local pattern discovery that are needed in this thesis. Beside the fundamental notions of databases, patterns, and listing algorithms, the most central concepts of this introduction were that of support equivalence and closedness. In particular, we have seen that families of representatives of the support equivalence relation—such as the closed sets—can provide a condensed basis for non-redundant collections of interesting local patterns, and that they can be enumerated efficiently by, e.g., the lcm algorithm.

In the context of this statement, the underlying notion of algorithmic efficiency is particularly noteworthy because it deviates from the notion usually considered in computer science. Namely, we regard a pattern listing algorithm as efficient if it has an amortized polynomial time complexity per listed pattern and uses only polynomial space. Switching to output-sensitive efficiency notions is motivated by the observation that the number of closed sets of a dataset is not polynomially bounded in the input size, and, therefore, the standard efficiency notion (polynomially bounded time complexity) is not suitable for differentiating between any two closed set listing algorithms. There are two important theoretical phenomena resulting from considering output-sensitive efficiency notions: Firstly, polynomial space complexity is not implied even by the strictest of these notions (polynomial delay). Moreover, the standard rationale of problem complexity analysis, i.e., polynomial time reductions, cannot be directly used to infer the intractability of an enumeration problem and, if used improperly, may even yield misleading results: On the one hand, it is possible to efficiently enumerate all solutions of a set of instances of an **NP**-hard problem—as we have seen for the minimum generators of all equivalence classes in an anti-monotone enumeration space. One the other hand, computing their solutions individually is regarded intractable. The reason for this discrepancy is that for listing one per se accepts input exponential time complexities by considering output sensitive efficiency.

While the criteria "amortized polynomial time" and "polynomial space" are theoretical conditions with respect to worst-case situations, the presented experimental results demonstrate that both are also relevant in practice: Depth-first subset search (considered as non output polynomial time algorithm for listing all potentially interesting equivalence classes) is outperformed by the two amortized polynomial time algorithms for most configurations. Moreover, the non polynomial space breadth-first subset search with table lookup runs out of memory for several inputs for which the polynomial space lcm algorithm is feasible. However, albeit the existence of theoretically efficient algorithms, the presented experiments also show (in addition to the experiments in Chapter 1) that there are practical limits for closed set enumeration combined with anti-monotone pruning. There simply can be so many non-redundant patterns respectively potentially interesting equivalence classes that even a listing time proportional to their number is infeasible. In such cases, it is inevitable to focus on smaller collections of non-redundant patterns, and it is essential that these collections retain both: controlled semantics reflecting some notion of interestingness as well as efficient enumerability. This is the starting point from which advanced pattern discovery approaches are developed in subsequent chapters.

3. Closed Sets with Structural Constraints

This chapter investigates the theoretical background of combining closed set listing with the paradigm of constraint-based pattern discovery [Srikant et al., 1997], i.e., the idea of filtering discovered patterns for *feasible patterns* based on certain user-specified constraints. This paradigm has the advantage that constraints can be selected specifically for the application at hand and that they can build on all types of background knowledge—in particular structural information on the ground set, which is the case we focus on in this chapter. An application relying on structural information is given in the following example.

Example 3.1. Let $G = (V, \mathcal{E})$ be an undirected graph and $\mathcal{D}$ a dataset over ground set V. Such datasets can for instance model movements of individuals in a street network and occur in *track mining* applications (see, e.g., Nanni et al. [2008]). If a user is interested in traffic hot spots of variable extend, she could ask for all sets of street segments that are frequent in $\mathcal{D}$ and satisfy the additional constraint of being *connected* in G. $\Diamond$

There are several research papers on grouping constraints into classes and discussing these classes with respect to algorithmic efficiency (e.g., Bonchi and Lucchese [2007], Bucila et al. [2003], Pei and Han [2000]). However, they do not consider the interplay of constraints with closed pattern discovery. The combination of these two ideas is proposed by Bonchi and Lucchese [2004] in the following manner: instead of straightforwardly listing all sets that are feasible and have no extension of equal support, they propose to list all sets that are feasible and have no extension of equal support *among the feasible sets*. With this definition of *closed constrained sets* one removes redundancies from the feasible sets without loosing representativity. However, it is unclear whether the closed constrained sets according to this notion can be enumerated efficiently.

For the original closed set family, efficient enumerability results from the fact that for any input database there is an efficiently computable closure operator that is inducing the closed sets, i.e., the closed set family is exactly the family of fixpoints of that operator. Perhaps surprisingly, it turns out that this connection does not generally hold for closed constrained sets. Moreover, even if an inducing closure operator exists, the efficient enumerability of its fixpoints is not straightforward because such an operator may only be partially defined on the power set of the ground set. While it is well known that the fixpoints of a globally defined closure operator can be listed with an amortized linear number of closure computations, the situation is unclear for partial closure operators.

Hence, in this chapter, after giving basic facts and definitions (Section 3.1), we show that the problem of enumerating all fixpoints of a partially defined closure operator cannot be solved efficiently for domains induced by weakly anti-monotone constraints (Section 3.2). In contrast, we show that a simple divide-and-conquer algorithm efficiently and correctly enumerates all fixpoints if and only if the domain is strongly accessible (Section 3.3). Following up, we relate this result back to the problem of enumerating constrained closed sets by characterizing constraint classes (respectively the domains they describe) that have an inducing closure operator (Section 3.4). Finally, we give some exemplary applications of our main results (Section 3.5). These applications are purely theoretical in nature aiming to show the existence of efficient algorithms for certain constrained closed set enumeration problems.

3.1. Closed Constrained Set Discovery

In this section, we briefly recap constrained-based pattern discovery and emphasize the connection between constraints and set systems. Moreover, we formalize closed constrained sets and relate them to the problem of enumerating fixpoints of a partially defined closure operator.

3.1.1. Constraints and Set Systems

Generally, a **constraint** is a function $c\colon \mathcal{P}(E) \to \{0,1\}$ where E denotes some finite ground set as usual. Just as interestingness measures, constraints can depend on an input dataset and possibly additional parameters. The interestingness conditions from Chapter 2 are a simple example of constraints. For instance the frequency measure straightforwardly induces a minimum frequency constraint defined by

$$c_{\mathrm{freq}(\mathcal{D},t)}(F) = \begin{cases} 1, & \text{if } q_{\mathrm{freq}}(\mathcal{D},F) \geq t \\ 0, & \text{otherwise} \end{cases} .$$

In this chapter we consider a wider range of constraints that may depend on additional and more complex parameters. For instance the setting of Example 3.1 involves the following connectivity constraint that is parameterized by a graph $G = (V, \mathcal{E})$ on the ground set of the input dataset:

$$c_{\mathrm{conn}(G)}(F) = \begin{cases} 1, & \text{if } G[F] \text{ is connected} \\ 0, & \text{otherwise} \end{cases} ,$$

where $G[F]$ denotes the subgraph of G induced by a vertex set $F \subseteq V$. Often we are not interested in the precise definitions of constraints but rather consider whole constraint classes that are defined based on certain algorithmically exploitable properties. Similar to interestingness measures a constraint $c\colon \mathcal{P}(E) \to \{0,1\}$ is called **anti-monotone** if $X \subseteq Y$ implies $c(X) \geq c(Y)$ for all $X, Y \subseteq E$ and **monotone** if conversely $X \supseteq Y$ implies $c(X) \geq c(Y)$ for all $X, Y \subseteq E$. Moreover, c is called **weakly anti-monotone** if for all $F \subseteq E$ with $c(F) = 1$ and $F \neq \emptyset$ it holds that there is an $e \in F$ such that $c(F \setminus \{e\}) = 1$. Such constraints are sometimes also referred to as "loosely anti-monotone". Finally, a constraint c is called **convertible** if one can choose an ordering of the ground set $E = \{e_1, \ldots, e_n\}$ such that c is anti-monotone with respect to set prefixes based on that order, i.e., if $c(\emptyset) = 1$ and for $F \subseteq E$ with $c(F) = 1$ it holds that $c(F \cap \{e_1, \ldots, e_i\}) = 1$ for all $i \in \{1, \ldots, n\}$. For further reading on constraints and constraint classes see Bonchi and Lucchese [2007].

In the remainder of this chapter, we investigate mappings that have their domain defined by a constraint. Therefore, it is convenient for us to switch from the intensional perspective of constraints to the extensional perspective of the set systems they define. Namely, a (finite)

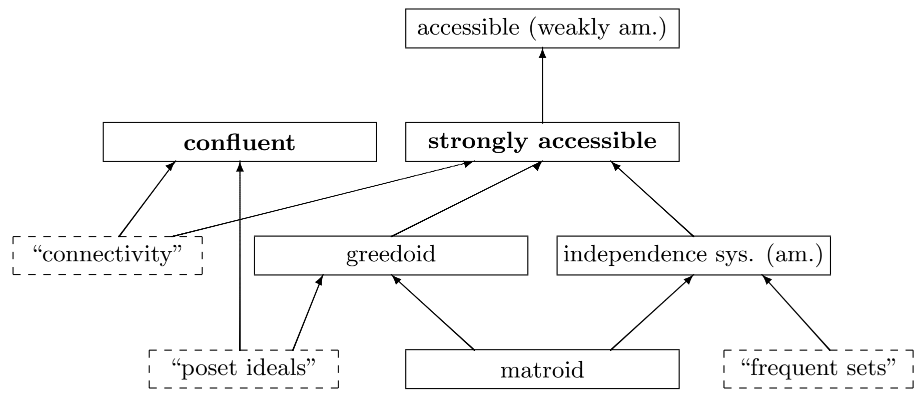


Figure 3.1.: Relations among the introduced classes of set systems (solid boxes) and application examples (dashed boxes). Accessible set systems correspond to weakly anti-monotone constraints, independence systems correspond to anti-monotone constraints.

set system is a pair $(E, \mathcal{F})$ with a finite ground set E and $\mathcal{F} \subseteq \mathcal{P}(E)$ and a constraint $c \colon \mathcal{P}(E) \to \{0,1\}$ is an implicit description of $\mathcal{F} = \{F \subseteq E \colon c(F) = 1\}$. That is, we regard c as **membership oracle** of $\mathcal{F}$. Finite set systems are also investigated in combinatorial optimization (see, e.g., Korte and Lovász [1985], Oxley [2006]). Correspondingly, we refer to the elements of $\mathcal{F}$ as **feasible sets**. In this thesis we consider only non-empty set systems. Particularly, we assume $\emptyset \in \mathcal{F}$, i.e., we consider constraints that are always satisfied by the empty set.

The theoretical results of this chapter are given with respect to structural classes of the involved set systems, some of which correspond to the constraint classes introduced above. Some well-known classes are defined as follows. A (non-empty) set system $(E, \mathcal{F})$ is called **accessible** if for all $X \in \mathcal{F} \setminus \{\emptyset\}$ there is an $e \in X$ such that $X \setminus \{e\} \in \mathcal{F}$ (such set systems correspond to **weakly anti-monotone** constraints), an **independence system** if $Y \in \mathcal{F}$ and $X \subseteq Y$ together imply $X \in \mathcal{F}$ (corresponding to **anti-monotone** constraints), a **greedoid** if it is accessible and satisfies the augmentation property, i.e., for all $X, Y \in \mathcal{F}$ with $|X| < |Y|$, there is an element $e \in Y \setminus X$ such that $X \cup \{e\} \in \mathcal{F}$, and a **matroid** if it is a greedoid and an independence system.

A central contribution of this chapter is the definition of two new classes of set systems that stand in close relation to the efficient enumerability of closed sets in combination with constraints. The first class characterizes set systems for which divide-and-conquer fixpoint enumeration as introduced in Section 3.3 is correct.

Definition 3.1 (Strongly Accessible)**.** A set system $(E, \mathcal{F})$ is called **strongly accessible** if it is accessible and if for all $X, Y \in \mathcal{F}$ with $X \subset Y$, there is an element $e \in (Y \setminus X)$ such that $(X \cup \{e\}) \in \mathcal{F}$. ◊

Strong accessibility is a combinatorial generalization of connectivity in the sense that the family of connected vertex sets of a graph always form a strongly accessible set system. Moreover, by definition, strongly accessible set systems are also accessible, and it is easy to see that strong accessibility generalizes independence systems as well as greedoids.

The second class of set systems we introduce does not stand in any containment relation with those given previously. It characterizes constraints for which listing closed sets is equivalent to enumerating the fixpoints of some inducing closure operator.

Definition 3.2 (Confluent)**.** A set system $(E, \mathcal{F})$ is called **confluent** if for all $X, Y \in \mathcal{F}$ with $(X \cap Y) \neq \emptyset$ it holds that $(X \cup Y) \in \mathcal{F}$. ◊

The relations among all introduced set system classes are illustrated in Figure 3.1 along with the application problems considered in Section 3.5.

3.1.2. Closed Constrained Sets

	a	b	c	d
1	1	1	0	0
2	1	0	1	1
3	1	1	0	1
4	0	0	1	1

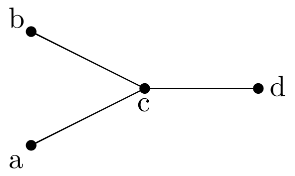


Figure 3.2.: Dataset $\mathcal{D}$ over ground set $V = \{a, b, c, d\}$ (left-hand side) and graph $G = (V, \mathcal{E})$ (right-hand side). The closed sets of $\mathcal{D}$ are $\mathcal{C} = \{\emptyset, a, d, ab, ad, cd, acd, abd, V\}$. Let $\mathcal{F}_{\text{conn}} \subseteq \mathcal{P}(V)$ denote the vertex sets inducing a connected subgraph of G. Then we have $\mathcal{C} \cap \mathcal{F}_{\text{conn}} = \{\emptyset, a, d, cd, acd, V\}$ whereas $\mathcal{CC}(\mathcal{F}_{\text{conn}}, \mathcal{D}) = \{\emptyset, a, b, d, cd, acd, V\}$.

For integrating constraints into closed set discovery, a straightforward combination is to consider only sets satisfying both: being feasible with respect to all constraints and being closed with respect to the input dataset. With this approach, however, one potentially does not *represent* all support-equivalence classes within the feasible sets, i.e., there can be a feasible set $F \in \mathcal{F}$ such that no member of $[F]$ is included in $\mathcal{F} \cap \mathcal{C}$ where $\mathcal{C}$ denotes the closed sets. This can be observed in the example given in Figure 3.2 that corresponds to the scenario described in Example 3.1. Here, the feasible set $\{b\}$ with $[\{b\}] = \{\{b\}, \{a, b\}\}$ has no representative in $\mathcal{C} \cap \mathcal{F}$ because the closed set $\{a, b\}$ does not induce a connected subgraph (see also Bonchi and Lucchese [2004, Example 1]). For this reason, we consider the following modified definition of closedness, which does not suffer from this weakness.

Definition 3.3 (Closed Constrained)**.** Let $(E, \mathcal{F})$ be a set system and $\mathcal{D}$ a dataset over E. A set $X \in \mathcal{F}$ is called **closed constrained** if for all $X \subset Y \in \mathcal{F}$ it holds that $\mathcal{D}[X] \supset \mathcal{D}[Y]$. By $\mathcal{CC}(\mathcal{F}, \mathcal{D})$ we denote the family of all closed constrained sets in $\mathcal{F}$ with respect to $\mathcal{D}$. ◊

With this definition there is a representative $R \in [F]$ in $\mathcal{CC}(\mathcal{F}, \mathcal{D})$ for all $F \in \mathcal{F}$—for instance in Figure 3.2 the set $\{b\}$ is represented by itself in $\mathcal{CC}(\mathcal{F}, \mathcal{D})$ (while it is not represented in $\mathcal{F} \cap \mathcal{C}$). Note that the family $\mathcal{CC}(\mathcal{F}, \mathcal{D})$ extended by the corresponding support counts, i.e., the set $\{(C, |\mathcal{D}[C]|) : C \in \mathcal{CC}(\mathcal{F}, \mathcal{D})\}$ precisely is what is referred to as "closure of the frequency theory of $\mathcal{D}$ and the constraints defining $\mathcal{F}$" in Bonchi and Lucchese [2004]. Here, for simplicity we only focus on the sets itself without their support counts (the frequency information can usually be added by an algorithm enumerating $\mathcal{CC}(\mathcal{F}, \mathcal{D})$ without asymptotic time overhead).

Analogously to Problem 2.3 for ordinary closed sets, we define the problem of listing all closed constrained sets. We call a dataset $\mathcal{D}$ over E **non-redundant** if no element of E is contained in all data records, i.e., if for all $e \in E$ there is a $D \in \mathcal{D}$ with $e \notin D$. Then the problem definition is as follows.

Problem 3.1 (LIST-CLOSED-CONSTRAINED)**.** *Given* a set system $(E, \mathcal{F})$ by means of a membership oracle and a non-redundant dataset $\mathcal{D}$ over E, *list* the family of closed constrained sets $\mathcal{CC}(\mathcal{F}, \mathcal{D})$. ◊

We include the requirement of the database being non-redundant in the problem definition because in case of $\emptyset \in \mathcal{F}$ it holds that a dataset $\mathcal{D}$ is non-redundant if and only if $\emptyset \in \mathcal{CC}(\mathcal{F}, \mathcal{D})$—a fact that we exploit below. Though considering only non-redundant datasets is a minor restriction, it makes our results applicable to more problems of practical interest (e.g., for the setting of Example 3.1; see Problem 3.5 in Section 3.5).

Perhaps surprisingly, we can show that Problem 3.1 is intractable even for anti-monotone constraints. The following result assumes that the constraint/membership oracle can be evaluated in linear time.

Theorem 3.1. *There is no algorithm solving* LIST-CLOSED-CONSTRAINED *restricted to independence systems in output polynomial time (unless* ***P=NP****).*

Proof. Let $(E, \mathcal{F})$ be an independence system. For $\mathcal{D} = \{\emptyset, E\}$ the problem of listing $\mathcal{CC}(\mathcal{F}, \mathcal{D})$ is equivalent to listing the *bases* of $(E, \mathcal{F})$, i.e., the maximal elements of $\mathcal{F}$. For the latter problem it was shown by Lawler et al. [1980] that it cannot be solved in output polynomial time (unless $\mathbf{P} = \mathbf{NP}$). □

3.1.3. Inducing Operators

In order to find tractable subclasses of Problem 3.1, it is a reasonable strategy to identify the structural properties that allowed an efficient algorithm for the special case of listing the closed frequent sets and try to generalize them as far as possible. Namely, for the efficient enumeration of closed frequent sets it is essential that the task can be reduced to computing closures of already enumerated closed sets augmented by single elements. That is, one can exploit an *underlying* closure operator.

We formalize this notion as follows. A mapping $\rho \colon \mathcal{F} \to \mathcal{F}$ **induces** a family $\mathcal{F}' \subseteq \mathcal{F}$ if $\mathcal{F}'$ is the image of $\mathcal{F}$ under ρ, i.e., $\mathcal{F}' = \rho(\mathcal{F}) = \{\rho(F) \colon F \in \mathcal{F}\}$. For a closure operator ρ this means that $\mathcal{F}'$ corresponds to the fixpoints of ρ. In case a closed constrained set family $\mathcal{CC}(\mathcal{F}, \mathcal{D})$ is induced by a closure operator ρ we call ρ the **support closure operator** of $\mathcal{F}$ with respect to $\mathcal{D}$. For instance one can check that the following mapping is a support closure operator for the setting of Example 3.1:

$$\sigma_{\text{conn}} \colon F \mapsto \text{vertices of connected component of } G[\cap\mathcal{D}[F]] \text{ containing } F \ .$$

Note that σ_{conn} is only a partial function of $\mathcal{P}(E)$ because in general there is no connected component in $G[\cap\mathcal{D}[F]]$ containing an unconnected F. For instance in the example of Figure 3.2 the image $\sigma_{\text{conn}}(\{a, b\})$ is undefined. Whenever there is an efficiently computable support closure operator, Problem 3.1 becomes equivalent to the following problem.

Problem 3.2 (LIST-FIXPOINTS). *Given* a set system $(E, \mathcal{F})$ with $\emptyset \in \mathcal{F}$ and a closure operator $\rho : \mathcal{F} \to \mathcal{F}$, *list* the elements of $\rho(\mathcal{F})$. $\Diamond$

This problem may be interesting in its own right as the total function case is also studied in application-independent domains (see, e.g., Ganter and Reuter [1991]). The partial function case has not yet been investigated and, as we show in the next section, behaves very differently in terms of computational complexity.

3.2. Complexity of Fixpoint Listing

Clearly, for any set system $(E, \mathcal{F})$ and closure operator ρ on $\mathcal{F}$, the family of fixpoints $\rho(\mathcal{F})$ can be listed in total time $O(2^n)$ by a deterministic algorithm that has access to $\mathcal{F}$ only by means of membership oracle and closure computations (if the invocation of the membership oracle and the closure computation are both charged by unit time). Theorem 3.2 below shows that this bound cannot be substantially improved for accessible set systems, and also implies that there is no deterministic algorithm solving LIST-FIXPOINTS for this problem fragment in output polynomial time, i.e., by an algorithm having a time complexity that is polynomially bounded in $n + |\rho(\mathcal{F})|$.

On the one hand, the result is of theoretical interest as it shows that the partial function case is harder than the total function case, for which we know that it can be solved by an amortized linear number of closure and membership computations. On the other hand, the concrete case of accessible set systems is of practical interest as it occurs in the local pattern discovery literature in its own right. Examples are *error-tolerant* frequent sets [Yang et al., 2001] or *constraints on the variance* of certain weights that are associated to each element of the ground set [Bonchi and Lucchese, 2007].

Theorem 3.2. *For accessible set systems $(E, \mathcal{F})$ and closure operators ρ on $\mathcal{F}$ such that $|\rho(\mathcal{F})| \leq 2$, there is no deterministic algorithm that has access to $\mathcal{F}$ only by means of membership oracle and closure computations, and correctly solves problem* LIST-FIXPOINTS *by invoking the membership oracle and computing the closure operator at most $2^{n/4}$ times where $n = |E|$.*

Proof. Let $\mathfrak{A}$ be a deterministic algorithm solving the problem described in the claim by invoking the membership oracle and computing the closure operator at most $2^{n/4}$ times. We show that $\mathfrak{A}$ is incorrect by constructing two problem instances such that $\mathfrak{A}$ fails to compute the correct fixpoint family for at least one of them.

For a positive integer $n > 4$ dividable by 4, consider the set system $(E, \mathcal{F})$ with $E = X \cup Y$ such that X, Y are disjoint sets, both of cardinality $n/2$, $\mathcal{F} = \mathcal{P}(X)$, and define the mapping $\rho : \mathcal{F} \to \mathcal{F}$ by $\rho(F) = X$ for all $F \in \mathcal{F}$. Clearly $(E, \mathcal{F})$ is accessible, ρ is a closure operator, and $|\rho(\mathcal{F})| = 1$. Thus, $(E, \mathcal{F})$ and ρ form an instance of the problem described in the claim.

Since $\mathfrak{A}$ invokes the membership oracle or computes the closure operator at most $2^{n/4} < \binom{n/2}{n/4}$, there is at least one subset of Y of cardinality $n/4$, say $Y' = \{e_1, \ldots, e_{n/4}\}$, such that $\mathfrak{A}$ on input $\mathcal{F}$ and ρ does not access (neither by the membership oracle nor by the closure computation) $Y' \cup F$ for all $F \subseteq X$. For the same reason there is a subset $X' \subseteq X$ of cardinality $n/4$ such that $\mathfrak{A}$ does not access $X' \cup F$ for all $F \subseteq Y$. Let $X \setminus X' = \{e_{n/4+1}, \ldots, e_{n/2}\}$ and consider the set system $(E, \mathcal{F}')$ with $\mathcal{F}' = \mathcal{F} \cup \{S_1, \ldots, S_{n/2}\}$, where $S_0 = X'$ and $S_i =$

$S_{i-1} \cup \{e_i\}$ for every $i = 1, \ldots, n/2$. Note that $S_{n/2} = X \cup Y'$. Let the function $\rho' : \mathcal{F}' \to \mathcal{F}'$ be defined by

$$\rho' : F \mapsto \begin{cases} X, & \text{if } F \in \mathcal{F} \\ S_{n/2}, & \text{otherwise} \end{cases} .$$

One can easily check that $(E, \mathcal{F}')$ is accessible, ρ' is a closure operator on $\mathcal{F}'$, and $|\rho'(\mathcal{F}')| = 2$. Hence, $(E, \mathcal{F}')$ and ρ' form a second instance of the problem defined in the statement.

Let $A_1(F_1), \ldots, A_k(F_k)$ be the sequence of membership queries and closure computations performed by $\mathfrak{A}$ on the first instance defined by $(E, \mathcal{F})$ and ρ. That is, $A_i(F_i)$ is either $M_{\mathcal{F}}(F_i)$ or $\rho(F_i)$ for some $F_i \in \mathcal{P}(E) \setminus \{S_0, S_1, \ldots, S_{n/2}\}$ for every $i = 1, \ldots, k$, where $M_{\mathcal{F}}$ denotes the membership oracle for $\mathcal{F}$. Since $\mathfrak{A}$ is deterministic, $F_i \notin \{S_0, S_1, \ldots, S_{n/2}\}$, and $M_{\mathcal{F}}(X) = M_{\mathcal{F}'}(X)$ and $\rho(X) = \rho'(X)$ for every $X \in \mathcal{P}(E) \setminus \{S_0, S_1, \ldots, S_{n/2}\}$, $\mathfrak{A}$ performs the same sequence of membership queries and closure computations for the second instance defined by $(E, \mathcal{F}')$ and ρ' and generate the same family of fixpoints. But this implies that $\mathfrak{A}$ is incorrect on at least one of the two instances, as $\rho(\mathcal{F}) \neq \rho'(\mathcal{F}')$. □

3.3. Divide-And-Conquer Fixpoint Listing

Algorithm 5 Divide-and-Conquer Fixpoint Listing

```
Input  : finite set system $(E, \mathcal{F})$ with $\emptyset \in \mathcal{F}$ and closure operator $\rho$ on $\mathcal{F}$
Output: family of fixpoint $\rho(\mathcal{F})$

MAIN:
 1. print $\rho(\emptyset)$
 2. LIST $(\rho(\emptyset), \emptyset)$

LIST$(C, B)$:
 1. choose an element $e \in E \setminus (C \cup B)$ satisfying $C \cup \{e\} \in \mathcal{F}$ if such an $e$ exists; otherwise
    return
 2. $C' \leftarrow \rho(C \cup \{e\})$
 3. if $C' \cap B = \emptyset$ then
 4.   print $C'$
 5.   LIST $(C', B)$
 6. LIST $(C, B \cup \{e\})$
```

In this section we analyze a simple divide-and-conquer algorithm for Problem 3.2 that recursively applies the following principle: For the current fixpoint C, first list all closed supersets of C containing some augmentation element e and then all closed supersets of C not containing e. This is a well-known listing scheme for globally defined closure operators (see Besson et al. [2005], Cerf et al. [2009], Gély [2005]), and it can be modified easily such that it is defined for any $\mathcal{F} \subseteq \mathcal{P}(E)$ with $\emptyset \in \mathcal{F}$. See Algorithm 5 for a pseudocode. In contrast to the references cited above, we present the algorithm on a high level of abstraction that considers general closure operators instead of the standard support closure operator with respect to a given binary dataset. Hence, any speed-ups specific to that concrete closure operator are not discussed here. Instead we show the algorithms correctness relative to the domain of the input closure operator and discuss its asymptotic amortized performance.

3.3.1. An Algorithmic Characterization of Strong Accessibility

We start with showing that, for strongly accessible set systems, the algorithm correctly solves LIST-FIXPOINTS. In contrast, if the input set system is not strongly accessible, Algorithm 5 is not even "correct" for the identity map as closure operator. This statement builds on the following notion of *correctness*:

(i) Algorithm 5 *behaves correctly* on input $(E, \mathcal{F})$ and ρ if it exactly and non-redundantly prints the elements of $\rho(\mathcal{F})$ for *all correct implementations of line 1* in LIST, i.e., for all correct choices of augmentation elements.

(ii) Moreover, we say that Algorithm 5 is correct for a set system $(E, \mathcal{F})$ if for *all closure operators* ρ on $\mathcal{F}$ it behaves correctly on input $(E, \mathcal{F})$ and ρ.

The motivation for requiring correct behavior for all choices of augmentation elements (item (i) above) is that, in case there is only some sequences of choices leading to the correct output, it is unclear how to find such a sequence in general.

Theorem 3.3. *Let $(E, \mathcal{F})$ be a set system with $\emptyset \in \mathcal{F}$. Algorithm 5 is correct for $(E, \mathcal{F})$ if and only if $(E, \mathcal{F})$ is strongly accessible.*

Proof. ("$\Leftarrow$") Let $(E, \mathcal{F})$ be strongly accessible and ρ be a closure operator on $\mathcal{F}$. For $C, B \subseteq E$, let $\mathcal{C}(C, B) = \{C' \in \rho(\mathcal{F}) \colon C' \supset C \wedge C' \cap B = \emptyset\}$. We prove by induction on the height h of the recursion tree of LIST(C, B) that

$$\text{LIST}(C, B) \text{ prints exactly } \mathcal{C}(C, B) \text{ and} \tag{3.1}$$

$$\text{LIST}(C, B) \text{ prints no set more than once.} \tag{3.2}$$

Since MAIN calls LIST$(\rho(\emptyset), \emptyset)$ and prints only the fixpoint $\rho(\emptyset)$, this concludes the proof of the sufficiency. For $h = 0$, no augmentation element is selected in line 1. Therefore, LIST(C, B) prints no element on the one hand and, as $(E, \mathcal{F})$ is strongly accessible, $\mathcal{C}(C, B) = \emptyset$ on the other hand, from which (3.1) and (3.2) directly follow. For the induction step $h > 0$ we must have that an augmentation element $e \in E \setminus (C \cup B)$ has been selected in line 1. We distinguish two cases depending on $C' = \rho(C \cup \{e\})$ computed in line 2:

(i) Suppose $C' \cap B \neq \emptyset$. Then the set of fixpoints printed by LIST(C, B) is equal to the set $\mathcal{L}$ printed by LIST$(C, B \cup \{e\})$. Applying the induction hypothesis to LIST$(C, B \cup \{e\})$, we get $\mathcal{L} = \mathcal{C}(C, B \cup \{e\})$ and (3.2). Thus, to prove (3.1) it suffices to show that $\mathcal{C}(C, B \cup \{e\}) = \mathcal{C}(C, B)$. Clearly, $\mathcal{C}(C, B \cup \{e\}) \subseteq \mathcal{C}(C, B)$. Conversely, let $C'' \in \mathcal{C}(C, B)$. Then $e \notin C''$, as otherwise we would have $\rho(C \cup \{e\}) \subseteq \sigma(C'') = C''$ by the monotonicity and idempotence of ρ and hence, $C'' \cap B \neq \emptyset$ contradicting $C'' \in \mathcal{C}(C, B)$. Thus, $C'' \in \mathcal{C}(C, B \cup \{e\})$.

(ii) Suppose $C' \cap B = \emptyset$. Then the family printed by LIST(C, B) is equal to $\{C'\} \cup \mathcal{L}_1 \cup \mathcal{L}_2$, where C' is the fixpoint printed in line 4 and $\mathcal{L}_1, \mathcal{L}_2$ are the families printed by LIST(C', B)

and $\textsc{list}(C, B \cup \{e\})$, respectively. Let $C'' \in \mathcal{C}(C, B)$ with $e \in C''$ and $C' \neq C''$. Then $C'' \in \mathcal{C}(C', B)$ for $C \cup \{e\} \subseteq C''$ and $C' = \rho(C \cup \{e\}) \subset \rho(C'') = C''$. Thus, $\mathcal{C}(C, B) = \{C'\} \cup \mathcal{C}(C', B) \cup \mathcal{C}(C, B \cup \{e\})$, from which (3.1) directly follows by applying the induction hypothesis to $\textsc{list}(C', B)$ and $\textsc{list}(C, B \cup \{e\})$. Since $C' \notin \mathcal{L}_1 \cup \mathcal{L}_2$ and $\mathcal{L}_1 \cap \mathcal{L}_2 = \emptyset$ because $\mathcal{C}(C', B) \cap \mathcal{C}(C, B \cup \{e\}) = \emptyset$, we get (3.2) by applying the induction hypothesis to $\textsc{list}(C', B)$ and $\textsc{list}(C, B \cup \{e\})$.

("$\Rightarrow$") Suppose that $(E, \mathcal{F})$ is not strongly accessible. Then choose $X = \{x_1, \ldots, x_k\} \in \mathcal{F}$ minimal such that there is a $Y \in \mathcal{F}$ with $X \subset Y$ and $(X \cup \{y\}) \notin \mathcal{F}$ for all $y \in Y \setminus X$. Let ρ be the identity map on $\mathcal{F}$, which trivially is a closure operator. We show that there are possible choices of augmentation elements that result in an incorrect output. Consider the sequence of recursive calls

$$\textsc{list}(X_0, \emptyset), \textsc{list}(X_1, \emptyset), \ldots, \textsc{list}(X_k, \emptyset)$$

with $X_i = \{x_1, \ldots, x_i\}$, which arises as prefix of the call sequence in case x_i is chosen as augmentation element in line 1 of $\textsc{list}(X_{i-1}, \emptyset)$. If there is an augmentation element e in $\textsc{list}(X_k, \emptyset)$ then $e \notin Y$ by the choice of X and Y. Thus, Y can neither be found in that incarnation of $\textsc{list}$ nor in any of its subcalls $\textsc{list}(C, B)$ because $e \in C$ for all such calls. For all other subsequent calls backtracking has occurred at least once. Thus, $x_i \in B$ for some $i \in \{1, \ldots, k\}$ for all such calls $\textsc{list}(C, B)$, and consequently Y will be rejected by the check in line 4 in case it is found. Altogether, Algorithm 5 does not print Y and, hence, is incorrect for input $(E, \mathcal{F})$ and ρ. □

As a byproduct of Theorem 3.3 we get an *algorithmic characterization* of strong accessibility. Note that the degree of freedom of the closure operator is not necessary for the proof of the "only if"-part of that theorem. Thus, the characterization can simply be stated as in Theorem 3.4 below. Recall that the underlying notion of correctness involves correct computation for *all* implementations that are in accord with the pseudo-code specification of Algorithm 5; in particular this means correctness for all valid choices of augmentation elements in line 1.

Theorem 3.4. *Let $(E, \mathcal{F})$ be a set system with $\emptyset \in \mathcal{F}$. Then $(E, \mathcal{F})$ is strongly accessible if and only if Algorithm 5 correctly lists $\mathcal{F}$ on input $(E, \mathcal{F})$ and the identity operator on $\mathcal{F}$ (as closure operator).*

This characterization is a distinctive feature that separates Algorithm 5 from related algorithms such as the generalized LCM algorithm (see Chapter 2). Section 3.6 gives a more detailed comparison between these algorithms.

3.3.2. Performance

We now turn to the complexity of Problem 3.2 restricted to strongly accessible set systems. In addition to the size of the input ground set E, the complexity also depends on the representation of the set system and on the closure operator. Accordingly, we will study the time and space complexity also in terms of those of (i) checking membership in $\mathcal{F}$ and (ii) computing the closure of an element in $\mathcal{F}$. We denote by $T_{\mathcal{F}}$, $S_{\mathcal{F}}$, T_ρ, and S_ρ the maximum time and space requirements of these operations for an input of size $|E|$, respectively. We assume that single elements of E can be stored, compared, or otherwise manipulated in time and space $O(1)$. For environments violating this assumption all complexities have to be multiplied by $\log |E|$.

Algorithm 6 Modified Divide-and-Conquer Fixpoint Listing

```
MAIN′:
 1. initialize $C$ and $B$ to be empty stacks
 2. let $e_1, \ldots, e_n$ be an arbitrary fixed ordering of $E$
 3. push all $c \in \rho(\emptyset)$ onto $C$
 4. print $C$
 5. LIST′(1)

LIST′($d$):
 1. push $\perp$ onto $C$, push $\perp$ onto $B$ //set restoration point
 2. for $i = 1, \ldots, n$ do
 3.   if $e_i \in (C \cup B)$ or $C \cup \{e_i\} \notin \mathcal{F}$ then continue with next $i$
 4.   push all $c \in (\rho(C \cup \{e_i\}) \setminus C)$ onto $C$
 5.   if $C \cap B = \emptyset$ then
 6.     if $d$ is even then
 7.       print $C$
 8.       LIST′($d + 1$)
 9.     else
10.       LIST′($d + 1$)
11.       print $C$
12.   while top of $C$ not equal to $\perp$ do pop $C$ //restore $C$
13.   push $e_i$ onto $B$
14. while top of $B$ not equal to $\perp$ do pop $B$ //restore $B$
15. pop $C$, pop $B$ //remove restoration point
```

For our analysis we consider a modified formulation of the divide-and-conquer algorithm given in Algorithm 6. While the two algorithms are input-output equivalent, Algorithm 6 allows us to prove a stronger performance statement. The changes are:

M1 The tail-recursion of the LIST procedure is replaced by a for-loop iterating over all potential augmentation elements $e \in E \setminus (C \cup B)$ using an arbitrary but fixed ordering of E.

M2 The parameters of the LIST procedure are replaced by global variables. In particular this can be realized by implementing them as stacks of single elements of the ground set because—due to the recursive structure of the algorithm—elements are added and removed in a last-in-first-out fashion.

M3 For odd recursion depths the order of lines 4 and 5 of Algorithm 5 is changed, i.e., the new fixpoint is printed only after the recursive call backtracked.

Modifications M1 and M2 are only equivalent reformulations. The third modification follows an idea described by Nakano and Uno [2004] in the context of listing trees of a fixed diameter. It does not change what is printed—and thus does not affect correctness—but "holds back" some of the output elements for a certain time until eventually printing them. As a result, the points in time in which fixpoints are printed are more evenly distributed during the running time of the algorithm. This improves the *delay*, i.e., the maximum time between the generation of two consecutive fixpoints, by a factor of $|E|$ over that of the original formulation, without changing the total time. It is not surprising that this is possible because one can show that Algorithm 5 has cumulative polynomial delay, and any algorithm satisfying this property can be transformed in such a way (see Section 2.2 and Goldberg [1993]). Still, the explicit transformation is instructive and therefore given here.

Theorem 3.5. *Restricted to strongly accessible set systems,* LIST-FIXPOINTS *can be solved with*

$$\text{delay } O(|E|\ (T_{\mathcal{F}} + T_\rho + |E|))\ \ , \text{ and} \tag{3.3}$$
$$\text{space } O(|E| + S_{\mathcal{F}} + S_\rho)\ \ . \tag{3.4}$$

Proof. Let $n = |E|$. To see the *delay*, first observe that the algorithm cannot backtrack more than two times without printing a new set or terminating. Moreover, if LIST$'(d)$ is called with an even d, without printing a new fixpoint or backtracking there can be at most

- n membership checks (cost $T_{\mathcal{F}}$ each),
- n closure computations (cost T_ρ each) and
- n manipulations and accesses to a constant number of variables of size at most $O(n)$ (cost $O(n)$ each).

Finally, for LIST$'(d)$ with an odd d there cannot be more than the same operations of time $O(T_{\mathcal{F}} + T_\rho + n)$ without calling LIST$'$ with an even d or backtracking. The claimed delay follows by noting that MAIN$'$ prints a set and calls LIST$'$ after time $O(T_\rho + n)$.

The *space complexity* is straightforward: since $C \cup B \subseteq E$ always holds, there are never more than $O(n)$ elements to be stored. Equation (3.4) directly follows. □

From this theorem it immediately follows a bound on the total computation time (i.e., $|\rho(\mathcal{F})|$ times the delay given in Equation (3.3)) because of the exactness and non-redundancy guaranteed by Theorem 3.3. In fact this bound on the total time and the space bound can already be shown for the algorithm incorporating only the equivalent reformulations M1 and M2 above. The amortized cost of a single invocation of LIST is similar to the one found in the proof of Theorem 3.5 and non-redundancy and exactness imply that LIST(C', B) in line 5 of Algorithm 5 is called at most once for each $C' \in \rho(\mathcal{F})$. Closing this section, we can thus note:

Remark 3.1. Restricted to strongly accessible set systems, Algorithm 5 can be implemented to solve LIST-FIXPOINTS with total time

$$O(|E|\ (T_{\mathcal{F}} + T_\rho + |E|)\ |\rho(\mathcal{F})|)$$

and space as given in Equation (3.4). $\diamondsuit$

3.4. Existence of Support Closure Operators

After having investigated what set systems allow the efficient listing of closure operator fixpoints, we now turn to the question what set systems guarantee the existence of a support closure operator, i.e., a closure operator that induces the closed constrained sets. As a first observation we note that such an operator can always be expressed in a canonical form. That is, if the support closure operator exists it is uniquely defined as follows:

Lemma 3.6. *Let $(E, \mathcal{F})$ be a set system and $\mathcal{D}$ a dataset over E. If a support closure operator σ on $\mathcal{F}$ with respect to $\mathcal{D}$ exists then it is well-defined by*

$$\begin{aligned} &\sigma(F) = \max \Sigma(F) \\ &\text{where } \Sigma(F) = \{F' \in \mathcal{F} \colon F \subseteq F' \wedge \mathcal{D}[F] = \mathcal{D}\big[F'\big]\} \ , \end{aligned} \tag{3.5}$$

i.e., the family $\Sigma(F)$ has a unique maximal element for all $F \in \mathcal{F}$.

Proof. Note that for all $F \in \mathcal{F}$,

$$F \in \mathcal{CC}(\mathcal{F}, \mathcal{D}) \iff \exists G \in \mathcal{F} \text{ such that } F \text{ is maximal in } \Sigma(G) \ . \tag{3.6}$$

Let σ be a support closure operator on $\mathcal{F}$ and F' be a maximal element in $\Sigma(F)$. Then F' is closed constrained by (3.6) and hence $\sigma(F') = F'$. Since $F \subseteq F'$, we have $F \subseteq \sigma(F) \subseteq \sigma(F') = F'$ by extensivity and monotonicity of σ. But this implies $\mathcal{D}[F] = \mathcal{D}[\sigma(F)]$, as $\mathcal{D}[F] = \mathcal{D}[F']$ by $F' \in \Sigma(F)$. Thus $\sigma(F) \in \Sigma(F)$. But then, $\sigma(F)$ must be maximal in $\Sigma(F)$, as it is maximal in $\Sigma(\sigma(F))$ by (3.6). Hence, $\sigma(F) = F'$ because $\sigma(F) \subseteq F'$. $\square$

A unique maximal element of $\Sigma(F)$ does not always exist (e.g., $\Sigma(\emptyset) = \{\{a\}, \{b\}\}$ for $\mathcal{F} = \{\emptyset, \{a\}, \{b\}\}$ and $\mathcal{D} = \{\{a, b\}\}$), and even if it exists, σ defined by (3.5) is not always monotone. For instance, consider $\mathcal{F} = \{\emptyset, \{a\}, \{a, b\}, \{a, c\}\}$ with $\mathcal{D} = \{\{a, b\}, \{a, b, c\}\}$. Then, although $\{a\} \subseteq \{a, c\}$, we have $\sigma(\{a\}) = \{a, b\} \not\subseteq \sigma(\{a, c\}) = \{a, c\}$. If, however, $\max \Sigma(F)$ is unique for every $F \in \mathcal{F}$ and σ as defined above is monotone, than σ must also be the support closure operator.

We now show that confluence characterizes the existence of the support closure operator for arbitrary non-redundant datasets. That is, restricted to confluent set systems, Problem 3.1 is a subproblem of Problem 3.2.

Theorem 3.7. *Let $(E, \mathcal{F})$ be a set system. The support closure operator for $\mathcal{F}$ with respect to $\mathcal{D}$ exists for all non-redundant datasets $\mathcal{D}$ over E if and only if $(E, \mathcal{F})$ is confluent.*

Proof. ("$\Leftarrow$") Suppose $(E, \mathcal{F})$ is confluent and let $\mathcal{D}$ be a non-redundant dataset over E. Let $\sigma(F)$ be $\max \Sigma(F)$ for all $F \in \mathcal{F}$ as defined in (3.5). We prove that σ is a support-closure operator by showing that

(i) σ is a function,

(ii) $\sigma(F)$ is closed constrained for all $F \in \mathcal{F}$, and

(iii) σ is a closure operator.

Regarding (i), we show that for all $F \in \mathcal{F}$, there exists a unique maximal element in $\Sigma(F)$. For $F = \emptyset$ this is trivial because $\Sigma(\emptyset) = \{\emptyset\}$ by the non-redundancy of $\mathcal{D}$. Let $F \neq \emptyset$. Existence is implied by the finiteness of $\mathcal{F}$. For uniqueness assume there are distinct sets $F', F'' \in \mathcal{F}$ that are both maximal in $\Sigma(F)$. Since $F \neq \emptyset$, $F \subseteq F'$, and $F \subseteq F''$, it follows from the confluence of $\mathcal{F}$ that $(F' \cup F'') \in \mathcal{F}$. As $\mathcal{D}[F' \cup F''] = \mathcal{D}[F'] \cap \mathcal{D}[F'']$ and $\mathcal{D}[F'] = \mathcal{D}[F''] = \mathcal{D}[F]$, we have $\mathcal{D}[F' \cup F''] = \mathcal{D}[F]$. Thus $F' \cup F'' \in \Sigma(F)$ contradicting the maximality of F' and F''. Hence, there is a unique maximal element of $\Sigma(F)$.

Property (ii) is immediate by (3.6), as $\sigma(F)$ is maximal in $\Sigma(F)$. To see (iii), the extensivity follows by definition. For idempotence we have $\sigma(F) = \max \Sigma(F) \in \Sigma(F)$ implying $\mathcal{D}[\sigma(F)] = \mathcal{D}[F]$. Hence $\sigma(\sigma(F)) = \sigma(F)$, as $\sigma(F)$ is maximal. For monotonicity, let $F', F'' \in \mathcal{F}$ with $F' \subseteq F''$. The case $F' = \emptyset$ is trivial because $\sigma(\emptyset) = \emptyset$, as $\mathcal{D}$ is non-redundant. Let $F' \neq \emptyset$. Since $F' \subseteq \sigma(F')$ and $F' \subseteq F''$, $\sigma(F') \cup F'' \in \mathcal{F}$ by the confluence of $\mathcal{F}$. For the support of $\sigma(F') \cup F''$ we have $\mathcal{D}[\sigma(F') \cup F''] = \mathcal{D}[\sigma(F')] \cap \mathcal{D}[F'']$, which, in turn, is equal to $\mathcal{D}[F'']$ because $\mathcal{D}[\sigma(F')] = \mathcal{D}[F']$ by the definition of σ and $\mathcal{D}[F''] \subseteq \mathcal{D}[F']$ by $F' \subseteq F''$. Hence $\sigma(F') \cup F'' \in \Sigma(F'')$ and $\sigma(F') \subseteq \sigma(F') \cup F'' \subseteq \sigma(F'')$ by maximality of $\sigma(F'')$.

("$\Rightarrow$") Suppose that for all non-redundant datasets $\mathcal{D}$ over E the support closure of $\mathcal{F}$ with respect to $\mathcal{D}$ exists. In order to show that $(E, \mathcal{F})$ is confluent let $I, X, Y \in \mathcal{F}$ with $I \neq \emptyset$, $I \subseteq X$, and $I \subseteq Y$. We show that $\sigma(I) = X \cup Y$ for the support closure operator $\sigma\colon \mathcal{F} \to \mathcal{F}$ with respect to the dataset $\mathcal{D} = \{\emptyset, X \cup Y\}$. Since, on the one hand, σ is support preserving it follows that $\sigma(I) \subseteq X \cup Y$. On the other hand, $\sigma(I)$ is closed constrained. Together with $\mathcal{D}[I] = \mathcal{D}[X] = \mathcal{D}[Y]$ this implies that $X \subseteq \sigma(I)$ and $Y \subseteq \sigma(I)$. Hence, it also holds that $\sigma(I) \supseteq X \cup Y$ as required. $\square$

Theorem 3.7 can be used to characterize the instances of LIST-CLOSED-CONSTRAINED that are also instances of LIST-FIXPOINTS. But even in case that the support closure operator exists, it is unclear whether its computation is tractable. In the following lemma we show that if a support closure operator has a strongly accessible domain, it can be computed efficiently by reducing it to the **augmentation problem** (line 1 of Algorithm 5), i.e., to the problem of finding an element $e \in E \setminus (B \cup C)$ with $(C \cup \{e\}) \in \mathcal{F}$ or deciding that none exists, given $B, C \subseteq E$. We denote the required time to solve this problem by T_a. Note that it can always be solved with $|E \setminus (C \cup B)|$ membership queries (and no additional space). We nevertheless make T_a an explicit parameter of the result below, because usually it can be implemented more efficiently than by the naive approach via membership queries (see the examples from Section 3.5).

Lemma 3.8. *Let $(E, \mathcal{F})$ be a strongly accessible set system and $\mathcal{D}$ be a non-redundant dataset over E. If the support closure operator of $\mathcal{F}$ with respect to $\mathcal{D}$ exists it can be computed in time $O(|E|\ (|\mathcal{D}| + T_a))$ and space $S_\mathcal{F}$.*

Proof. Let σ be a support closure operator. Define $F_0 = F$ and

$$F_{i+1} = \begin{cases} F_i \cup \{e\} & \text{if } \exists e \in \bigcap \mathcal{D}[F] \setminus F_i \text{ such that } F_i \cup \{e\} \in \mathcal{F} \\ F_i & \text{otherwise} \end{cases}$$

for $i \geq 0$. Since the sequence $F = F_0 \subseteq F_1 \subseteq \ldots$ is bounded by $\bigcap \mathcal{D}[F]$, there is a smallest index $k < |E|$ such that $F_k = F_{k+1}$. Clearly, $\mathcal{D}[F] = \mathcal{D}[F_i]$ and hence, $F_i \in \Sigma(F)$ for every $i = 0, 1, \ldots, k$. Since there is no further augmentation element $e \in (\bigcap \mathcal{D}[F] \setminus F_k)$ and $(E, \mathcal{F})$ is strongly accessible, it follows that F_k is maximal in $\Sigma(F)$. Thus, by Lemma 3.6, $F_k = \sigma(F)$ as required. By the definition above, F_k can be computed by calculating $\bigcap \mathcal{D}[F]$ and by finding at most $|E|$ augmentation elements. The statement about the time then follows because $\bigcap \mathcal{D}[F]$ can be computed in time $O(|E|\ |D|)$. For the required space note that for the computation of the result F_k there is no additional storage required beside that for computing an augmentation element, which can be reused. $\square$

Combining Theorem 3.5 with the results of this section, we can identify a fairly general, tractable subproblem of LIST-CLOSED-CONSTRAINED. While the theorem below may not yield the strictest bounds for concrete problems where more structural assumptions hold, its conditions can usually be checked easily and it serves as a baseline for more specialized methods.

Theorem 3.9. *Restricted to set systems that are confluent and strongly accessible* LIST-CLOSED-CONSTRAINED *can be solved with delay $O(|E|^2\ (|\mathcal{D}| + T_a))$ and space $O(|E| + S_\mathcal{F})$ respectively with delay $O(|E|^2\ (|\mathcal{D}| + |E|\, T_\mathcal{F}))$ if the augmentation problem is solved by membership queries.*

Note that it is crucial for Theorem 3.9 that Theorem 3.5 holds for closure operators that are only a partial function of the power set of the ground set. The support closure operator is in general not defined for arbitrary members of the power set.

3.5. Application of Main Results

In this section, we give exemplary applications of Theorems 3.5 and 3.9. Beside two problems involving structural constraints, we also reconsider the problem of listing ordinary closed frequent sets. This yields an alternative proof of the efficient enumerability of the closed sets and shows that the depth-first search based approach and the divide-and-conquer approach have a similar theoretical performance for that problem.

3.5.1. Frequent Sets

As a first example, consider our standard problem of listing all closed frequent sets. For a dataset $\mathcal{D}$ over E and a threshold t let $\mathcal{F}_t = \{F \subseteq E : c_{\text{freq}(\mathcal{D},t)} = 1\}$ denote the family of t-frequent sets. The minimum frequency constraint with respect to a dataset $\mathcal{D}$ has the special property of being a function of the support equivalence classes (with respect to the same dataset). Hence, it holds that $\mathcal{CC}(\mathcal{F}_t, \mathcal{D}) = \mathcal{C}(\mathcal{D}) \cap \mathcal{F}_t$, and the following problem statement is a specialized version of Problem 2.3 from Chapter 2.

Problem 3.3 (LIST-CLOSED-FREQUENT). *Given* a dataset $\mathcal{D}$ over ground set E and a frequency threshold $t > 0$, *list* the family of closed frequent sets $\mathcal{CC}(\mathcal{F}_t, \mathcal{D})$ $(= \mathcal{C}(\mathcal{D}) \cap \mathcal{F}_t)$. $\Diamond$

Although we already know that this problem can be solved efficiently (Proposition 2.8), it is very useful to clarify some aspects of Theorem 3.9. Namely, it is perhaps confusing that the theorem states that the support closure operator exists for all datasets only for confluent set systems. This may appear contradictory to the fact that for the generally non-confluent set system of frequent sets (w.r.t. all datasets and frequency thresholds) a support closure operator always exists. For this case, however, the universal quantification over all datasets is in fact not used—one only needs the support closure operator with respect to the same dataset that induces the frequent set family. Indeed, the support closure operator does not always exist if one considers closedness and minimum frequency with respect to different datasets.

Now, the following line of argumentation shows how we can apply our theorems in the context of frequent sets. If we discard the frequency requirement, the underlying set system is $\mathcal{F} = \mathcal{P}(E)$, for which the support closure operator always exists and is equivalent to the standard support closure operator, i.e., $\sigma(F) = \bigcap \mathcal{D}[F]$. It can be computed in time $O(\text{size}(\mathcal{D}))$—recall that we define the size of $\mathcal{D}$ as $\text{size}(\mathcal{D}) = \sum_{D \in \mathcal{D}} |D|$. Moreover, as σ is frequency-preserving, the image of the family of t-frequent sets $\mathcal{F}_t$ under σ is a subset of $\mathcal{F}_t$. Hence, the restriction of σ to $\mathcal{F}_t$ is a mapping $\sigma_t : \mathcal{F}_t \to \mathcal{F}_t$ that is the support closure operator of $(E, \mathcal{F}_t)$ with respect to $\mathcal{D}$. As $\mathcal{F}_t$ is an independence system and a membership test can be performed in time $O(\text{size}(\mathcal{D}))$, by Theorem 3.5 we get:

Corollary 3.10. *The* LIST-CLOSED-FREQUENT *problem can be solved with delay* $O(|E| \text{ size}(\mathcal{D}))$, *and space* $O(|E|)$.

In this example we used a general observation about support closure operators of a set system $(E, \mathcal{F})$: their restriction to the family $\mathcal{F}_t$ of t-frequent sets of $\mathcal{F}$ remains a support closure operator (with respect to the same dataset $\mathcal{D}$). Thus, if $\mathcal{F}$ is confluent and strongly accessible, Theorem 3.9 can still be applied when a frequency constraint is added to Problem 3.1.

Nevertheless, for LIST-CLOSED-FREQUENT the application of Theorem 3.5 over Theorem 3.9 leads to a stronger performance bound because the support closure operator for frequent sets can be computed faster than by the algorithm used in the proof of Lemma 3.8. The bound in Corollary 3.10 achieved by the divide-and-conquer approach is in fact equal to the best known theoretical bound for LIST-CLOSED-FREQUENT achieved by the lcm algorithm (Proposition 2.8). Hence, both algorithms have an equally well performance for that problem. For general constraints and closure operators, however, the two approaches exhibit subtle differences as discussed in Section 3.6.2.

3.5.2. Poset Ideals

The next example makes use of the fact that greedoids are strongly accessible. Let $(E, \leq)$ be a poset. Then $F \subseteq E$ is called a (lower) **ideal** if for all $e \in F$ and for all $e' \in E$, $e' \leq e$ implies $e' \in F$. By $\mathcal{F}_{\text{idea}(\leq)}$ we denote the family of all ideals of $(E, \leq)$. Using this notion, we can state the following listing problem:

Problem 3.4 (LIST-CLOSED-IDEALS)**.** *Given* a finite poset $(E, \leq)$ and a non-redundant dataset $\mathcal{D}$ over E, *list* the family $\mathcal{CC}(\mathcal{F}_{\text{idea}(\leq)}, \mathcal{D})$ of closed ideals. $\Diamond$

This problem is inspired by *recommendation systems* where partial orders are used for modeling a collection of user preferences (see, e.g., Bridge and Ferguson [2002]). In addition, partial orders are used in market basket analysis for modeling hierarchies of product categories [Srikant et al., 1997]. Maximal elements of a frequent ideal represent product groups with a high autocorrelation.

We show that LIST-CLOSED-IDEALS can be solved with polynomial delay and space assuming the partial order $\leq$ is given as an explicit list of its direct successor relation. It is well-known that the set system $(E, \mathcal{F}_{\text{idea}(\leq)})$ forms a greedoid, the so-called "poset greedoid" [Korte and Lovász, 1985], which implies that $(E, \mathcal{F})$ is strongly accessible. Furthermore, $(E, \mathcal{F})$ is confluent as poset greedoids are closed under union. An augmentation element (see line 1 of Algorithm 5) can be found in time $O(\text{size}(\leq))$ by touching each element (x, y) of the direct successor relation of $\leq$ and checking whether $x \in F$ and $y \notin (F \cup B)$. Altogether, by Theorem 3.9, we have the following result:

Corollary 3.11. *The* LIST-CLOSED-IDEALS *problem can be solved with delay* $O(|E|^2\ (|\mathcal{D}| + \text{size}(\leq)))$ *and space* $O(|E|)$*.*

3.5.3. Connected Induced Subgraphs

Finally, we consider the problem of listing all connected induced subgraphs of a graph $G = (V, E)$ that are closed with respect to a dataset over V. That is, we go back to Example 3.1 from the beginning of this chapter. Let $\mathcal{F}_{\text{conn}(G)} = \{X \subseteq V : c_{\text{conn}(G)}(X) = 1\}$ denote the vertex sets inducing connected subgraphs of G. The formal problem statement is:

Problem 3.5 (LIST-CLOSED-CONNECTED)**.** *Given* an undirected graph $G = (V, \mathcal{E})$ and a non-redundant dataset $\mathcal{D}$ over V, *list* the family $\mathcal{CC}(\mathcal{F}_{\text{conn}(G)}, \mathcal{D})$ of closed connected[1] sets. $\Diamond$

We note that the family of vertex sets that induce connected subgraphs of G is not an independence system because a subgraph of a connected graph is not necessarily connected. It is, however, strongly accessible and also confluent.

Lemma 3.12. *For all graphs $G = (V, \mathcal{E})$ the set system $(V, \mathcal{F}_{\text{conn}(G)})$ is strongly accessible and confluent.*

[1]Note that in contrast to standard problems in graph mining, Problem 3.5 does not rely by any means on subgraph isomorphism. In this work, closedness is always only defined with respect to set inclusion.

Proof. Recall that for a graph $G = (V, \mathcal{E})$ and $X \subseteq V$ we denote by $G[X]$ the subgraph of G induced by X. The confluence follows since the union of any two connected subgraphs of G with non-disjoint vertex sets is also connected. For strong accessibility, let $X, Y \in \mathcal{F}$ with $X \subset Y$. Assume there is no vertex $v \in Y \setminus X$ such that $X \cup \{v\} \in \mathcal{F}$. Then X and $Y \setminus X$ are disconnected in $G[Y]$ contradicting the choice of Y. □

As for the previous example, an augmentation element can be found in time $O(|\mathcal{E}|)$ by touching each edge e once and checking whether $x \in F$ and $y \notin (F \cup B)$ for $x, y \in e$. Therefore we obtain by Theorem 3.9:

Corollary 3.13. *The* LIST-CLOSED-CONNECTED *problem can be solved with delay* $O(|V|^2\ (|\mathcal{D}| + |\mathcal{E}|))$ *and space* $O(|V|)$.

This example also illustrates the role of the non-redundancy requirement in the definition of Problem 3.1. With respect to redundant datasets the support closure operator for the connectivity constraint does not always exist as the following example shows:

Example 3.2. Let $G = (\{a, b, c\}, \{\{a, b\}, \{b, c\}\})$ a path of length two and $\mathcal{D} = \{\{a, c\}\}$. Then there is no support closure operator σ for $\mathcal{F}$ with respect to $\mathcal{D}$ because each possible choice of $\sigma(\emptyset)$, either $\{a\}$ or $\{c\}$, would violate monotonicity. ◊

3.6. Summary and Discussion

In this chapter, we have investigated the theoretical background of closed constrained set discovery. In particular, we have shown under what conditions closed constrained set listing can be reduced to fixpoint enumeration of a partially defined closure operator. For the latter problem, we have given a lower bound for the complexity of listing all fixpoints of a closure operator defined on an accessible domain. In contrast, we have shown that efficient divide-and-conquer fixpoint listing is correct for all closure operators of a domain if and only if this domain is a strongly accessible set system. Strong accessibility generalizes the traditional data mining assumption of dealing with an anti-monotone constraint, i.e., an independence system.

We have shown several exemplary problems that demonstrate that all of our problem generalizations are indeed necessary. Indeed, for the problems LIST-CLOSED-IDEALS and LIST-CLOSED-CONNECTED the support closure operator is in general only a partial function on $\mathcal{P}(E)$ because σ is undefined for non-ideals and for vertex sets inducing disconnected graphs, respectively. Moreover, the set systems considered for both problems are not necessarily independence systems. Finally, while for LIST-CLOSED-IDEALS, the induced family of closed constrained sets always forms a *closure system*, i.e., E is an ideal of $(E, \leq)$ and $\mathcal{F}$ is closed under intersection, this is not the case for LIST-CLOSED-CONNECTED.

Generally, the course of this chapter was driven by two conceptual choices: the modification of the standard closedness notion to constrained closedness and the focus on the divide-and-conquer algorithm for fixpoint enumeration. In the remainder of this section we discuss possible alternatives to both of these choices before concluding with a list of potential follow-up research topics.

3.6.1. Constraints and the Standard Closedness Notion

The results presented in this chapter are based on partially defined support closure operators abounding from the notion of closed constrained sets. It is important to note that closed constrained sets and standard closed sets are in general incomparable in terms of the subset relation (see the Example of Fig. 3.2). In particular, the family of closed constrained sets can be much larger than the family of closed sets. While this effect is usually desired in order to retain representativity of the discovered sets, it can potentially override the output reduction aspect of closed set discovery[2]. Therefore, here we briefly investigate how the fixpoint enumeration technique transfers to the problem of enumerating all sets that are feasible and closed according to the standard notion—a family, which is guaranteed to be a subfamily of all closed sets (again according to the standard notion).

More precisely, we consider the task: given a set system $(E, \mathcal{F})$ and a dataset $\mathcal{D}$ over E, enumerate all elements of $\mathcal{F}$ that are also a fixpoint of the standard support closure operator σ induced by $\mathcal{D}$. The crucial difference in this scenario is that this operation is in general not closed on $\mathcal{F}$, i.e., there can be $F \in \mathcal{F}$ such that $\sigma(F) \notin \mathcal{F}$. Moreover, it is not efficient to apply unmodified depth-first or divide-and-conquer fixpoint enumeration of σ, because $\mathcal{C} = \sigma(\mathcal{P}(E))$ can be arbitrarily large compared to $\mathcal{C} \cap \mathcal{F}$. A straightforward adaption, that retains efficiency, is to prune subcalls for closed sets that are not an element of $\mathcal{F}$. It is relatively easy to see that this approach is complete for independence systems: all generators of a closed set $C \in (\mathcal{C} \cap \mathcal{F})$ must also be an element of $\mathcal{F}$; correctness follows by applying induction.

For strongly accessible set systems, however, this approach is incomplete. Consider the scenario given in Figure 3.3, which again is based on the connectivity constraint. All generators that divide-and-conquer and depth-first search construct from the feasible and closed set $\emptyset$ lead to the infeasible sets $\{a, d\}$ respectively $\{c, b\}$. By pruning these sets, the algorithm terminates without ever generating the feasible and closed set $\{a, b, c, d\}$. It is an open question whether there is an efficient algorithm that correctly enumerates all closed sets that are also member of a strongly accessible set system.

	a	b	c	d
1	1	0	0	1
2	0	1	1	0
3	1	1	1	1

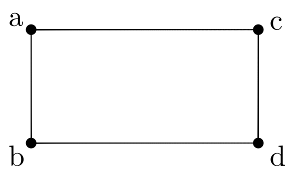


Figure 3.3.: Dataset $\mathcal{D}$ over ground set $V = \{a, b, c, d\}$ (left-hand side) and graph $G = (V, \mathcal{E})$ (right-hand side). The closed sets of $\mathcal{D}$ are $\mathcal{C} = \{\emptyset, ad, bc, V\}$. As $\sigma(\{a\}, \{b\}, \{c\}, \{d\}) = \{\{a, d\}, \{b, c\}\}$ is disjoint with $\mathcal{F}_{\text{conn}(G)}$, the set $V \in (\mathcal{C} \cap \mathcal{F}_{\text{conn}(G)})$ is not enumerated if infeasible closed sets are pruned.

[2]This comment only refers to the possibility of $|\mathcal{C}(\mathcal{D})| < |\mathcal{CC}(\mathcal{D}, \mathcal{F})|$. Of course we always have $|\mathcal{F}| > |\mathcal{CC}(\mathcal{D}, \mathcal{F})|$, i.e., using closed constrained set discovery always reduces the output compared to constraint-based pattern discovery.

3.6.2. Comparison of Divide-and-Conquer to Depth-First Approach

In contrast to traditional algorithms that assume $\mathcal{F} = \mathcal{P}(E)$ (e.g., Ganter and Reuter [1991]), straightforward depth-first traversal of the closed sets as well as the generalized lcm algorithm [Arimura and Uno, 2009] are also applicable to partially defined closure operators. In fact, as we show below, the class of set systems for which these algorithms are correct for *all* closure operators properly contains the class of strongly accessible set systems while it is properly contained in the class of accessible set systems. Thus, on the one hand, these algorithms are correct for a larger set of inputs than the divide-and-conquer approach (Alg. 5), but, on the other hand, for non-strongly accessible set systems it is intractable to decide whether their output is correct. In fact this observation sheds light on a distinctive feature of Algorithm 5: in contrast to the other algorithms it yields an exact algorithmic characterization of strong accessibility (see Section 3.3). In order to show the claims above, we briefly review both algorithms.

The depth-first-search approach can be seen as a straightforward traversal, starting in $\rho(\emptyset)$, of the **generator graph** $\mathbf{G}_{\mathcal{F},\rho} = (\mathbf{V}, \mathbf{E})$ induced by the input set system $(E, \mathcal{F})$ and the closure operator ρ (see also Chapters 2 and 5; p. 32 resp. p. 93), i.e., the directed graph with vertices $\mathbf{V} = \rho(\mathcal{F})$ and edges

$$\mathbf{E} = \{(C, C') \in (\rho(\mathcal{F}) \times \rho(\mathcal{F}))\colon \exists e \in E \setminus C,\ (C \cup \{e\}) \in \mathcal{F} \wedge \rho(C \cup \{e\}) = C'\} \ .$$

Consequently, it runs in total time $O(Nn(T_{\mathcal{F}} + T_\rho + n))$ but potentially exponential space $O(Nn)$, where $n = |E|$ and $N = |\rho(\mathcal{F})|$, because it explicitly stores each visited vertex. The algorithm of Arimura and Uno improves on the naive traversal of $\mathbf{G}_{\mathcal{F},\rho}$ in that it traverses only a spanning tree without this explicit storage. This results in an efficient space complexity $O(n + S_{\mathcal{F}} + S_\rho)$ and total time $O\big(N(n^3 T_{\mathcal{F}} + n^2 T_\rho)\big)$.

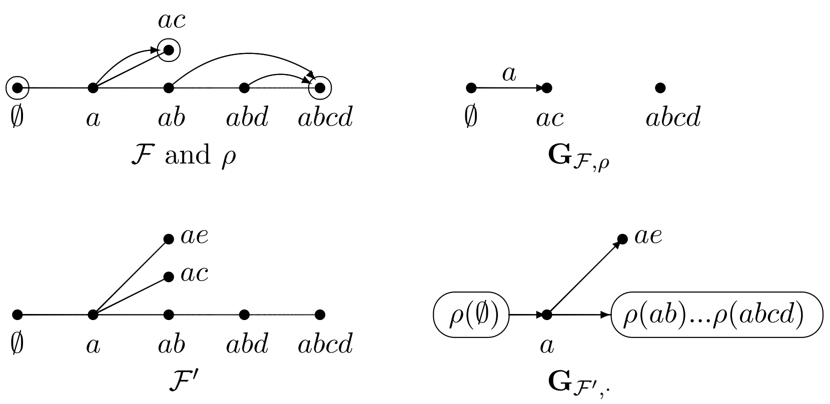
However, both algorithms are incomplete, thus incorrect, if $\mathbf{G}_{\mathcal{F},\rho}$ is unconnected and correct if it is connected. Recall that we call a closed set listing algorithm correct for a set system $(E, \mathcal{F})$ if it behaves correctly for *all* closure operators on $\mathcal{F}$. For the generator graph traversal algorithms this means that they are correct for $(E, \mathcal{F})$ if $\mathbf{G}_{\mathcal{F},\rho}$ is connected for *all* closure operators ρ on $\mathcal{F}$. It is straightforward to check that this connectivity criterion is, on the one hand, implied by $(E, \mathcal{F})$ being strongly accessible and, on the other hand, at least requires $(E, \mathcal{F})$ to be accessible (e.g., choose ρ to be the identity map). The two examples below (see also Figure 3.4) now show that this condition lies strictly between strong and ordinary accessibility.

First we give an accessible set system that is not strongly accessible and a closure operator such that the corresponding generator graph is unconnected.

Example 3.3. Let $(E, \mathcal{F})$ be the accessible set system defined by $E = \{a, b, c, d\}$ and $\mathcal{F} = \{\emptyset, a, ab, ac, abd, abcd\}$. Moreover, define $\rho : \mathcal{F} \to \mathcal{F}$ by $\rho(\emptyset) = \emptyset$, $\rho(a) = \rho(ac) = ac$, and $\rho(ab) = \rho(abd) = \rho(abcd) = abcd$. $\Diamond$

On the other hand, there are accessible set systems that are not strongly accessible and still have a connected generator graph for *all* closure operators. This is witnessed by the set system $(E', \mathcal{F}')$ of the following example.

Example 3.4. Let $(E', \mathcal{F}')$ be the accessible set system defined by $E' = E \cup \{e\}$ and $\mathcal{F}' = \mathcal{F} \cup \{ae\}$ with $(E, \mathcal{F})$ of Example 3.3. $\Diamond$



ac
$\emptyset$ a ab abd $abcd$
$\mathcal{F}$ and ρ
a
$\emptyset$ ac $abcd$
$\mathbf{G}_{\mathcal{F},\rho}$
ae
ac
$\emptyset$ a ab abd $abcd$
$\mathcal{F}'$
ae
$\rho(\emptyset)$
$\rho(ab)...\rho(abcd)$
a
$\mathbf{G}_{\mathcal{F}',\cdot}$

Figure 3.4.: The accessible set system $(E, \mathcal{F})$ and the closure operator ρ from Example 3.3 (top left) together with the corresponding generator graph $\mathbf{G}_{\mathcal{F}',\rho}$ (top right). The closed set $abcd$ is not reachable from $\emptyset$ in $\mathbf{G}_{\mathcal{F}',\rho}$. In addition the accessible set system $(E', \mathcal{F}')$ from Example 3.4 (bottom left). The corresponding generator graphs are connected for all closure operators (sketch bottom right).

Although $(E', \mathcal{F}')$ just like $(E, \mathcal{F})$ is not strongly accessible, one can check that $\mathbf{G}_{\mathcal{F},\rho}$ is connected for all closure operators ρ on $\mathcal{F}$. This is caused by a being a maximal subset of two distinct globally maximal sets ae and $abcd$. It follows due to monotonicity that a is a fixpoint of all closure operators on $\mathcal{F}$. In the corresponding generator graph a must connect $\sigma(\emptyset)$ (possibly itself) to both “arms”: $\{ae\}$ as well as $\{\rho(ab), \rho(ac), \rho(abc), \rho(abcd)\}$. Consequently and unlike Algorithm 5, the algorithms based on generator graph traversals are correct on $(E, \mathcal{F})$ for all closure operators.

Thus, these algorithms are correct for a larger set of inputs. In general, however, it is intractable to decide whether a given pair of an accessible set system and a closure operator induces a connected generator graph. This can be shown by using a similar construction as in the proof of Theorem 3.2. Accordingly, for a given accessible set system and closure operator it is intractable to decide in general whether the output of these algorithms is complete.

3.6.3. Directions for Future Research

The constraint classes introduced in this chapter are motivated from graphical applications such as the one of Example 3.1. A next step is to relate these classes to the many others that have been proposed in the pattern discovery community and clarify the status of important individual constraints. An exemplary question here is, which of the constraints from the literature that are known to be accessible are in fact strongly accessible?

In addition, it is important to investigate the combination of all types of constraints because constrained-based pattern discovery ideally allows a user to freely choose and combine constraints from a rich set of options. This also includes intra-class combinations such as for instance the constraint of being connected in two or more underlying graphs. In contrast to independence systems, however, the class of strongly accessible set systems is not closed under intersection, i.e., for strongly accessible set systems $(E, \mathcal{F})$ and $(E, \mathcal{F}')$, the set system $(E, \mathcal{F} \cap \mathcal{F}')$ is generally not strongly accessible. While this prevents the direct application of the divide-and-conquer algorithm, it does not rule out an efficient fixpoint enumeration algorithm for such set systems (cf., e.g., Lawler [1975] for an efficient algorithms for computing a maximum set in the intersection of two matroids—a class also not closed under intersection). Altogether, this raises the important theoretical questions of finding a precise characterization of set systems that allow efficient fixpoint enumeration for all closure operators. So far we only know that this class lies strictly between accessible and strongly accessible set systems.

Finally, note that the existence of support closure operators for *all* (non-redundant) datasets is a relatively strong property of a set system. In order to achieve a wider applicability of our results on fixpoint listing, it would be useful to have statements on restricted classes of datasets that have support closure operators even for set systems that are generally not confluent. Particularly useful in this context are classes that occur often in real-world applications and that allow an efficient membership check. They would lead to new practically effective algorithms for enumerating closed constrained sets.

4. A Quantification of Closedness

The standard approach for reducing output and computation time of closed frequent set discovery is to successively raise the frequency threshold until the resulting computation is tractable. By following this path we are eliminating less frequent sets in favor for sets that are more frequent; hence, essentially, we make use of the obvious fact that we can compare sets in terms of their frequency. In this chapter, we extend this idea to closedness, i.e., we develop a quantification of this notion that allows us to state that some sets are "more closed" than others. This leaves us with two adjustment parameters, one for frequency and one for closedness, which, as we show below, lead to efficiently enumerable pattern families that potentially are much more interesting than families that are described by minimum frequency constraints alone.

In order to define a quantification of closedness, recall that the defining property of a closed set is that there is a difference between its support and the support of all its augmentations. Thus, quantifying closedness can naturally be done by quantifying this difference. This intuition is captured by the following notion; a set is called "strongly closed", or "Δ-closed", if it cannot be augmented by any further element without reducing its support by at least Δ where Δ is some positive integer parameter. That is, strongly closed sets are sets that are at the boundary of a sharp drop in frequency. Beside being a reasonable generalization of ordinary closedness (i.e., 1-closedness), experimental results with different datasets indicate two other desirable properties of strongly closed sets: they are able to capture *long* patterns in reasonable time, even when this problem is difficult for frequency-based approaches and they are *stable*, i.e., robust against certain kinds of noise and/or dynamic changes in the data.

In a certain sense, the idea of strongly closed sets is a contrary approach to the constraint-based closedness from Chapter 3: Constrained closedness is a relaxation of the ordinary notion of closedness—instead of requiring all possible augmentations to be support-reducing, constrained closedness only asks for feasible augmentations being support-reducing. This is motivated by the goal of reducing redundancy among the feasible sets while at the same time retaining representativity, i.e., having at least one representative of each support equivalence class among the feasible sets. In contrast, strong closedness is a stricter notion than ordinary closedness. It deliberately drops representativity in order to enforce a stronger form of non-redundancy: the symmetric difference of the support sets between two distinct Δ-closed sets of a dataset is at least of size Δ, i.e., the resulting pattern collection satisfies a stronger *distinctiveness condition* (see De Raedt and Zimmermann [2007]).

While constrained closedness and strong closedness differ in how they balance representativity and non-redundancy, there is also an important connection to the methodology of Chapter 3; namely we can again design an efficient listing algorithm via the identification of an inducing closure operator. We show that such an operator always exists, give an efficiently computable algorithmic definition of it, and integrate it into the divide-and-conquer fixpoint enumeration algorithm discussed in Chapter 3. Moreover, we empirically investigate different aspects of the performance of that algorithm for the task of listing all frequent strongly closed sets. These experiments also serve as practical application of Algorithm 5 from Chapter 3 where only a theoretical analysis is provided.

In the remainder of the chapter, we first precisely define the notion of Δ-closed sets and demonstrate that by using strongly closed sets, it is possible to arrive at semantically meaningful and stable result sets containing long patterns, without using support threshold parameters (Section 4.1). After that, we show that the family of strongly closed sets is always induced by a closure operator that can be computed efficiently (Section 4.2). We embed this closure operator in the divide-and-conquer fixpoint listing algorithm and investigate its computational properties (Section 4.3). Moreover, we provide an empirical evaluation of the use of strongly closed sets on real-world data (Section 4.4). Finally, we discuss similarities and differences of strong closedness to related concepts (Section 4.5) before giving a concluding summary and outlook (Section 4.6).

4.1. Strong Closedness

In this section, we first recap the definition of Δ-closedness formally and state some basic propositions that directly follow from that definition. Then we give empirical indications on two advanced properties: the family of strongly closed sets is robust against certain kinds of noise and/or dynamic changes in the input dataset and they can lead to result patterns of relatively high complexity (in the sense of size or area) when compared to families that result from minimum support constraints only.

4.1.1. Definition and Basic Properties

As mentioned in the beginning of this chapter, the notion of strong closedness emerges from the idea of quantifying the gap in support between a set and all of its supersets. Namely, the formal definition is as follows.

Definition 4.1 (Strongly Closed). Let $\mathcal{D}$ be a dataset over E. A set $F \subseteq E$ is called **strongly closed (or Δ-closed)** with respect to an integer $\Delta \geq 0$ if for all F' with $F \subset F' \subseteq E$ it holds that $|\mathcal{D}[F']| \leq |\mathcal{D}[F]| - \Delta$. $\Diamond$

That is, a set F is Δ-closed if any augmentation reduces its support by at least Δ. The name "Δ-closed" is justified by the fact that Δ measures the degree of closedness. In particular, for $\Delta = 1$ we obtain the definition of ordinary closed sets. For an integer support threshold[1] $t \geq 0$, the family of Δ-closed *frequent* sets is denoted by $\mathcal{C}_{\Delta,t,\mathcal{D}}$, i.e.,

$$\mathcal{C}_{\Delta,t,\mathcal{D}} = \{X \in \mathcal{F}_{t,\mathcal{D}} : X \text{ is } \Delta\text{-closed}\}$$

where $\mathcal{F}_{t,\mathcal{D}}$ is the family of t-frequent sets of a dataset $\mathcal{D}$, i.e., $\mathcal{F}_{t,\mathcal{D}} = \{F \subseteq E \colon q_{\text{supp}}(F) \geq t\}$. As usual, if $\mathcal{D}$ or t are clear from the context then they are omitted from the indices in $\mathcal{C}_{\Delta,t,\mathcal{D}}$ and $\mathcal{F}_{t,\mathcal{D}}$.

We now give an example illustrating Δ-closed sets. Consider the dataset $\mathcal{D}$ over $E = \{a, b, \ldots, f\}$ given in Figure 4.1. The set ae is 1-closed because $ae = \bigcap \mathcal{D}[ae]$. Since, in addition,

[1]While in previous chapters we used a relative *frequency* threshold to define frequent sets, in this chapter we switch to an integral *support* threshold for a more convenient interaction with the strength parameter Δ.

	a	b	c	d	e	f
1	0	1	1	0	1	1
2	0	0	1	1	1	1
3	1	0	0	0	1	1
4	1	1	0	0	1	1
5	1	1	1	1	0	1
6	1	1	1	0	0	0
7	1	1	1	1	0	0
8	1	1	1	1	1	0

Figure 4.1.: Example dataset $\mathcal{D}$ over ground set $E = \{a, \ldots, f\}$; the corresponding closed set family is $\mathcal{C}_1 = \{\emptyset, a, ab, abc, abcd, abcde, abcdf, abe, \ldots, E\}$, whereas the 2-closed and 3-closed set families are $\mathcal{C}_2 = \{\emptyset, abcd, ef, E\}$ and $\mathcal{C}_3 = \{\emptyset, E\}$, respectively.

$|\mathcal{D}[ae]| = 3$, we have $ae \in \mathcal{C}_{1,3}$. However, ae is not 2-closed because for its augmentation with b we have

$$2 = |\mathcal{D}[abe]| > |\mathcal{D}[ae]| - 2 = 1 \ .$$

In contrast, ef is a 2-closed set because all its augmentations reduce its support count of 4 by at least 2. Indeed, we can check that

$$\begin{array}{rcccccl} 2 &=& |\mathcal{D}[aef]| &\leq& |\mathcal{D}[ef]| - 2 &=& 2 \\ 2 &=& |\mathcal{D}[bef]| &\leq& |\mathcal{D}[ef]| - 2 &=& 2 \\ 2 &=& |\mathcal{D}[cef]| &\leq& |\mathcal{D}[ef]| - 2 &=& 2 \\ 1 &=& |\mathcal{D}[def]| &\leq& |\mathcal{D}[ef]| - 2 &=& 2 \ . \end{array}$$

The following proposition describes some direct implications of the definition of strong closedness.

Proposition 4.1. *Let $\mathcal{D}$ be a dataset over E. Then for all integers $\Delta, t \geq 0$ it holds that*

(i) the full ground set is always Δ-closed, i.e., $E \in \mathcal{C}_{\Delta,0}$,

(ii) the families of strongly closed t-frequent sets form a monotone increasing sequence bounded by the t-frequent sets for decreasing strength, i.e., $\mathcal{C}_{|\mathcal{D}|,t} \subseteq \mathcal{C}_{|\mathcal{D}|-1,t} \subseteq \ldots \subseteq \mathcal{C}_{0,t} = \mathcal{F}_t$,

(iii) all Δ-closed sets (except the full ground set) must also be Δ-frequent, i.e., $\forall F \subsetneq E$, $F \in \mathcal{C}_{\Delta,0} \Rightarrow F \in \mathcal{F}_\Delta$.

In particular, property (ii) tells us that we can use the strength parameter Δ analogue to the minimum support parameter t for successively reducing the output of (strongly) closed frequent set discovery. That said, if combining the two, values for t less or equal to Δ do not provide further refinement because by property (iii) Δ-closed sets (except E) must also be Δ-frequent.

In addition, we note that strongly closed sets do not only provide a quantification of closedness but also a quantification of non-redundancy in the following sense: the support sets of two distinct Δ-closed sets differs in at least Δ data records:

Proposition 4.2. *Let $\mathcal{D}$ be a dataset over E and $C, C' \in \mathcal{C}_\Delta$ with $C \neq C'$. Then it holds for the symmetric difference of the support sets of C and C' that $|\mathcal{D}[C] \ominus \mathcal{D}[C']| \geq \Delta$.*

Proof. Let C and C' be distinct Δ-closed sets in $\mathcal{D}$. Assume that $|\mathcal{D}[C] \ominus \mathcal{D}[C']| < \Delta$. Then in particular $|\mathcal{D}[C] \setminus \mathcal{D}[C']| < \Delta$. The case $C \subset C'$ directly contradicts the definition of Δ-closedness. Hence, it holds that $C \not\supseteq C'$. In this case we have $(C \cup C') \supset C$ and

$$\begin{aligned} \left|\mathcal{D}[C \cup C']\right| &= |\mathcal{D}[C]| - \left|\mathcal{D}[C] \setminus \mathcal{D}[C']\right| \\ &> |\mathcal{D}[C]| - \Delta \end{aligned}$$

contradicting the Δ-closedness of C as desired. □

Thus, as mentioned earlier, the parameter Δ can be used to enforce a high distinctiveness between the members of the result family $\mathcal{C}_\Delta$.

4.1.2. Stability

Now we turn to a perhaps less obvious feature of Δ-closed sets: they exhibit a certain robustness against changes in the input dataset. We first state a straightforward proposition interpreting the definition of strong closedness with respect to ordinary closedness.

Proposition 4.3. *Let Δ be a positive integer and $\mathcal{D}$ and $\mathcal{D}'$ be two datasets over the same ground set E such that $|\mathcal{D} \setminus \mathcal{D}'| < \Delta$. Then for all sets $C \in \mathcal{C}_{\Delta,\mathcal{D}}$ it holds that $C \in \mathcal{C}_{1,\mathcal{D}'}$.*

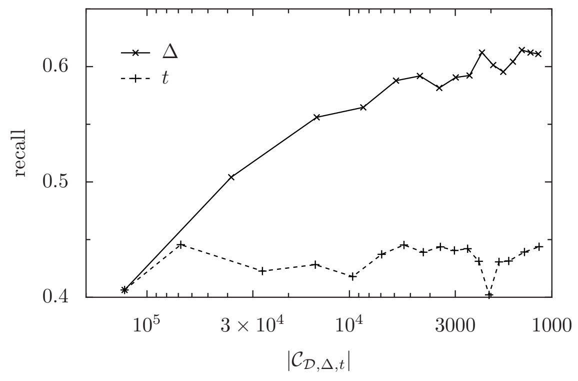
Proof. Let C be a Δ-closed set of $\mathcal{D}$. For $C = E$ the claim holds trivially. Now suppose $C \subset E$ and let $C' \subseteq E$ be an arbitrary strict superset of C. For a multiset of data records $\mathcal{R}$ over E we denote by $\mathcal{X}_\mathcal{R}$ the dataset $\mathcal{R}[C] \setminus \mathcal{R}[C']$, i.e., the data records of $\mathcal{R}$ that support C but not C'. We have

$$\begin{aligned} \left|\mathcal{D}'[C]\right| - \left|\mathcal{D}'[C']\right| &= |\mathcal{X}_{\mathcal{D}'}| \\ &= |\mathcal{X}_\mathcal{D}| - \left|\mathcal{X}_{\mathcal{D}\setminus\mathcal{D}'}\right| + \left|\mathcal{X}_{\mathcal{D}'\setminus\mathcal{D}}\right| \\ &\geq \underbrace{\left(|\mathcal{D}[C]| - \left|\mathcal{D}[C']\right|\right)}_{\geq\Delta} - \underbrace{\left|\mathcal{D} \setminus \mathcal{D}'\right|}_{<\Delta} + 0 \\ &\geq 1 \ . \end{aligned}$$

Hence, C is 1-closed with respect to $\mathcal{D}'$ as desired. □

That is, a set F that is Δ-closed in a dataset $\mathcal{D}$ remains closed (i.e., 1-closed) after deleting or changing any $\Delta - 1$ data records from $\mathcal{D}$. In this sense, the feature of F to be closed in $\mathcal{D}$ is *stable*. In addition, strong closedness itself possesses a similar stability, i.e., a Δ-closed set is likely to remain Δ-closed in face of changes of the dataset. In the following we present the result of an experiment with real-world Web data supporting this claim. It is important to note, though, that this experiment is only a statistically inspired motivation for the use of strongly closed sets rather than a thorough investigation of the statistical properties of Δ-closedness (and frequency).

We consider the dataset "Anonymous web data from www.microsoft.com" (msweb) from the UCI Machine Learning repository (see Section 4.4). It contains the areas of www.microsoft.com



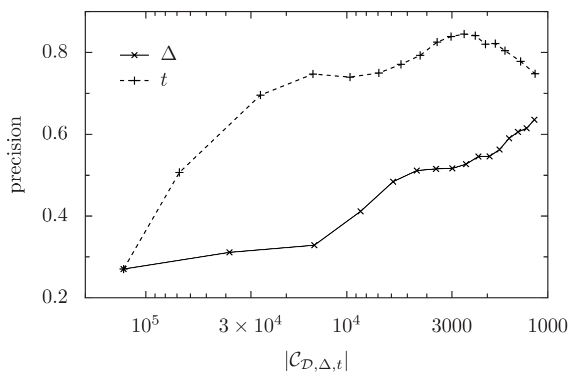


Figure 4.2.: Recall and precision of closed frequent set families respectively strongly closed set families plotted against their sizes. Note that x-axis is reversed, i.e., points from left to right correspond to shrinking families respectively growing threshold parameters.

in 1998 as ground set and roughly 32,000 data records, each of which representing one randomly selected user (visitor) of the website. A data record contains a web site if the corresponding user visited it during some fixed one week time frame in February 1998. Moreover, we investigate changes of the dataset that are generated by the following geometric perturbation procedure for individual data record:

1. Flip an unfair coin with some fixed success probability p.
2. Stop if the flip fails, otherwise flip a fair coin to determine whether to remove or to add an element from the data record. This element is chosen uniformly at random from the transaction respectively from its complement.
3. Go back to 1.

In the application example of discovering *cyber-communities* (see, e.g., Kumar et al. [1999]), where one is interested in Web communities that are implicitly defined by a group of users having common site preferences, the perturbation operation models the developing interests of a single user: a few new Web sites may attract her attention while she abandons interest in a few old ones. Furthermore, the expected data record sizes do not differ from the original data and thus, the dataset keeps its level of density. For the cyber-community example this means that a user's "capacity" is not expected to change drastically. A uniform perturbation operator that changes every bit with a certain probability would not satisfy these intuitions.

By applying this procedure to every data record with success probability $p = 0.9$ a dataset $\mathcal{D}'$ is created from the original msweb dataset $\mathcal{D}$. This success probability results in roughly nine changes on average per transaction. While this relatively high number of changes may be unrealistic in practice, it accentuates the differences between the families $\mathcal{C}_{\Delta,t,\mathcal{D}}$ and $\mathcal{C}_{\Delta,t,\mathcal{D}'}$. We are then interested in:

- the **recall** $r(\Delta, t) = \left|\mathcal{C}_{\Delta,t,\mathcal{D}} \cap \mathcal{C}_{\Delta,t,\mathcal{D}'}\right| / \left|\mathcal{C}_{\Delta,t,\mathcal{D}}\right|$, i.e., the fraction of Δ-closed t-frequents sets from $\mathcal{C}_{\Delta,t,\mathcal{D}}$ that "survived" the perturbation (appears in $\mathcal{C}_{\Delta,t,\mathcal{D}'}$),

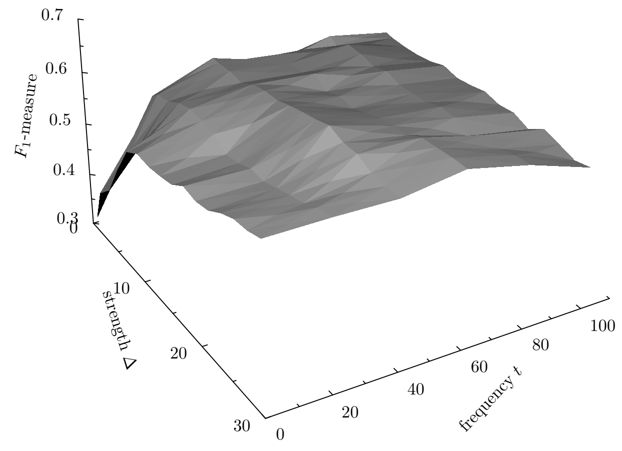


Figure 4.3.: F_1-measure achieved by the combination of minimum strength and minimum support.

- the **precision** $p(\Delta, t) = \left|\mathcal{C}_{\Delta,t,\mathcal{D}} \cap \mathcal{C}_{\Delta,t,\mathcal{D}'}\right| / \left|\mathcal{C}_{\Delta,t,\mathcal{D}'}\right|$, i.e., the fraction of strongly closed frequents sets from the perturbed dataset that were already strongly closed and frequent before the perturbation,

- and their combination, the F_1**-measure**, i.e., the harmonic mean of recall and precision.

In particular, we are interested in the recalls $r(\Delta, 0)$ and the precisions $p(\Delta, 0)$ for different strength parameters

$$\Delta \in \{1\} \cup \{2i\colon\ 1 \leq i \leq 15\}$$

compared to the recalls $r(1, t)$ and the precisions $p(1, t)$ that can be achieved by varying only the minimum support parameter

$$t \in \{5i\colon\ 0 \leq i \leq 10\} \ .$$

With these numbers we end up with closed set collections having sizes that roughly cover the possible orders of magnitude. The averaged results over two repetitions of this experiment are presented in Figure 4.2. It shows the recalls (left-hand plots) respectively the precisions (right-hand plots) against the cardinalities $|\mathcal{C}_{\Delta,t,\mathcal{D}}|$ using a reversed logarithmic scale. One can indeed observe better recalls for shrinking families corresponding to greater values of Δ and t. It is notable that the recalls do not increase anti-monotonically to the decreasing family sizes but show some fluctuations. Most importantly, for cardinalities $l \approx 30,000$ and lower we can always find values for Δ such that the resulting family $\mathcal{C}_{\Delta,1,\mathcal{D}}$ is roughly of cardinality l and has members that are more resistant to data changes when compared to the members of the family $\mathcal{C}_{1,t,\mathcal{D}}$ with cardinality closest to l. In contrast the precision was raised more effectively by the support threshold. This motivates the combination of strength and support thresholds in order to optimize the F_1-measure. Indeed, as depicted in Figure 4.3, the F_1-measure is maximized for $\Delta = 20$ and $t = 40$.

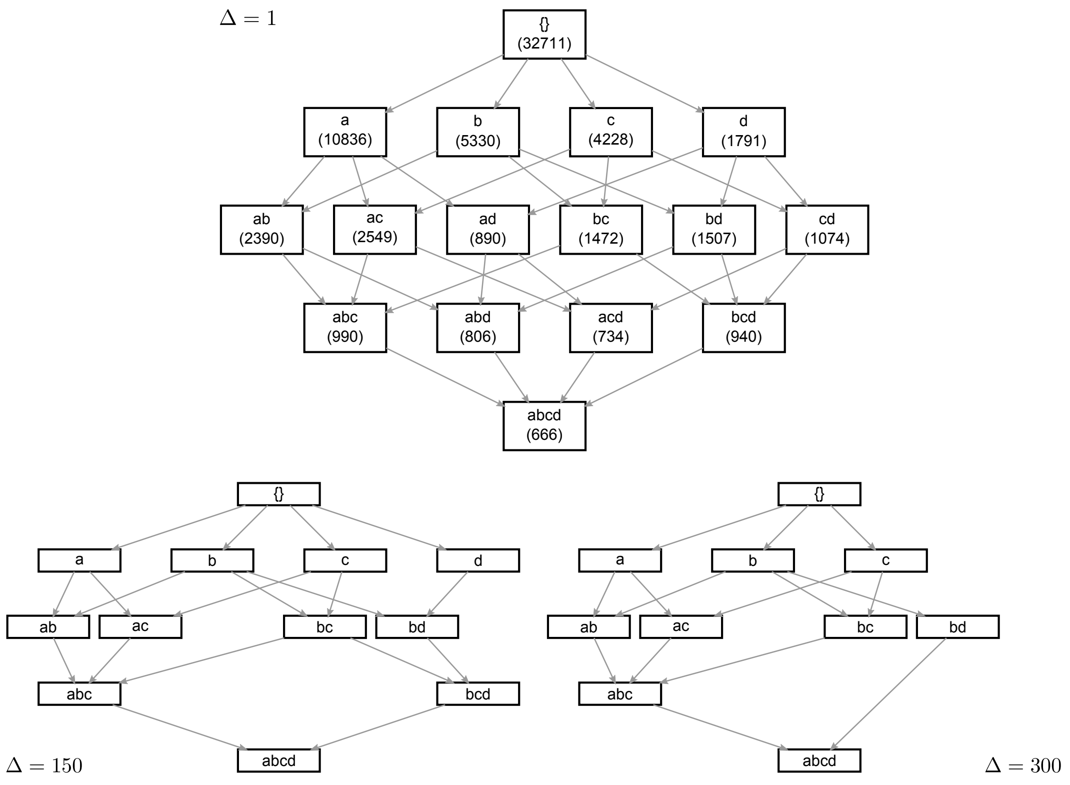


Figure 4.4.: Excerpts from strongly closed set lattices of "msweb" for different Δ.

4.1.3. Pattern Complexity

Another reason to consider strongly closed sets is that they often contain rather complex patterns even while there are only a small number of patterns in the output (i.e., for large values of Δ). If one enforces the same degree of output reduction using only a minimum support constraint, one tends to end up with less interesting or even only trivial result sets. This can be observed on two levels: for the individual result patterns as well as for the structure of the complete pattern family.

For the latter aspect, again the "msweb" dataset can serve as an illustration. Figure 4.4 shows a small excerpt from the Δ-closed set lattice that is successively reduced by increasing Δ. The semantic of the items is a: 'free downloads', b: 'isapi', c: 'Windows Family of OSs', and d: 'Win95 support'. The numbers in the first graph below the sets denote their support count. Note that *abcd* is a member of $\mathcal{C}_{300,1}$ while its subsets *d*, *ad*, *cd*, *abd*, *acd*, and *bcd* are not. Output families resulting only from a support threshold can never possess such a constellation because of the anti-monotonicity of support. In this sense, a strength threshold allows output families with less restrictions to emerge.

The complexity of an individual result pattern F can be measured for instance by its size $|F|$ or by its area $q_{\text{area}}(\mathcal{D}, F) = |F|\,|\mathcal{D}[F]|$. It turns out that the difference between closed frequent sets and strongly closed sets in this respect is connected to the density of the input dataset. For instance, consider the results for the datasets "retail" and "chess" (see Section 4.4) in Figure

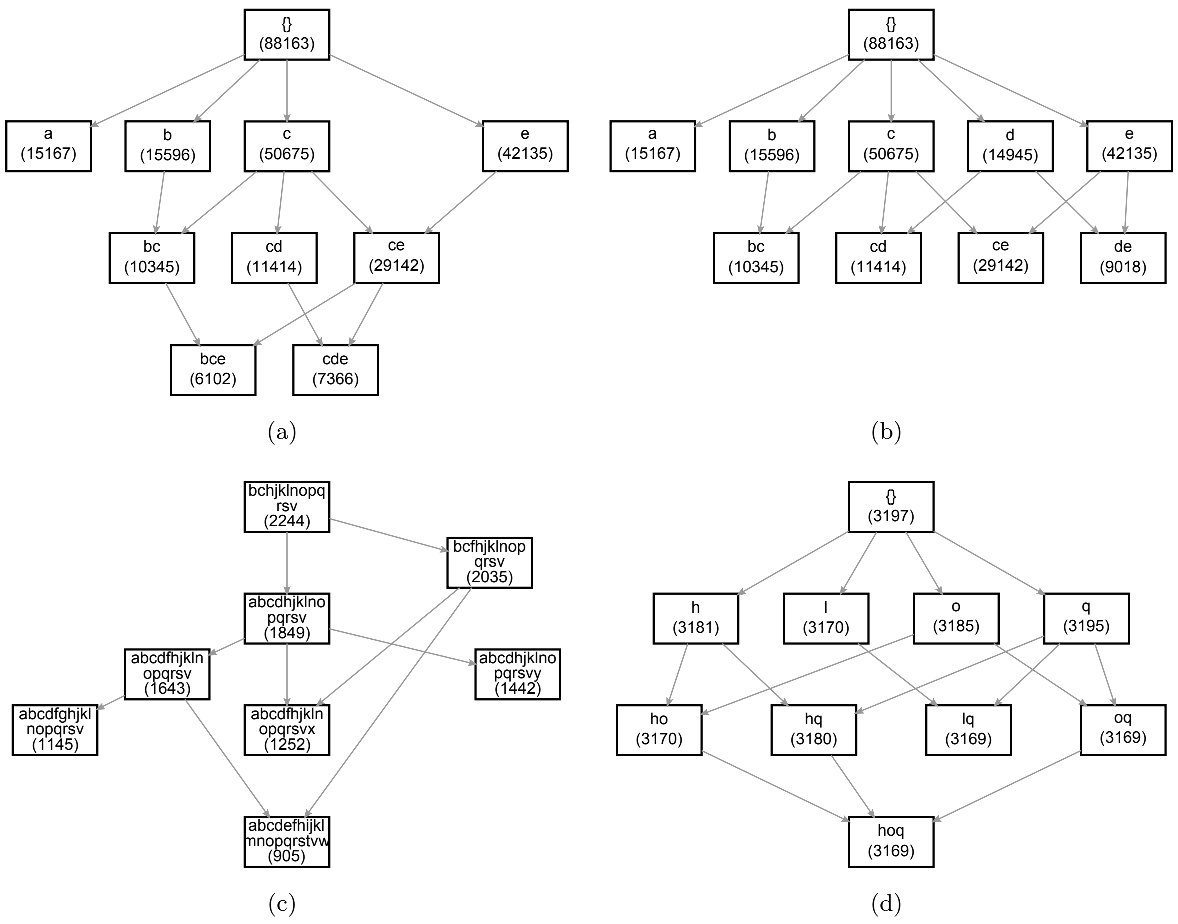


Figure 4.5.: The closed sets of highest frequency versus the 1-frequent sets of highest closedness of the datasets "retail" ((a) top closed, (b) top frequent) and "chess" ((c) top closed, (d) top frequent).

4.5. The first dataset contains retail market basket data from an anonymous Belgian retail supermarket store. It is very sparse and consists of a ground set of roughly 16,000 elements and of 88,000 data records. In the second dataset the data records are a collection of chess endgame positions and the elements of the ground set represent different chess pieces together with their board coordinates. Its main characteristic is that, while it contains only 75 items and 3179 data records, it is very dense and contains a huge number of closed sets.

The figures show top-10 frequent sets ((b) and (d)) and top-10 respectively top-8 strongly closed sets ((a) and (c)) for both datasets. For the sparse dataset "retail" there are only minor differences between the two closed set families. In contrast, they significantly differ for "chess": while the longest pattern among the ten most frequent 1-closed sets has only length three, there is a pattern of length 21 among the ten sets of strongest closedness. There is a similar behavior regarding the area: For the sparse dataset the set of greatest area is the empty set in both result families. In contrast, for the dense dataset the greatest area among the most frequent closed sets is 9507 whereas the greatest area among the most closed sets is 23072.

4.2. Inducing Closure Operator

Next we turn to an important structural property of strong closedness, namely that it is always, i.e., for all datasets, induced by an efficiently computable closure operator (recall that we refer to a family $\mathcal{C} \subseteq \mathcal{P}(E)$ as induced by a mapping $\rho \colon \mathcal{P}(E) \to \mathcal{P}(E)$ if $\mathcal{C} = \rho(\mathcal{P}(E)) = \{\rho(F) \colon F \subseteq E\}$). This property is important, because it allows us to reduce the problem of listing strongly closed sets to the problem of listing the fixpoints of the inducing operator—using, e.g., efficient Algorithms 2 and 5. Moreover, it implies a certain mathematical soundness that is for instance reflected in the fact that strong closedness can be combined with contraint-based closedness (see Sec. 4.6.1).

In the following, we give a two-step construction of a closure operator inducing the strongly closed sets of an input dataset $\mathcal{D}$. First we define a preclosure operator $\hat{\sigma}_\Delta$, i.e., an extensive and monotone mapping, and show that the fixpoints of $\hat{\sigma}_\Delta$ correspond to the family of strongly closed sets. In a second step we transform this preclosure operator in a straightforward way to a proper closure operator σ_Δ. We start with the definition of $\hat{\sigma}_\Delta$ that is based on augmenting a set by all single elements that reduce its support by less than Δ.

Definition 4.2 (Delta Augmentation)**.** Let $\mathcal{D}$ be a dataset over E and Δ a non-negative integer. Then the **delta augmentation** operator $\hat{\sigma}_\Delta : \mathcal{P}(E) \to \mathcal{P}(E)$ is defined by

$$\hat{\sigma}_\Delta : F \mapsto F \cup \{x \in E \setminus F \colon |\mathcal{D}[F \cup \{x\}]| > |\mathcal{D}[F]| - \Delta\} \ . \qquad \Diamond$$

This definition already possesses some of the desired properties: The delta augmentation operator is extensive per definition and it can be computed in time $O(\text{size}(\mathcal{D}[F]))$ by a simple iteration over all transactions in $\mathcal{D}[F]$, during which $|\mathcal{D}[F \cup \{x\}]|$ can be computed for all $x \in E \setminus F$. Moreover, its fixpoints are exactly the Δ-closed sets, i.e., it does indeed induce $\mathcal{C}_\Delta$. This is shown in the following lemma.

Lemma 4.4. *Let $\mathcal{D}$ be a dataset over E and Δ a non-negative integer. Then for all sets $F \subseteq E$ it holds that $F \in \mathcal{C}_\Delta \Leftrightarrow \hat{\sigma}_\Delta(F) = F$.*

Proof. Let F be a fixpoint of $\hat{\sigma}_\Delta$. Then

$$\forall x \in E \setminus F,\ |\mathcal{D}[F \cup \{x\}]| \leq |\mathcal{D}[F]| - \Delta \ . \tag{4.1}$$

Assume there is an $F' \supsetneq F$ with $|\mathcal{D}[F']| > |\mathcal{D}[F]| - \Delta$. Let $x \in F' \setminus F$. It follows by the anti-monotonicity of support that

$$|\mathcal{D}[F \cup \{x\}]| \geq \left|\mathcal{D}\left[F'\right]\right| > |\mathcal{D}[F]| - \Delta \ ,$$

which contradicts (4.1). Hence, F is Δ-closed.

Conversely, suppose $F \neq \hat{\sigma}_\Delta(F)$. Then, as $\hat{\sigma}_\Delta$ is extensive, there must be an $x \in \hat{\sigma}_\Delta(F) \setminus F$. Then $|\mathcal{D}[F \cup \{x\}]| > |\mathcal{D}[F]| - \Delta$ and thus $F \notin \mathcal{C}_\Delta$. $\square$

In general, however, the delta augmentation operator is not idempotent: the image of a set F under $\hat{\sigma}_\Delta$ can have a reduced support set such that more augmentation elements are within a support range of Δ for $\hat{\sigma}_\Delta(F)$ than for F. This can be observed in the example given in Figure 4.1, where for the set $\{a\}$ we have

$$\hat{\sigma}_2(a) = ab \neq abc = \hat{\sigma}_2(ab) = \hat{\sigma}_2(\hat{\sigma}_2(a)) \ .$$

Thus, $\hat{\sigma}_\Delta$ is not a closure operator. It is, however, a preclosure operator, i.e., extensive and monotone, as we show in the following lemma.

Lemma 4.5. *Let $\mathcal{D}$ be a dataset over E and Δ a non-negative integer. Then the map $\hat{\sigma}_\Delta$ is a preclosure operator on $\mathcal{P}(E)$.*

Proof. Extensivity is a direct implication of the definition. In order to show monotonicity let $F' \subseteq F \subseteq E$ and $x \in E$. Now suppose $x \notin \hat{\sigma}_\Delta(F)$, i.e.,

$$|\mathcal{D}[F]| - |\mathcal{D}[F \cup \{x\}]| \geq \Delta \ . \tag{4.2}$$

Then it follows for the support of $F' \cup \{x\}$ that

$$\begin{aligned}\left|\mathcal{D}\left[F' \cup \{x\}\right]\right| &= \left|\mathcal{D}\left[F'\right]\right| - \left|\mathcal{D}\left[F'\right] \setminus \mathcal{D}[\{x\}]\right| \\ &\leq \left|\mathcal{D}\left[F'\right]\right| - |\mathcal{D}[F] \setminus \mathcal{D}[\{x\}]| \\ &= \left|\mathcal{D}\left[F'\right]\right| - (|\mathcal{D}[F]| - |\mathcal{D}[F \cup \{x\}]|)\end{aligned}$$

and because of (4.2)

$$\leq \left|\mathcal{D}\left[F'\right]\right| - \Delta \ .$$

This implies that $x \notin \hat{\sigma}_\Delta(F')$ and consequently $\hat{\sigma}_\Delta(F') \subseteq \hat{\sigma}_\Delta(F)$. □

As $\hat{\sigma}_\Delta$ is a preclosure operator over a finite domain, iteration is a straightforward way of turning it into a closure operator: for all $F \subseteq E$ the sequence

$$F, \hat{\sigma}_\Delta(F), \hat{\sigma}_\Delta(\hat{\sigma}_\Delta(F)), \ldots$$

is monotonically increasing (with respect to the subset relation) and bounded by E and, hence, has a fixpoint. Thus, assigning this fixpoint to F is a well-defined operation giving rise to the following definition.

Definition 4.3 (Iterated Delta Augmentation)**.** Let $\mathcal{D}$ be a dataset over E and Δ a non-negative integer. Then the **iterated delta augmentation** operator $\sigma_\Delta : \mathcal{P}(E) \to \mathcal{P}(E)$ is defined by

$$\begin{aligned}\hat{\sigma}^1_\Delta &: F \mapsto \hat{\sigma}_\Delta(F) \\ \hat{\sigma}^{i+1}_\Delta &: F \mapsto \hat{\sigma}_\Delta(\hat{\sigma}^i_\Delta(F)) \\ \sigma_\Delta &: F \mapsto \hat{\sigma}^k_\Delta(F),\ k = \min\{i\colon\ \hat{\sigma}^i_\Delta(F) = \hat{\sigma}^{i+1}_\Delta(F)\} \ .\end{aligned}$$ ◊

Clearly, σ_Δ is idempotent. Moreover, it is easy to see that it inherits extensivity, monotonicity, and the fixpoints from $\hat{\sigma}_\Delta$. Hence, the iterated delta augmentation is the desired closure operator inducing the strongly closed sets, i.e., we have the following theorem.

Theorem 4.6. *Let $\mathcal{D}$ be a dataset over E and Δ a non-negative integer. Then σ_Δ is a closure operator that generates C_Δ, i.e., for all $F \subseteq E$ it holds that $F \in C_\Delta \Leftrightarrow F = \sigma_\Delta(F)$.*

Moreover, the iterated delta augmentation is efficiently computable. Note that for a given $F \subseteq E$, the value k of the closure $\sigma_\Delta(F)$ as defined in Definition 4.3 is bounded by $|E \setminus F|$. This is because $\sigma_\Delta(F) \subseteq E$ and in every iteration $i < k$, at least one element is added, i.e., $\left|\hat{\sigma}^i_\Delta(F) \setminus \hat{\sigma}^{i-1}_\Delta(F)\right| \geq 1$. As noted above, $\hat{\sigma}_\Delta(F)$ can be computed in time $\text{size}(\mathcal{D}[F])$, so computing σ_Δ by an iterative application of $\hat{\sigma}_\Delta$ has time complexity

$$O(k\,\text{size}(\mathcal{D}[F])) = O(|E \setminus F|\,\text{size}(\mathcal{D}[F])) \quad .$$

We call the iteration length k the **stair number** of F, because intuitively the columns and rows of the matrix representing $\mathcal{D}$ can be reordered such that the rows corresponding to $\mathcal{D}[F]$ and the columns corresponding to $\sigma_\Delta(F)$ visually form a "staircase" of 1-entries with k stairs. Advantages and disadvantages of this trivial implementation of σ_Δ and other algorithmic issues are discussed in the next section.

4.3. Listing Frequent Strongly Closed Sets

Having discussed structural properties of strong closedness, we are now ready to investigate its algorithmic side. As mentioned in the beginning of this chapter (and motivated further in Section 4.1.2), we aim to combine strong closedness with minimum support as closed set selection criteria. Thus, our goal is to design an algorithm that efficiently solves the following problem.

Problem 4.1 (LIST-STRONGLY-CLOSED-FREQUENT-SETS)**.** *Given* a dataset $\mathcal{D}$ over E, strength threshold Δ, and support threshold t, *list* the family of strongly closed frequent sets $\mathcal{C}_{\Delta,t,\mathcal{D}}$. $\Diamond$

We achieve this by using the iterated delta augmentation σ_Δ within a slightly modified version of the divide-and-conquer fixpoint listing algorithm from Chapter 3. That is, we make use of the fact that the iterated delta augmentation is a closure operator inducing the strongly closed sets.

4.3.1. Closure Computation

A central component of the fixpoint enumeration approach is an efficient algorithmic realization of the corresponding closure operator. In the following, we discuss two alternative approaches to this: one direct implementation that almost explicitly iterates over the preclosure operator $\hat{\sigma}_\Delta$ and one more involved that deliberately leaves this path.

We start with the direct approach. As stated in Section 4.2, the closure $\sigma_\Delta(F)$ of a set $F \subseteq E$ with stair number k can be computed in time $O(k\,\text{size}(\mathcal{D}[F]))$ by iteratively computing the preclosure $\hat{\sigma}_\Delta$. Algorithm 7 almost exactly implements this naive strategy with the difference that it scans the dataset column-wise instead of row-wise. By using a dataset representation

Algorithm 7 Direct Iterated Delta Augmentation

```
Input   : set $F \subseteq E$, integer $\Delta > 0$, a dataset $\mathcal{D}$ over $E$
Output : the iterated delta augmentation $\sigma_\Delta(F)$

1. $C \leftarrow F$, $\mathcal{D}' \leftarrow \mathcal{D}[F]$
2. repeat
3.   for all $e \in E \setminus C$ do
4.     if $|\bar{\mathcal{D}}'[\{e\}]| < \Delta$ then
5.       $C \leftarrow C \cup \{e\}$; $\mathcal{D}' \leftarrow \mathcal{D}'[\{e\}]$
6. until $\mathcal{D}'$ was not changed during step 5
7. return $C$
```

of item incidence lists, this modification does not modify the worst-case computation time. On the other hand, it can lead to practical speed-ups because more than one iteration of the preclosure operator can potentially be computed during a single iteration of the outer loop.

Indeed, for the real-world datasets used in the experiments presented in Section 4.4 below, Algorithm 7 turns out to be very effective. The reason is that, for these datasets, the number of iterations of the outer loop is very small (in average close to 1) for the vast majority of closure computations. However, it is easy to construct worst-case examples for which the number of iterations is in $\Omega(|E|)$ and thus the overall complexity becomes $\Theta(|E \setminus F| \operatorname{size}(\mathcal{D}[F]))$, i.e., the worst-case bound is sharp. For instance, we can construct for any positive integer n a dataset $\mathcal{D}_n = \{D_1, \ldots, D_{n+1}\}$ over ground set $\{e_1, \ldots, e_n\}$ with $D_i = \{e_i, \ldots, e_n\}$ for $1 \leq i \leq n$ and with $D_{n+1} = \emptyset$. This results in a dataset with a binary matrix representation of the following form:

$$\begin{array}{cccc}
1 & \ldots & 1 & 1 \\
0 & 1 & \ldots & 1 \\
\vdots & \ddots & \ddots & \vdots \\
0 & \ldots & 0 & 1 \\
0 & 0 & \ldots & 0
\end{array}$$

For $\mathcal{D}_n$, $\Delta = 1$, and $F = \emptyset$, Algorithm 7 will only select one element per iteration, namely e_{n-i+1} in iteration i. Until the solution $\sigma_\Delta(\emptyset) = E$ is completed, the algorithm will run through n iterations where iteration i takes time $(n-i)(|\mathcal{D}_n| - i)$. Thus, the overall running time for these examples is in $\Theta(n^3)$.

We now present an alternative algorithm for computing σ_Δ that has a better worst-case time bound—at least for non-sparse datasets. This can be achieved by avoiding the explicit iteration of the preclosure operator (for which a linear time complexity $|\mathcal{D}[F]|$ appears to be inherent). The key idea of this algorithm is to regard the input dataset as *negated adjacency matrix* of a bipartite graph and to compute the closure via a modified depth-first traversal of that graph. Note that, in order to compute the closure of a set $F \subseteq E$, only the information of the reduced dataset $\mathcal{D}[F]$ projected on the items $E \setminus F$ is required. Thus, we consider the bipartite graph $G_{\mathcal{D}[F]}$ with the data records $\mathcal{D}[F]$ as vertices in one part of the bipartition and

Algorithm 8 Iterated Delta Augmentation by Zero-Elimination

```
Input  : subset $F \subseteq E$,
         positive integer $\Delta$,
         dataset $\mathcal{D}$ over $E$ restricted to $\mathcal{D}[F] \times (E \setminus F)$
         given by incidence list complements
         $\bar{D} = \{e \in E \setminus F \colon e \notin D\}$, for all $D \in \mathcal{D}[F]$
Output : $C$ equal to $\sigma_\Delta(F)$

deleteColumn($e$):
1. $C \leftarrow C \cup \{e\}$
2. for all $D \in L_e$ do
3.    if $D \notin X$ then deleteRow($D$)

deleteRow($D$):
1. $X \leftarrow X \cup \{D\}$
2. for all $e \in \bar{D}$ do
3.    $\Sigma(e) \leftarrow \Sigma(e) - 1$
4.    if ($\Sigma(e) < \Delta$ and $e \notin C$) then deleteColumn($e$)

main:
1. $C \leftarrow F$, $X \leftarrow \emptyset$
2. for all $e \in E \setminus F$ do
3.    $L_e \leftarrow \{D \in \mathcal{D}[F] \colon e \notin D\}$
4.    $\Sigma(e) \leftarrow |L_e|$
5. for all $e \in E \setminus F$ do
6.    if ($\Sigma(e) < \Delta$ and $e \notin C$) then deleteColumn($e$)
7. return $C$
```

the elements of $E \setminus F$ in the other. Moreover, as edges we regard the pairs

$$Z = \{\{e, D\} \colon e \notin D, e \in E \setminus F, D \in \mathcal{D}[F]\} \ .$$

That is, there is an edge for every *zero* in the binary matrix representation of $\mathcal{D}[F]$. Now, in order to compute the closure $\sigma_\Delta(F)$, consider the following modified depth-first traversal of $G_{\mathcal{D}[F]}$:

1. initialize $C \leftarrow F$
2. for all vertices $e \in (E \setminus F)$ having degree less than Δ start a traversal of $G_{\mathcal{D}[F]}$ as follows:
   - if the current vertex is an item $e \in E$, add e to C, delete all edges incident to e, and visit all vertices that have been neighbors of e prior to this deletion
   - if the current vertex is a data record $D \in \mathcal{D}$, delete all edges incident to D, and visit all vertices that have been neighbors of D prior to this deletion *and* that now have degree less than Δ
3. return C

Since this algorithm successively deletes edges, and edges correspond to zeros in the binary matrix representation of $\mathcal{D}$, we call it "zero-elimination".

In order to analyze the complexity of this approach, let $\overline{\text{size}}(\mathcal{D}) = \sum_{D \in \mathcal{D}} |E \setminus D|$ denote the sum of all data record complements of a dataset $\mathcal{D}$, i.e., the number of zero entries in the binary matrix representation of $\mathcal{D}$. Clearly, the size of the graph $G_{\mathcal{D}[F]}$, i.e., the sum of the number of its vertices and edges, is bounded by $2\,\overline{\text{size}}(\mathcal{D}[F])$, and $G_{\mathcal{D}[F]}$ can also be constructed in this time given $\mathcal{D}[F]$. Since a depth-first traversal of a graph G can be implemented to run in time linear in the size of G, zero-elimination has a worst-case complexity of $O\big(\overline{\text{size}}(\mathcal{D}[F])\big)$.

A more realistic in-place formulation of this algorithm is given in Algorithm 8. For a proof of its correctness and its time complexity—and thus of the following theorem—we refer to Appendix A.2.

Theorem 4.7. *Let $\mathcal{D}$ be a dataset over E. Given $F \subseteq E$, $\mathcal{D}[F]$ represented by the complements of its data records, and a non-negative integer Δ, the closure $\sigma_\Delta(F)$ can be computed with time and space $O(\overline{\text{size}}(\mathcal{D}[F]))$.*

For non-sparse datasets the worst-case complexity of the zero-elimination algorithm is better than that of Algorithm 7 by a factor k. In practice, however, we observed that Algorithm 7 strictly outperformed the zero-elimination algorithm. This has at least three reasons: (i) most benchmark datasets are relatively sparse, i.e., $\text{size}(\mathcal{D}) \ll \overline{\text{size}}(\mathcal{D})$, (ii) the density of the subdatasets for which closures have to be computed even decreases, and (iii) k is on average very small (see Section 4.4). For that reason we use Algorithm 7 for closure computations in the subsequent discussion.

4.3.2. Integration into Fixpoint Listing

Based on efficient closure computations we can now adapt the divide-and-conquer fixpoint listing algorithm from Chapter 3 for the task of listing all frequent strongly closed sets given an input dataset $\mathcal{D}$ along with minimum strength and minimum support parameters Δ and t. To recap this approach in a nutshell, it recursively lists, given some fixpoint C of a closure operator σ, all fixpoints that are a superset of $C \cup \{a\}$ for some valid augmentation element a, and then all fixpoints that are supersets of C but that do not contain a. Moreover, Theorem 3.3 tells us that the algorithm is correct even for partially defined closure operators $\sigma \colon \mathcal{F} \to \mathcal{F}$ if the domain $\mathcal{F}$ is at least strongly accessible.

Although the family of t-frequent sets $\mathcal{F}_t$ is even an independence system, which implies strong accessibility, the inducing closure operator σ_Δ is not a closed operation on $\mathcal{F}_t$: there can be frequent sets $F \in \mathcal{F}_t$ with $\sigma_\Delta(F) \notin \mathcal{F}_t$. For such cases we need a slight modification of the algorithm that avoids printing and refining infrequent sets. Namely, let $C \subseteq E$ be the last frequent strongly closed set computed by the algorithm and $B \subseteq E$ the set of previous augmentation elements that must be disjoint with every set derived from C. Then, the modified algorithm works as follows:

1. select an element a from $E \setminus (C \cup B)$ such that $C \cup \{a\}$ is frequent if such an element exists; otherwise return
2. compute the closure $C' = \sigma_\Delta(C \cup \{a\})$
3. if C' is disjoint with B *and frequent* then print C' and list recursively the frequent strongly closed supersets of C', which are disjoint with B
4. lists recursively the frequent strongly closed supersets of C that are disjoint with $B \cup \{a\}$.

This essentially applies the original algorithm to the extended domain $\mathcal{F}_t \cup \sigma_\Delta(\mathcal{F}_t)$ and prunes away the infrequent branches. Given that the base algorithm correctly lists the fixpoints of σ_Δ, it is easy to check that the modified version correctly lists $\mathcal{C}_{\Delta,t,\mathcal{D}}$: only frequent fixpoints can be listed because of the check in step 3, and the pruning does not miss any frequent fixpoints because of the anti-monotonicity of the minimum support constraint. Also the non-redundancy is inherited from the base algorithm, i.e., no set is printed more than once.

The only aspect that may appear problematic is the time complexity because, in contrast to the base algorithm, the modified algorithm does not print all sets it generates (recall that we measure efficiency in terms of the relation between computation time and the amount of output produced). A closer look at the proof of Theorem 3.5, however, reveals that our modification does not affect the time complexity. The important observation in that proof is that there can be at most n "failed" augmentations before the algorithm backtracks. A failed augmentation means a closure computation $C' = \sigma(C \cup \{a\})$ such that $C' \cap B \neq \emptyset$, i.e., C' was redundantly generated again. While in the original algorithm this is the only reason for a failure, in our modified case there is the additional reason of ending up in an infrequent set. However, there are no more that n augmentation elements that can lead to a failed closure computation—whether due to a redundant enumeration or due to infrequency.

Thus, all guarantees of Theorem 3.5 also apply to the problem of listing frequent strongly closed sets. Namely, this problem can be solved with delay $O(|E|\ (T_{\mathcal{F}} + T_\rho + |E|))$ and space $O(|E| + S_{\mathcal{F}} + S_\rho)$ where $T_{\mathcal{F}}$, T_ρ, $S_{\mathcal{F}}$, and S_ρ denote the maximum time and space complexities of checking minimum support and computing a closure, respectively. The support of a set can easily be counted in time $O(\text{size}(\mathcal{D}))$ (respectively in time $O\big(\overline{\text{size}}(\mathcal{D})\big)$ depending on the representation of $\mathcal{D}$). Combining this with the complexity of the zero-elimination algorithm given in Theorem 4.7 respectively the complexity of Algorithm 7 we have the following result.

Theorem 4.8. *Given a dataset $\mathcal{D}$ over E and thresholds Δ, t, the family $\mathcal{C}_{\mathcal{D},\Delta,t}$ of Δ-closed t-frequent sets can be listed with delay*

$$O(|E| \min\{\overline{\text{size}}(\mathcal{D}),\ |E| \operatorname{size}(\mathcal{D})\}) \ ,$$

corresponding total time

$$O(|E|\ |\mathcal{C}_{\Delta,t,\mathcal{D}}| \min\{\overline{\text{size}}(\mathcal{D}),\ |E| \operatorname{size}(\mathcal{D})\}) \ ,$$

and space

$$O(|E| + \min\{\overline{\text{size}}(\mathcal{D}),\ \operatorname{size}(\mathcal{D})\}) \ .$$

Note that the bound on the delay only holds for a modified version of the algorithm that artificially postpones the printing of some already computed sets in order to achieve more evenly distributed output times (see Section 3.3.2). The bound on the total time, however, holds right away for the algorithm presented in this section.

For datasets that are neither sparse nor dense, i.e., datasets $\mathcal{D}$ with $\overline{\text{size}}(\mathcal{D}) \approx \operatorname{size}(\mathcal{D})$, the total time bound above boils down to $O(|E|\,|\mathcal{C}_{\mathcal{D},\Delta,t}| \operatorname{size}(\mathcal{D}))$, which is equal to the best known theoretical time bound of a closed frequent set listing algorithm (see Section 3.5). This is a notable observation as the family of Δ-closed (frequent) sets is a generalization of the usual closed (frequent) sets. Whether this bound can also be achieved for sparse datasets is an open question.

4.3.3. Implementation and Speedups

Our approach for listing frequent strongly closed sets is a reduction to the problem of listing fixpoints of a closure operator. However, in contrast to Chapter 3, here we have a concrete closure operator that we do not need to treat like a black box. This allows us to discuss some implementation aspects and practical speedups in more detail. Although these additions do not affect the theoretical worst-case bounds stated in Theorem 4.8, they ensure the algorithm's factual applicability in many real-world scenarios (with common computation time budgets like 48 hours). A summarizing pseudo code is given in Algorithm 9 (p. 75). In the remainder of this section we describe all details (including the $\uparrow_\Delta$ operation used in the last line).

The enumeration scheme as well as the closure computation rely on computing the support set of a given set $F \subseteq E$ augmented by a single item $e \in E \setminus F$. That is, they have to compute intersections of the form

$$\mathcal{D}[F] \cap \mathcal{D}[e] = \mathcal{D}[F \cup \{e\}] \ .$$

This operation is best supported by representing the dataset via the *incidence lists of the individual items*, i.e., for each item $e \in E$ a list containing the indices of all data records containing e. A first speedup is *successive dataset reduction* that is proposed by most research papers on closed set enumeration in one way or another (e.g., Gély [2005], Uno et al. [2004], Wang et al. [2005a]). It is based on the idea that all recursive listing tasks below a Δ-closed set C can be performed based on the information of the reduced dataset $\mathcal{D}[C]$. Thus, the

Algorithm 9 Strongly Closed Frequent Set Listing

Input : dataset $\mathcal{D}$ over E, strength parameter Δ, and support threshold t
Output: family $\mathcal{C}_{\Delta,t,\mathcal{D}}$ of Δ-closed t-frequent sets

```
main:
1. global variables $C \leftarrow \emptyset$, $\mathcal{D}' \leftarrow \mathcal{D}$
2. closure($\emptyset$)
3. if $|\mathcal{D}'| \geq t$ then
4.   print $C$
5.   list($\emptyset$)

closure($B$):
1. repeat
2.   for all $e \in E \setminus C$ do
3.     if $|\bar{\mathcal{D}}'[\{e\}]| < \Delta$ then
4.       if $e \in B \vee |\mathcal{D}'[\{e\}]| < t$ then return
5.       $C \leftarrow C \cup \{e\}$; $\mathcal{D}' \leftarrow \mathcal{D}'[\{e\}]$
6. until $\mathcal{D}'$ was not changed during step 5

list($B$):
1. let $e \in E \setminus (C \cup B)$ with $|\mathcal{D}'[\{e\}]| \geq t$ or return if no such $e$ exists
2. $C \leftarrow C \cup \{e\}$, $\mathcal{D}' \leftarrow \mathcal{D}'[\{e\}]$
3. closure($B$)
4. if $|\mathcal{D}'| \geq t$ and $C \cap B = \emptyset$ then
5.   print $C$
6.   list($B$)
7. undo all changes done to $C$ and $\mathcal{D}'$ since line 3
8. list($B \cup \uparrow_\Delta (e, \mathcal{D}')$)
```

incidence lists of all items $e \in E \setminus C$ are reduced to incidence lists with respect to the dataset $\mathcal{D}' = \mathcal{D}[C]$. The required support sets $\mathcal{D}[C \cup \{e\}]$ are then directly available as $\mathcal{D}'[e]$. While this method requires an additional pass over the reduced data after every closure computation, it does not modify the overall theoretical time bound and is particularly worthwhile in the presence of many data records and correspondingly long item incidence lists. In order to reduce the memory footprint of this method we avoid creating (reduced) local copies of the input dataset for every recursive call. Instead global variables $C, \mathcal{D}'$ are used that satisfy the invariant $\mathcal{D}' = \mathcal{D}[C]$ throughout the execution of the algorithm. When C is augmented (lines 2 of LIST resp. 5 of CLOSURE), $\mathcal{D}'$ is reduced accordingly and deleted elements are stored on a stack. When the algorithm subsequently backtracks this stack is used to restore the dataset. While these stack operations only cause little time overhead, they preserve the theoretical space bound. They are also necessary for the practical performance. Otherwise the algorithm would have prohibitive memory requirements already for mid-sized datasets with many items.

Another simple yet powerful speedup is to insert an *early abort condition into the closure* *computation.* Note that a computed closure is “rejected” (neither printed nor further refined) in line 4 of LIST if it is either infrequent or contains an element of B. Since the closure is computed iteratively by augmenting the current set with one new element in each step (see loop 3–5 in Algorithm 7), the closure computation algorithm can immediately be aborted if either an element of B or an infrequent element with respect to the current reduced dataset has been added to the closure. This condition is checked in line 4 of CLOSURE.

Finally one can deduce from the algorithm’s total time bound that every Δ-closed set can be generated at most $|E|$ times. Although this factor is linear, it may be prohibitive especially in the presence of a large number of items and can be further improved for many problem inputs. One idea to reduce the number of redundant generations is to *extend the set B of blocked items* not only by e but also by all items e' that “imply” e in the current reduced dataset. For this purpose we can adapt the uparrow operator (“$\uparrow$”) used in Gély [2005] such that it takes into account the strength threshold Δ by defining a map $\uparrow_\Delta\colon E \times \mathcal{P}(\mathcal{D}) \to \mathcal{P}(E)$ with

$$\uparrow_\Delta\colon (e, \mathcal{D}') \mapsto \{e' \in E\colon \left|\mathcal{D}'[\{e'\}] \setminus \mathcal{D}'[\{e\}]\right| < \Delta\} \ .$$

With this definition we know that if $e' \in \uparrow_\Delta (e, \mathcal{D}[F])$ then

$$e \in \hat{\sigma}_\Delta(F' \cup \{e'\}) \subseteq \sigma_\Delta(F' \cup \{e'\})$$

for all F' with $F \subseteq F' \subseteq E$. Consequently step 4 of the enumeration scheme can be rewritten as

4’. lists recursively the closed supersets of C that are disjoint with $B \cup \uparrow_\Delta (a, \mathcal{D}[C])$

reflected by line 8 within LIST of Algorithm 9. On the one hand, the computation of $\uparrow_\Delta$ requires one pass over the (reduced) data and in the worst case it can happen that $\uparrow_\Delta (e, \mathcal{D}') = \{e\}$, i.e., no redundant closure computation is prevented. On the other hand, again this additional pass is already accounted for in the asymptotic time bound stated in Theorem 4.8, and in practice this modification avoids many redundant closure computations (see Section 4.4).

4.4. Performance Experiments

name	accidents	chess	kosarak	msweb	mushroom	pumsb	retail	T10l4D100K
$\lvert E\rvert$	468	75	41,270	285	119	7,116	16,469	1,000
$\lvert\mathcal{D}\rvert$	340,183	3,196	990,002	32,711	8,124	49,046	88,162	100,000
density	0.07224	0.49333	0.0002	0.01058	0.19325	0.0104	0.00063	0.0101
origin	1,2	2,3	2	3,4	2,3	2,3	2,5	2

Table 4.1.: Datasets used in experiments. Origin gives data repositories and references in which the respective dataset can be found respectively is referred to (1 - [Geurts et al., 2003], 2 - [Goethals and Zaki, 2004], 3 - [Asuncion and Newman, 2007], 4 - [Breese et al., 1998], 5 - [Brijs et al., 1999]).

In this section, we review experimental results of extracting strongly closed sets from publicly available real-world and synthetic datasets using Algorithm 9. On the one hand, they verify that the idea of strong closedness is practically useful for reducing output and computation time of closed set listing. On the other hand, the experiments also constitute an empirical evaluation of the general divide-and-conquer fixpoint listing approach. In particular, the goals of this study are

a) to show the influence of the strength parameter on the number of strongly closed sets, i.e., to demonstrate that this parameter indeed provides an effective approach to output reduction, and

b) to show the algorithm's efficiency with respect to the output size, i.e., that its computation time indeed scales linearly in the number of strongly closed sets.

In contrast, it is not meant to outperform the highly elaborated frequent closed set listing algorithms for the special case of $\Delta = 1$. While the employed C++ implementation[2] of Algorithm 9 uses some practical speedups (described in Section 4.3.3), it does not include the various pruning and database reduction techniques that can be applied in the context of frequency.

The experiment consists of generating the strongly closed sets of the datasets listed in Table 4.1 for different values of Δ. Beside ground set size ($|E|$), number of data records ($|\mathcal{D}|$), and origin, the table also contains the **density** $\text{size}(\mathcal{D})/(|E|\ |\mathcal{D}|)$ of each dataset, i.e., the fraction of one-entries in their binary matrix representation. The parameter Δ in the definition of strong closedness is an absolute value that is insensitive to the number of data records. Hence, in order to establish comparability between the performances for different datasets, it is advisable to use different Δ-values for the different datasets relative to their size. Our experiments use eleven different values for each dataset $\mathcal{D}$, namely $\Delta = \lceil \tilde{\Delta}\, |\mathcal{D}| \rceil$ with

$$\tilde{\Delta} \in \{(0.001 + 0.005i) \colon\ 0 \le i \le 10\} \ .$$

	accidents		chess		kosarak		msweb		mushroom		pumsb		retail		T10I4D100K	
$\tilde{\Delta}$	$\lvert\mathcal{C}_\Delta\rvert$	T	$\lvert\mathcal{C}_\Delta\rvert$	T	$\lvert\mathcal{C}_\Delta\rvert$	T	$\lvert\mathcal{C}_\Delta\rvert$	T	$\lvert\mathcal{C}_\Delta\rvert$	T	$\lvert\mathcal{C}_\Delta\rvert$	T	$\lvert\mathcal{C}_\Delta\rvert$	T	$\lvert\mathcal{C}_\Delta\rvert$	T
0.051	296	75	40	1	8	2	11	1	126	1	12	1	8	1	438	14
0.046	552	124	78	1	10	2	12	1	156	1	19	2	10	1	476	15
0.041	996	194	161	1	10	2	13	1	191	1	28	3	12	1	510	16
0.036	1809	313	344	1	11	3	13	1	226	1	57	4	13	1	569	17
0.031	3354	522	882	2	12	4	15	1	387	1	98	7	14	1	652	17
0.026	6510	922	2757	5	16	4	22	1	677	2	256	18	15	1	773	19
0.021	13K	29m	12K	17	18	4	27	1	1113	2	818	57	15	1	955	20
0.016	32K	70m	58K	66	26	5	42	1	1935	2	5941	351	20	1	1339	21
0.011	95K	3.7h	350K	301	42	9	60	1	4362	5	95K	77m	29	1	2241	24
0.006	481K	18h	2,7M	26m	88	17	121	1	9470	7	143M	38h	69	2	5929	29
0.001			143M	10h	1K	194	1K	1	76K	16			2K	37	76K	86

Table 4.2.: Number of strongly closed sets $|\mathcal{C}_\Delta|$ and corresponding computation time T (given in seconds unless stated otherwise and with large values rounded).

[2]The implementation is available for download at the author's homepage.

The results are presented in Table 4.2. It contains the total computation time[3] (T) and the corresponding number of strongly closed sets ($|\mathcal{C}_\Delta|$) for all combinations of parameter values and datasets. Times below one second are rounded to one. Other numbers are rounded to the nearest integer. For all datasets we can observe a sharp drop of the number of strongly closed sets with growing strength threshold. Moreover, the running time of the algorithm is linear or in fact mostly sublinear in the number of sets produced, i.e., the computation time per closed set tends to decrease with an increasing number of closed sets (or correspondingly with decreasing values for Δ). Figure 4.6(a) graphically shows the scaling relative to the output size for four datasets on a log-vs-log-scale compared to the identity map.

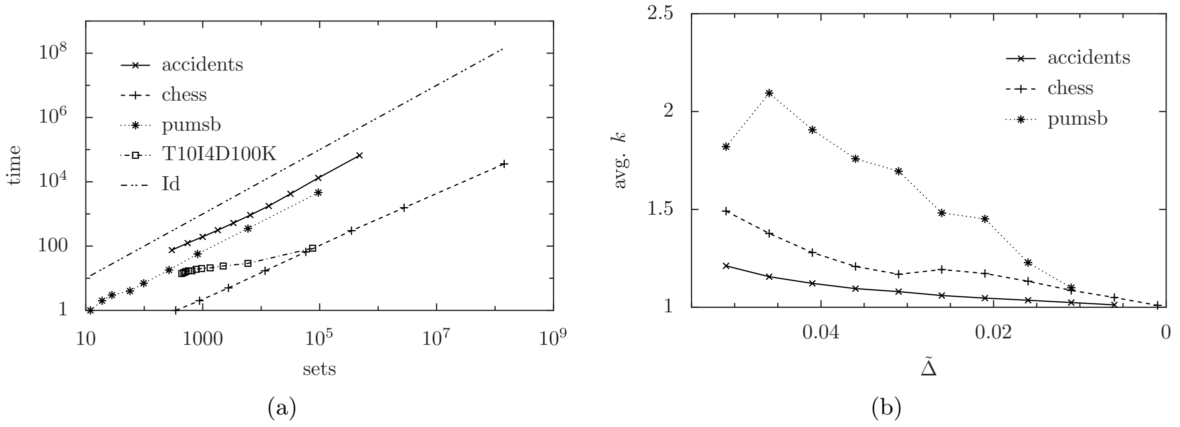


Figure 4.6.: Detailed behavior of Algorithm 9; namely, time versus number of closed sets (a) and average number of outer loop iterations k during closure computations (b).

The performance of the algorithm is determined by two factors: the total number of performed closure computations as well as the cost of individual closure computations. It is possible to relate the effect of our different algorithmic choices to either of the two aspects: the $\uparrow_\Delta$-speedup affects the total number of closure computations, whereas the choice between the direct closure implementation (Algorithm 7) and the Zero-Elimination algorithm (Algorithm 8) determines the cost of individual closure computations. Thus, we take a closer look to each of these factors separately.

As mentioned in Section 4.3.1, the direct closure implementation outperforms the Zero-Elimination algorithm in the given test configuration by a fair margin. The main reason for this, beside the low density of the test datasets, is that for all performed listing tasks the average number of iterations of the outer loop of Algorithm 7 is close to 1 and the maximum experienced numbers range from 3 (retail) to 13 (chess). Thus, in the test scenarios, Algorithm 7 computes the closures almost in linear time with a factor close to 1. This cost is better than what the Zero-Elimination algorithm achieves on the test data, despite the superior worst-case behavior of that approach. In fact, the average number of iterations of the outer loop even decreases for decreasing values of Δ. For three of the datasets this tendency is depicted in Figure 4.6(b).

[3]All experiments were performed on an Intel(R) Core(TM)2 Duo CPU E8400 @ 3.00GHz running 64 bit openSUSE 11.0 Linux.

Hence, we have an explanation for the sub-linear scaling of Algorithm 9 observed above.

	chess			msweb			retail		
$\tilde{\Delta}$	$\uparrow_\Delta$	w/o	t	$\uparrow_\Delta$	w/o	t	$\uparrow_\Delta$	w/o	t
0.041	2.7	22	2	1.8	260	3	1.3	15K	1.5K
0.031	3.1	18	5	2	262	3	1.4	15K	1.7K
0.021	2.6	14	5.9	1.8	269	4	1.6	15K	1.8K
0.011	2	10	5.5	1.9	271	5	1.8	16K	2.1K
0.001	1.6	6	3.8	2.6	257	16	2.6	14K	342

Table 4.3.: Effect of the $\uparrow_\Delta$-speedup on the number of closure computations and the total time.

We now turn to the total number of closure computations and to the degree it is effected by the $\uparrow_\Delta$-speedup. Table 4.3 shows the number of closure computations that were performed with ("$\uparrow_\Delta$") and without ("w/o") the speedup along with the relative amount of saved total computation time ("t") for "chess", "msweb", and "retail". Recall that the factor of redundant closure computations is at most $|E|$. Indeed, this bound is attained without the $\uparrow_\Delta$-extension. However, when $\uparrow_\Delta$-based pruning is in effect the factor goes down to small numbers that are apparently independent of the size of E. Consequently, the speedup is most significant for datasets over a large ground set, e.g., for "retail" ($|E| = 16,469$) where the total computation time is more than a thousand times faster with the extension than without it.

4.5. Related Work

In this section, we discuss similarities and differences between strong closedness and related notions from the data mining literature. We consider semantic as well as algorithmic aspects—in particular properties related to our algorithmic methodology of reducing pattern discovery to fixpoint enumeration problems.

4.5.1. Relative Strength or Tolerance

Our notion of strong closedness is defined with respect to an additive *absolute* value (i.e., a positive integer Δ). In some of the applications, however, one might be interested in strong closedness in a *relative* sense. Figure 4.7 gives an exemplary scenario motivating this idea. It shows the family $\mathcal{C}_1$ based on a fictitious dataset over elements $\{a, b, c\}$ containing 9089 times the data record ab, nine times ac, and once each of a and abc. In a relative sense the set ac is "much more closed" than the set a because augmenting a to ab would reduce the support count by a factor of only 0.001 while augmenting ac to abc would reduce the support count by a factor of 0.9. However, there is no value of Δ such that $\mathcal{C}_\Delta$ contains ac but not a.

The notion of relative strength has already been investigated in Cheng et al. [2006], referred to as "δ-tolerant closed frequent itemsets". We recall the definition from Cheng et al. [2006] in a slightly different, but equivalent form.

Definition 4.4 (relatively δ-closed)**.** Let $\mathcal{D}$ be a dataset over a ground set E. For a real number $\delta \in (0, 1]$, a set F is **relatively δ-closed** if $|\mathcal{D}[F']| < \delta \cdot |\mathcal{D}[F]|$ holds for every F' with

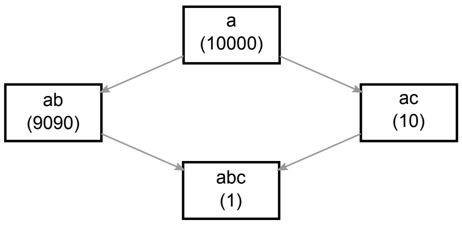


Figure 4.7.: Example motivating relative strength from Section 4.5.1.

$F \subset F' \subseteq E$. ◊

The notion can be combined with the concept of minimum frequency in the standard way; for an integer frequency threshold $t \geq 0$, the family of relatively δ-closed frequent sets is denoted by $\mathcal{C}^{\text{rel}}_{\delta,t,\mathcal{D}}$. In Cheng et al. [2006], the authors discuss the relationship between the families of relatively δ-closed frequent and closed frequent sets; a set $F \subseteq E$ is closed frequent if and only if it is relatively 1-closed frequent, i.e., $F \in \mathcal{C}^{\text{rel}}_{1,t,\mathcal{D}}$. Since $\mathcal{C}^{\text{rel}}_{\delta_1,t,\mathcal{D}} \subseteq \mathcal{C}^{\text{rel}}_{\delta_2,t,\mathcal{D}}$ holds for every $0 < \delta_1 \leq \delta_2 \leq 1$, each relatively δ-closed frequent set is also a closed frequent set for every $\delta \in (0,1]$.

In order to design an efficient algorithm for listing the relatively δ-closed sets, it would be useful to know whether the fixpoint enumeration technique can also be applied to them. Unfortunately, this is not the case, because the relatively δ-closed sets generally have no inducing closure operator for all $\delta \in (0,1]$. To see this consider again the example in Figure 4.7 and assume that there is a closure operator σ'_δ that generates $\mathcal{C}^{\text{rel}}_{0.5,0,\mathcal{D}} = \{ab, ac, abc\}$. Then $\sigma'_\delta(a)$ must be either ab or ac. But both options would violate monotonicity of σ'_δ: In case $\sigma'_\delta(a) = ab$ we have $a \subseteq ac$ but $\sigma'_\delta(a) = ab$ is not a subset of $\sigma'_\delta(ac)$, which has to be equal to ac. In case $\sigma'_\delta(a) = ac$ the argument is symmetric. Thus, the listing approach based on explicit closure computations can in general not be applied to enumerate $\mathcal{C}^{\text{rel}}_\delta$ for arbitrary parameters.

4.5.2. Free Sets

In data mining research, the term "minimal generator" (often only "generator") refers to minimal members of support equivalence classes (see Section 2.3.1). Alternatively, they can be defined as minimal sets generating a certain fixpoint of an arbitrary closure operator. For the standard support closure operator they play an important role because they can be used together with the closed sets to generate all minimal non-redundant association rules.

Yet another characterization of minimal generators are "(implication-) free sets" [Boulicaut et al., 2003]. A free set with respect to a dataset $\mathcal{D}$ over E is a set $F \subseteq E$ that cannot be decomposed into disjoint subsets $X, Y \subseteq F$ such that $\mathcal{D}[X] = \mathcal{D}[X \cup Y]$, i.e., such that X implies Y. Interestingly, just as we quantify closedness in this chapter, Boulicaut et al. [2003] also quantifies the property of being free by introducing the notion of "Δ-free sets", which is based on the concept of "Δ-strong rules".

Definition 4.5 (Δ-free)**.** Let $\mathcal{D}$ be a dataset over E and Δ be a positive integer. A **Δ-strong rule** is a rule $X \to Y$ over E (see p. 16) satisfying[4] $|\mathcal{D}[X \cup Y]| > |\mathcal{D}[X]| - \Delta$. A set $F \subseteq E$ is

[4]Note that the original definition of Boulicaut et al. [2003] uses $\geq$ instead of $>$, which shifts the resulting

called **Δ-free** if $X \to Y$ is *not* Δ-strong for all disjoint subsets $X, Y \subseteq F$ with $Y \neq \emptyset$. ◊

Since 1-free sets are the minimal generators of the standard support closure operator, i.e., σ_1, we are interested in whether this connection holds for arbitrary Δ. As it turns out, this is only true for one direction.

Proposition 4.9. *Let $\mathcal{D}$ be a dataset over E and $\Delta > 1$ be a positive integer. If a set $F \subseteq E$ is a minimal generator of its Δ-closure $\sigma_\Delta(F)$ (w.r.t. σ_Δ and $\mathcal{D}$) then F is Δ-free in $\mathcal{D}$.*

Proof. Suppose $F \subseteq E$ is not Δ-free in $\mathcal{D}$. Then there are disjoint $X, Y \subseteq F$ with $Y \neq \emptyset$ such that $|\mathcal{D}[X \cup Y]| > |\mathcal{D}[X]| - \Delta$. We show that in this case the subset $G = F \setminus Y$ of F generates $\sigma_\Delta(F)$, i.e., $\sigma_\Delta(G) = \sigma_\Delta(F)$—implying that F is not a minimal generator of its Δ-closure.

By the definition of the delta augmentation $\hat{\sigma}_\Delta$ we know that $\hat{\sigma}_\Delta(X) \supseteq Y$, and, extensivity and monotonicity of $\hat{\sigma}_\Delta$ imply that $\hat{\sigma}_\Delta(G) \supseteq (G \cup Y) = F$. Moreover, due to the monotonicity of the iterated delta augmentation σ_Δ, it holds that

$$\sigma_\Delta(G) = \sigma_\Delta(\hat{\sigma}_\Delta(G)) \supseteq \sigma_\Delta(F) \ .$$

Together with this statement, the monotonicity of σ_Δ also implies equality of $\sigma_\Delta(G)$ and $\sigma_\Delta(F)$. Hence, F is not a minimal generator of $\sigma_\Delta(F)$. □

The other direction is not true already for $\Delta = 2$. This is shown by the Example given in Figure 4.8. Thus, enumerating all minimal generators of σ_Δ (which could be done, e.g., using an adapted version of Algorithm 4) results only in enumerating a subset of all Δ-free sets of a dataset. All Δ-free sets can however easily be enumerated by an efficient algorithm based on their property of forming an independence system, i.e., an anti-monotone set system (again see Boulicaut et al. [2003]).

	a	b	c
1	1	1	1
2	1	0	1
3	1	0	0
4	0	1	0
5	0	1	0

Figure 4.8.: Example showing that Δ-free sets are not generally minimal generators of σ_Δ: ab does not contain a 2-strong rule, but $\sigma_2(a) = \sigma_2(ab) = abc$; hence, ab is not a minimal generator.

4.5.3. Metric Frequent Set Approximations

We know that the frequent 1-closed sets provide a lossless compact representation of the family of frequent sets $\mathcal{F}$, i.e., for all $F \in \mathcal{F}$ there is a representative closed set C having the same support as F. However, as mentioned earlier, Δ-closed sets do *not* provide a lossless compact

families by one. Here, we use this modified definition for the sake of consistency with Δ-closedness.

representation of frequent sets for any $\Delta > 1$. To see this e.g. for $\Delta = 2$, consider again the example given in Figure 4.1 (p. 61). For frequency threshold $t = 1$, we have $\mathcal{C}_{2,1} = \{\emptyset, abcd, ef\}$ and, e.g., the frequent set de has not representative in $\mathcal{C}_{2,1}$.

Generally there is an interest in approximations of frequent set families that are not necessarily lossless representations but perform some kind of trade-off between representativity and compactness (e.g., Afrati et al. [2004], Boulicaut et al. [2003], Xin et al. [2005]). There are different ways to specify the representativity of an approximation. For instance, in Xin et al. [2005] it is given with respect to a metric defined on the pattern space. More precisely, for a normalized distance measure μ between sets and for a given distance threshold $\delta \in [0, 1]$, the authors propose to compute a potentially small family $\mathcal{R}$ of *representative sets* such that for each frequent set F there is a representative set $R \in \mathcal{R}$ satisfying $F \subseteq R$ and $\mu(F, R) \leq \delta$. The family of frequent sets is then approximated by the family of sets covered by the union of the δ-balls of the representative sets.

Now considering Δ-closed sets, we know that they have an intrinsic connection to the pseudosemimetric[5] of symmetric support difference, i.e., $d : \mathcal{P}(E) \times \mathcal{P}(E) \to \mathbb{N}$ with $d(F, F') = |\mathcal{D}[F] \ominus \mathcal{D}[F']|$. In particular, we know by Proposition 4.2 that no two Δ-closed sets are Δ-close to each other with respect to d. On the other hand, as discussed above, the Δ-closed sets do not generally provide representatives with a bounded d-distance for all frequent sets. That is, strong closedness puts maximum emphasis on non-redundancy and neglects representativity.

If one is interested in shifting the closedness notion of this chapter towards a similar trade-off between non-redundancy and representativity as made by the metric clustering approach, one could consider the modified operator $\tilde{\sigma}_{\Delta,\Lambda} : \mathcal{P}(E) \to \mathcal{P}(E)$ with

$$\tilde{\sigma}_{\Delta,\Lambda}(F) = \begin{cases} \sigma_\Lambda(F), & \text{if } |\mathcal{D}[\sigma_\Lambda(F)]| \geq |\mathcal{D}[F]| - \Delta \\ \tilde{\sigma}_{\Delta,\Lambda-1}(F), & \text{otherwise} \end{cases} .$$

That is, the operation $\tilde{\sigma}_{\Delta,\Delta}$ represents a set F by its maximal Λ-closed superset for $\Lambda = 1, \ldots, \Delta$ having a support of at most Δ data records less than F. With this operator we have the guarantee that for all frequent sets $F \in \mathcal{F}_t$ of a dataset $\mathcal{D}$ there is an representative $C \in \tilde{\sigma}_{\Delta,\Delta}(\mathcal{F}_t)$ with $d(F, C) \leq \Delta$.

4.6. Summary and Discussion

In this chapter, we have introduced strong closedness—a quantification of closedness based on a numerical parameter Δ specifying the magnitude of drop in support when augmenting a set. Just as with a minimum frequency threshold, pruning based on minimum strength of closedness induces a monotone decreasing sequence of output sets for increasing strength thresholds. Moreover, for two distinct strongly closed sets we have the guarantee that their support has a symmetric difference of at least Δ, i.e., the family of strongly closed sets satisfies a distinctiveness condition with respect to their support sets. Thus, by selecting the parameter Δ appropriately, the user can control the size of the resulting output while at the same time having a semantically meaningful specification of what kinds of patterns to expect. This is especially useful in applications where increasing the support threshold would lead to an unacceptable

[5] By "pseudosemimetric" we refer to a positive symmetric mapping $d \colon X \to \mathbb{R}$ with $d(x, x) = 0$ for all $x \in X$.

loss of less frequent, but still interesting long patterns. At the same time, strongly closed sets are robust against noise and/or dynamic changes in the dataset. It is possible to generate strongly closed sets using a suitable closure operator. With the definition of this operator and the divide-and-conquer fixpoint listing approach it is possible to design an efficient algorithm for listing all strongly closed sets of a datasets. In particular, for non-sparse datasets its time complexity is equal to the best known time bound for listing ordinary closed frequent sets. Our experimental evaluation shows that strongly closed sets indeed give rise to small but interesting solution sets, and thus offer a viable alternative or addition to the standard minimum frequency approach.

4.6.1. Combination with the Constraint-Based Approach

In the beginning of this chapter, it was discussed that strong closedness and constrained closedness are contrary approaches in the sense that strong closedness is stricter than ordinary closedness and constrained closedness is partially weaker[6]. In fact, it is straightforwardly possible to combine both approaches as done in the following definition.

Definition 4.6 (Δ-Closed Constrained)**.** Let $(E, \mathcal{F})$ be a set system (induced by some constraint $c\colon \mathcal{P}(E) \to \{0,1\}$) and $\mathcal{D}$ a dataset over E. A set $X \subseteq E$ is called **Δ-closed constrained** for an integer $\Delta \geq 0$ if $X \in \mathcal{F}$ and if for all $Y \supset X$ with $Y \in \mathcal{F}$ it holds that $|\mathcal{D}[Y]| \leq |\mathcal{D}[X]| - \Delta$. $\Diamond$

Conveniently, the results of Chapter 3 also hold for this modified notion of closedness. In particular, the algorithmic approach of fixpoint enumeration remains applicable for well-behaved constraints. That is, we can show that, for confluent set systems $\mathcal{F}$ and non-redundant datasets $\mathcal{D}$, the operation $\rho\colon \mathcal{P}(E) \to \mathcal{P}(E)$ given by

$$\begin{aligned} \rho(F) &= \max \Sigma(F) \\ \text{where } \Sigma(F) &= \{F' \in \mathcal{F}\colon\ F \subseteq F' \wedge \sigma_\Delta(F) = \sigma_\Delta(F)\}\ , \end{aligned}$$

is a well-defined closure operator that induces the Δ-closed constrained sets. In fact, this modification of the operator introduced in Section 3.4 does not depend on specific features of Δ-closedness other than having an inducing closure operator: the same construction works for any closedness notion having an associated closure operator that can replace σ_Δ in the definition above.

4.6.2. Closedness Notions without Inducing Closure Operators

Beside the apparent generality of the closure operator method, we already encountered several well-motivated notions of "closedness" that do not generally have an inducing closure operator. Examples are the closed constrained sets for non-confluent set systems as well as the relative Δ-closedness discussed in Section 4.5.1. A further example is the extension of Δ-closedness to two dimensions: Let $\mathcal{D}$ be a dataset over a set E (columns) and let T denote the transaction

[6]To be precise, the defining condition for closed constrained sets consists of two parts, the latter of which ("all *feasible* extension are of reduced support") being a relaxation of ordinary closedness—the first ("set itself must be feasible") is stricter.

identifiers of $\mathcal{D}$ (rows). For integers $\Delta_1, \Delta_2 \geq 0$, a pair (A, B) with $A \subseteq T$ and $B \subseteq E$ is a (Δ_1, Δ_2)-*concept* if A is Δ_1-closed for the transpose matrix of $\mathcal{D}$ and B is Δ_2-closed (for $\mathcal{D}$). Note that ordinary concepts defined in formal concepts analysis (see Section 5.3 for an introduction) are $(1, 1)$-concepts. As shown by the example in Figure 4.9, generally there is no closure operator inducing the projections to the first component (usually referred to as attributes) of all (Δ_1, Δ_2)-concepts.

	a	b	c	d	e	f	g	h
1	0	0	0	0	1	0	0	0
2	0	1	1	1	1	0	0	0
3	0	1	1	1	1	0	0	0
4	0	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	0
6	0	0	0	1	1	1	1	0
7	0	0	0	1	1	1	1	0
8	0	0	0	1	0	0	0	0

Figure 4.9.: Example showing that the projections of (Δ_1, Δ_2)-closed concepts do not generally form a closure system: the only $(2, 2)$-closed concepts are $\langle bcde, 2345\rangle$ and $\langle defg, 4567\rangle$ but neither their attribute intersection nor their object intersection induces a $(2, 2)$-closed concept.

It is an interesting open research issue to determine the computational complexity for enumerating the closed patterns with respect to such non closure-induced closedness notions. For the very general problem of closed constrained set listing we know that there cannot be an output polynomial time algorithm (unless $\mathbf{P} = \mathbf{NP}$) but for the concrete settings mentioned above no hardness result is known and efficient algorithm are definitely worth looking for. Even the fixpoint enumeration method could potentially remain applicable if one considers other partial orders (w.r.t. to which a monotone inducing operator could exist) and other transformed ground sets.

5. Randomized Closed Set Generation

In this chapter, we utilize the Markov chain Monte Carlo technique (see, e.g., Randall [2006]) for sampling single closed sets of a database as an alternative to exhaustively listing all of them. In particular, we design a Metropolis-Hastings algorithm [Hastings, 1970] for generating from a given input $\mathcal{D}$ a closed set $C \in \mathcal{C}(\mathcal{D})$ according to any desired strictly positive distribution π. This can be used for controlled repeated pattern sampling—an approach to local pattern discovery that is fundamentally different to those of the previous chapters—as well as for pattern counting, which is useful for estimating parameter effects and computation time of enumeration algorithms.

While it is relatively simple to design an efficient Markov chain method that operates on the subset lattice, e.g., to generate sets according to their frequency, this approach is not directly applicable to the generation of closed sets. Using a set sampler within a rejection sampling approach does not lead to an efficient algorithm because the number of sets can grow exponentially in the number of closed sets. Consequently, the probability of successfully drawing a closed set with rejection sampling can be very small; for a small database like "msweb" from the UCI machine learning repository with 285 items and $32K$ transactions already as small as 10^{-80}. Similarly, mapping a drawn itemset to its smallest closed superset is not a viable option. In this case the probability of drawing a specific closed set would correspond to the size of the support equivalence class it represents. However, as we have no efficient way of inferring this size, this approach would introduce an uncontrollable bias to the closed set sampling.

Thus, after discussing the potential of sampling for pattern discovery in more detail (Section 5.1) and recapping the basics of Markov chain Monte Carlo methods (Section 5.2), we construct a random walk algorithm that directly uses the closed sets as state space (Section 5.3). Moreover, we present exemplary local pattern discovery applications (Section 5.4) and investigate the connection to pattern counting (Section 5.5). The latter insight allows us to provide some further analysis of the computational complexity of pattern sampling. Finally, the chapter is wrapped up with a concluding discussion.

5.1. Controlled Repeated Pattern Sampling

Chapters 3 and 4 introduced efficient algorithms that enumerate interesting collections of closed sets. It was assumed that these collections are the final output, and efficiency was defined in this respect. When viewed from a global (application-driven) perspective though, the enumerated patterns are usually only an intermediate result, from which a final pattern collection is selected. Hence, enumeration is only the first half of a surrounding local pattern discovery process. This two phase approach, which we want to refer to as "exhaustive search" is illustrated in Figure 5.1: during the enumeration step a part of the implicitly defined pattern space is physically constructed—we refer to that part as "enumeration space"—and then,

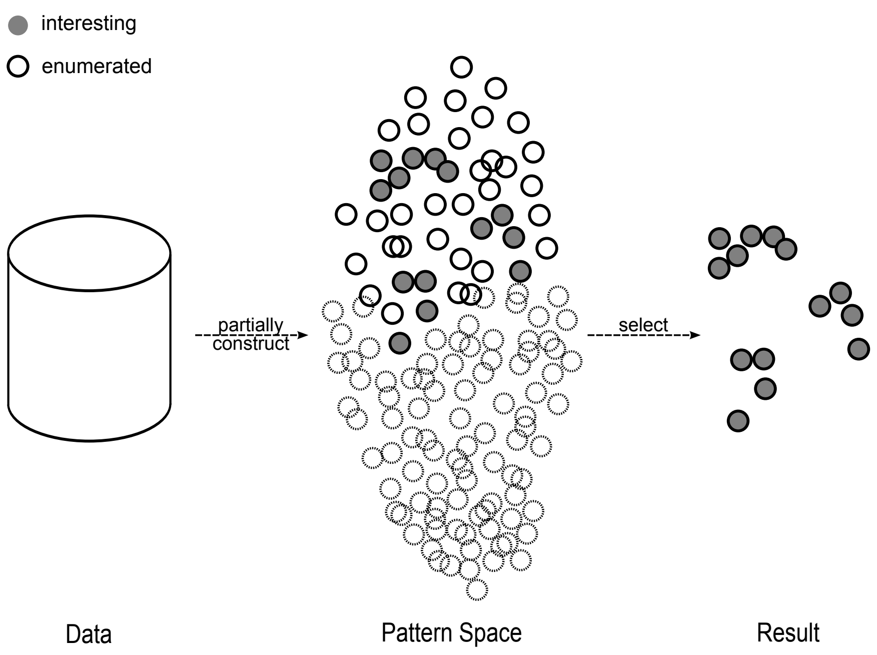


Figure 5.1.: The exhaustive search paradigm. It involves the complete generation of an enumeration space that is guaranteed to contain all interesting patterns. However, the size of that space usually has no reasonable bound with respect to the input size and is hard to predict.

during the selection step, the most valuable patterns from this enumeration space are collected with respect to some interestingness measure. An example for this paradigm is the optimistic estimator based approach for the subgroup discovery problem (see p. 21) where the enumeration space is the family of all potentially interesting sets (with respect to the estimator $\hat{q}_{\text{bino}}$) and the truly interesting patterns (with respect to q_{bino}) are selected for the result family. Note that, in this example, enumeration and selection are algorithmically interweaved, i.e., sets are already selected throughout the enumeration phase. Many additional examples emerge from the Lego-approach to data mining [Knobbe et al., 2008] where patterns are selected according to their utility for constructing global models.

For these approaches, the enumeration step can constitute a severe bottleneck. Even if enumeration is performed by an output polynomial time algorithm, its computation time is essentially unpredictable. As discussed in Chapter 1 (see p. 4), the number of enumerated sets cannot be directly controlled and the computation time is at least proportional to that number in case the complete enumeration space is generated. On the other hand, if one enforces a maximum computation time by aborting the execution at a certain point, one ends up with an uncontrolled subset of the enumeration space, which depends on the internal search order of the enumeration algorithm.

In contrast, suppose we have access to an efficient sampling procedure simulating some distribution $\pi : \mathcal{C} \to [0,1]$ that can be chosen as a parameter. Then it is possible to efficiently generate a subset of the pattern space that consists exactly of as many patterns as are truly required and that is representative for the distribution π. For instance, if π is defined with respect to some interestingness measure q, e.g., $\pi(\cdot) = q(\cdot)/Z$ where Z is a normalizing constant,

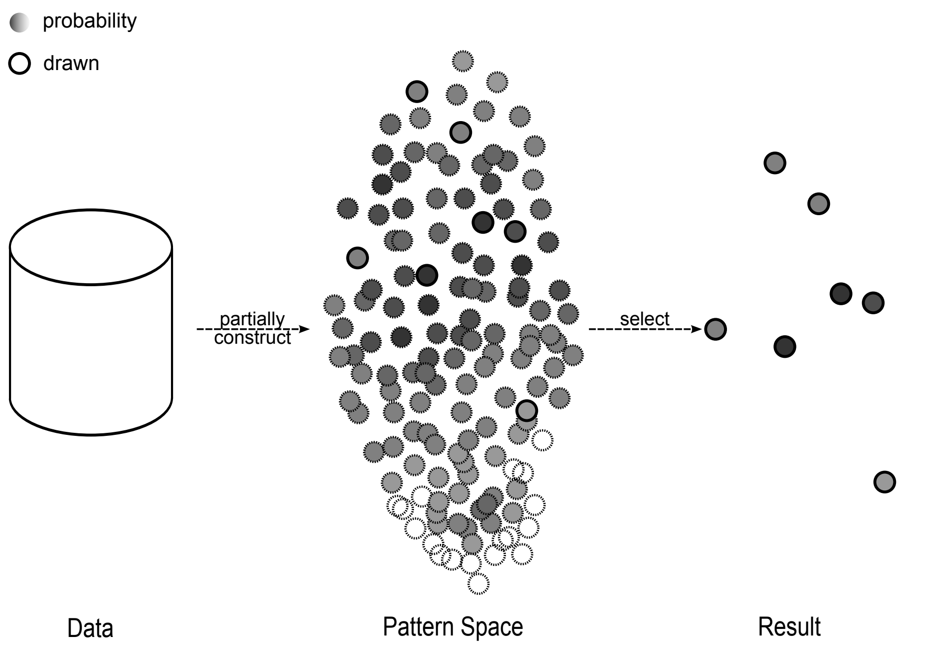


Figure 5.2.: The controlled repeated sampling paradigm. Instead of an explicit construction of a potentially huge part of the pattern space, only a small designated number of patterns is randomly generated. While this approach does not guarantee to find any or all patterns satisfying some hard interestingness threshold, it offers control over computation time and output size.

the resulting pattern collection reflects a controlled application-dependent notion of interestingness. In addition, a dynamic adjustment of the distribution can be used to achieve a good coverage/distinctiveness of the result elements (e.g., by successively reducing the weight of already covered transactions as proposed in Lavrac et al. [2004]). Figure 5.2 illustrates this alternative approach, which we want to refer to as "controlled repeated pattern sampling". A potentially positive side-effect of this paradigm is that instead of the usual hard constraints it utilizes parameter-free soft constraints [Bistarelli and Bonchi, 2007]. Hence, the user is freed of the often troublesome task of finding appropriate hard threshold parameters such as a minimum frequency threshold.

Altogether, this motivates the design of an efficient algorithm for the following computational problem, where by efficient we now refer to the standard notion of input polynomial time complexity.

Problem 5.1 (SAMPLE-CLOSED-SET). *Given* a dataset $\mathcal{D}$ over E and a probability mass function $\pi : \mathcal{C}(\mathcal{D}) \to [0,1]$ (via oracle access), *randomly generate* a closed set $C \in \mathcal{C}(\mathcal{D})$ according to π. $\Diamond$

5.2. Markov Chain Monte Carlo Methods

A naive approach for solving Problem 5.1 is to generate a list of all closed sets $C_1, \ldots, C_N$ with $\pi(C_i) > 0$, draw an $x \in [0,1]$ uniformly at random, and then return the unique set C_k with $\sum_{i=1}^{k-1} \pi(C_i) \leq x < \sum_{i=1}^{k} \pi(C_i)$. However, the exhaustive enumeration of all closed sets is precisely what we want to avoid. Hence, we need an indirect *non-enumerative* sampling algorithm. Markov chain Monte Carlo methods are such a technique based on the idea of simulating a discrete stochastic process that converges to the desired target distribution. In particular, a time-homogeneous Markov chain is used as stochastic process because the process simulation can then be reduced to the simulation of tractable local state transitions. If the convergence to stationarity is fast compared to the size of the state space, this way of sampling is much more efficient than enumerative sampling.

In this section, we recap the basics of Markov chain Monte Carlo methods and illustrate them with an exemplary algorithm for sampling frequent sets of a database. For a more detailed overview we refer to the survey of Randall [2006] respectively to the book of Levin et al. [2008] for an in-depth introduction.

5.2.1. Basics

Let Ω denote a finite set, acting as sample space in what follows. We identify a probability distribution Ω given by mass function $\mu \colon \Omega \to \mathbb{R}$ with the row vector $(\mu(\omega))_{\omega\in\Omega}$. The **uniform distribution** on Ω, i.e., $(1/|\Omega|, \ldots, 1/|\Omega|)$, is denoted $u(\Omega)$. The distance between two probability distributions μ and ν on Ω can be measured using the **total variation distance**

$$\|\mu, \nu\|_{\mathrm{tv}} = 1/2 \sum_{x\in\Omega} |\mu(x) - \nu(x)| \quad ,$$

which gives the maximum difference between the probabilities assigned by μ and ν to single event. For a (not necessarily discrete) random variable X, the fact that X is governed by a distribution ξ is denoted $X \sim \xi$.

A **Markov chain** on state space Ω is a sequence of discrete random variables $(X_t)_{t\in\mathbb{N}}$ with domain Ω satisfying the Markov condition, i.e., that

$$\mathbb{P}[X_{t+1} = x | X_1 = x_1, \ldots, X_t = x_t] = \mathbb{P}[X_{t+1} = x | X_t = x_t]$$

for all $t \in \mathbb{N}$ and $x, x_1, \ldots, x_t \in \Omega$ satisfying $\mathbb{P}[X_1 = x_1, \ldots, X_t = x_t] > 0$. In addition, in this work we assume that the probabilities $\mathbb{P}[X_{t+1} = x | X_t = x_t]$ do not depend on t, i.e., the chain is **time-homogeneous**. Thus, given a probability distribution μ_0 on the initial state X_0, the distribution of all X_t with $t \in \mathbb{N}$ is completely specified by the **state transition probabilities** $p(x,y) = \mathbb{P}[X_1 = y | X_0 = x]$ of all $x, y \in \Omega$. Namely, the stochastic **state transition matrix** $P = (p(x,y))_{x\in\Omega, y\in\Omega}$ specifies X_t recursively as follows: if X_{t-1} is distributed according to μ_{t-1} then the distribution μ_t of X_t results from multiplying μ_{t-1} to P from the left, i.e., $\mu_t = \mu_{t-1}P$. Moreover, the **t-step transition probabilities** $p^t(x,y) = \mathbb{P}[X_t = y | X_0 = x]$ are exactly the entries of the t-th power P^t of the transition matrix. The transition probabilities $p(x,x)$ for $x \in \Omega$ are called **self-loop probabilities**.

We call a state $y \in \Omega$ **reachable** from a state $x \in \Omega$ if there is a $t \in \mathbb{N}$ such that $p^t(x,y) > 0$. The chain $(X_t)_{t\in\mathbb{N}}$ is called **irreducible** if any two states are reachable from one another. Any irreducible Markov chain with transition matrix P has a unique **stationary distribution** π, i.e., $\pi P = \pi$. On the other hand, if there is a distribution $\pi' \colon \Omega \to [0,1]$ satisfying the **detailed balance condition**

$$\forall x, y \in \Omega, \pi'(x)p(x,y) = \pi'(y)p(y,x) \tag{5.1}$$

then π' is the stationary distribution and the chain is called **time-reversible**. Moreover, $(X_t)_{t \in \mathbb{N}}$ is called **aperiodic** if for all $x, y \in \Omega$ with x is reachable from y there is a $t_0 \in \mathbb{N}$ such that for all $t \geq t_0$ it holds that $p^t(x,y) > 0$, and it is called **ergodic** if it is irreducible and aperiodic. Any ergodic Markov chain converges to its stationary distribution independently of the distribution of X_0. In particular there are constants $\alpha \in (0,1)$ and $C > 0$ such that for all $t \in N$ it holds that

$$\max_{x \in \Omega} \|p^t(x,\cdot), \pi\|_{\text{tv}} \leq C\alpha^t \ . \tag{5.2}$$

In particular, one can show that the distance to the stationary distribution is monotonically decreasing, i.e., for all $x \in \Omega$ and $t \in \mathbb{N}$ it holds that $\|p^{t+1}(x,\cdot), \pi\|_{\text{tv}} < \|p^t(x,\cdot), \pi\|_{\text{tv}}$.

A Markov chain Monte Carlo method generates an element from Ω according to π by simulating $(X_t)_{t \in \mathbb{N}}$ up to a time s such that X_s is distributed "almost" according to π. For that approach one needs (a) some efficiently computable starting state $X_0 = x_0 \in \Omega$ and (b) be able to efficiently generate a "neighbor" $y \sim p(x,\cdot)$ for a given state x.

5.2.2. Sampling Frequent Sets

As illustrative Markov chain Monte Carlo method, consider the following algorithm for sampling a frequent set from a given input database with some frequency threshold. In fact, the approach does not use any properties of the minimum frequency constraint other than that it induces independence systems $(E, \mathcal{F})$.

Example 5.1. Let $\mathcal{D}$ be a dataset over ground set $E = \{e_1, \ldots, e_n\}$ and $f \in [0,1]$ be some frequency threshold. Then the following Markov chain Monte Carlo method can be used to sample an element F_s from the family of frequent sets $\mathcal{F} = \{F \subseteq E \colon q_{\text{freq}}(F) \geq f\}$: return the set $F_s \subseteq E$ that is computed as follows. Set $F_0 = \emptyset$ and then for $j = 0, \ldots, s-1$ repeatedly apply the following randomized perturbation procedure $F_j \mapsto F_{j+1}$:

1. set F' to F_j
2. draw $k \sim u(\{1, \ldots, n\})$
3. if $e_k \in F'$ set F' to $(F' \setminus \{e_k\})$; otherwise:
4. if $(F' \cup \{e_k\}) \in \mathcal{F}$ then set F' to $(F' \cup \{e_k\})$
5. with probability $1/2$ set F_{j+1} to F_j; otherwise: set F_{j+1} to F' $\Diamond$

Using the Markov chain basics from the previous section, we can verify that this example algorithm indeed generates frequent sets from $\mathcal{F}$ *almost* uniformly at random (for appropriate values of s). The perturbation procedure simulates one step of a Markov chain $(X_n)_{n \in \mathbb{N}}$ on $\mathcal{F}$. It is easy to see that corresponding state transition probabilities $p(\cdot,\cdot)$ are equal to $1/2n$

between sets with a symmetric difference of 1, and that all "remaining" probability is assigned to the self-loops, i.e.,

$$p(F,F') = \begin{cases} 1/2n, & \text{if } |F \ominus F'| = 1 \\ 1 - |\{F' \in \mathcal{F} \colon |F \ominus F'| = 1\}| /2n, & \text{if } F = F' \\ 0, & \text{otherwise} \end{cases}$$

where $\ominus$ denotes symmetric difference as usually. So the state transition probabilities $p(\cdot,\cdot)$ are symmetric. Consequently, the uniform distribution $u(\mathcal{F})$ satisfies the detailed balance condition (5.1) and, hence, is a stationary distribution of $(X_t)_{t\in\mathbb{N}}$.

In order to see that $u(\mathcal{F})$ is the unique stationary distribution and that it is indeed approached by the chain, it remains to check ergodicity. The symmetry of the transition probabilities also implies symmetry of the reachability relation of $(X_n)_{n\in\mathbb{N}}$. Together with the fact that $\mathcal{F}$ is closed under taking subsets this implies that $(X_n)_{n\in\mathbb{N}}$ is irreducible because all states are reachable from $\emptyset$. Moreover, there are non-zero self-loop probabilities $p(F,F)$ for every state $F \in \mathcal{F}$. This implies that $(X_n)_{n\in\mathbb{N}}$ is also aperiodic and together with irreducibility this means that $(X_n)_{n\in\mathbb{N}}$ is ergodic as required. The rate in which $(X_n)_{n\in\mathbb{N}}$ approaches the uniform distribution is shown in Figure 5.3 for two different input databases.

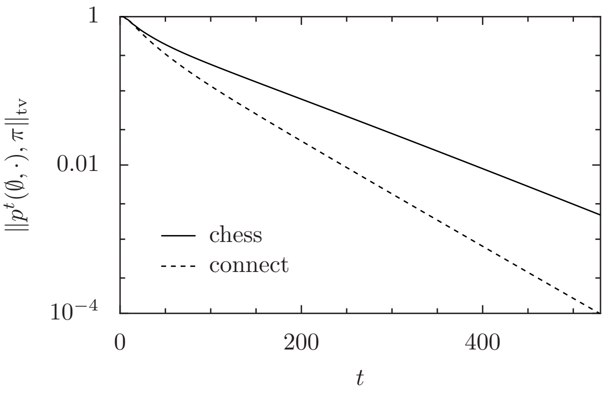


Figure 5.3.: Distributional convergence of the current state of the frequent set chain from Example 5.1 to the uniform distribution.

5.2.3. Mixing Times

After constructing a Markov chain $(X_t)_{t\in\mathbb{N}}$ that converges to a desired stationary distribution π and that can be simulated efficiently, there is one critical question that remains to be answered. Namely, after how many simulation steps t is the distribution of X_t "close enough" to π? This question is formalized by the **mixing time** of $(X_t)_{t\in\mathbb{N}}$ which is defined as

$$\tau(\epsilon) = \max_{x\in\Omega} \min\{t_0 \in \mathbb{N} \colon \forall t \geq t_0,\ \|p^t(x,\cdot),\pi\|_{\text{tv}} \leq \epsilon\}$$

as the minimum number of steps one has to simulate $(X_t)_{t\in\mathbb{N}}$ until the resulting distribution is guaranteed to be ϵ-close to its stationary distribution.

While for a single Markov chain we know, due to Equation 5.2, that $\tau(\epsilon) \in O(\ln 1/\epsilon)$, a Markov chain Monte Carlo algorithm induces a whole family of chains—one for each problem instance. Hence, the hidden constant in the mixing time bound is of critical importance: although the chains induced by most real-world datasets can have a small mixing time, there may be particularly hard problem instances inducing slow mixing. An example for this scenario is the algorithm of Example 5.1 that induces chains with a mixing time of less than $4n \ln n \ln(\epsilon^{-1})$ for all datasets of the FIMI repository [Goethals and Zaki, 2004], but has exponential mixing time for a sequence of constructed worst-case datasets.

In order to give such lower bounds on the mixing time of a Markov chain one can use the concept of **conductance** (sometimes also "bottleneck ratio"). For a set of states $S \subseteq \Omega$ its conductance, denoted Φ_S, is defined as $(1/\pi(S)) \sum_{x \in S, y \in \Omega \setminus S} \pi(x) p(x,y)$, and the conductance of the whole chain, denoted Φ, is the minimum conductance Φ_S over all state sets $S \subseteq \Omega$ with $\pi(S) \leq 1/2$. The following connection between the conductance of a chain and its mixing time holds.

Theorem 5.1 (Sinclair and Jerrum [1987]). *For the mixing time τ of a Markov chain with conductance Φ it holds that*

$$\tau(\epsilon) \geq \frac{1}{4\Phi} \log(\epsilon^{-1}) \ .$$

We can now show that for the chains of Example 5.1 the worst-case mixing time (over all possible input datasets) is slow, i.e., it has an exponential lower bound in the size of the ground set. In particular, for $n \in \mathbb{N}$ let $(X_t)_{t \in \mathbb{N}}$ be the frequent set Markov chain induced by the dataset $\mathcal{D} = \{\{1\}, \{2, \ldots, n\}, \{n+2, \ldots, 2n\}\}$ over $E = \{1, \ldots, 2n\}$ and frequency threshold $f = 1/|\mathcal{D}|$. Then it holds for the mixing time of $(X_t)_{t \in \mathbb{N}}$ that $\tau(\epsilon) \geq 2^{n-1} \log(1/2\epsilon)$. This can be seen as follows. Recall that the stationary distribution π of $(X_t)_{t \in \mathbb{N}}$ is the uniform distribution on the frequent sets $u(\mathcal{F})$. For the state space $\mathcal{F}$ it holds that

$$\mathcal{F} = \{\{1\}\} \cup \mathcal{P}(\{2, \ldots, n\}) \cup \mathcal{P}(\{n+2, \ldots, 2n\})$$

and $|\mathcal{F}| = 2^n$. If we choose $S = \{\{1\}\} \cup \mathcal{P}(\{2, \ldots, n\}) \setminus \{\emptyset\}$ then $|S| = 2^{n-1}$ and consequently $\pi(S) = 1/2$. Since only the n pairs $(\{i\}, \emptyset)$ with $i \in \{1, \ldots, n\}$ contribute to Φ_S this means for the conductance of $(X_t)_{t \in \mathbb{N}}$ that

$$\begin{aligned} \Phi \leq \Phi_S = \frac{\sum_{x \in S, y \notin S} \pi(x) p(x,y)}{\pi(S)} &= 2 \sum_{x \in \{\{1\}, \ldots, \{n\}\}} \pi(x) p(x, \emptyset) \\ &= 2n \frac{1}{2^n} \frac{1}{2(2n)} = \frac{1}{2^{n+1}} \ . \end{aligned}$$

Plugging this into the bound of Theorem 5.1 yields the claimed exponential lower bound on the mixing time.

5.3. Sampling Closed Sets

In this section we present a Markov chain Monte Carlo method for sampling closed sets. In contrast to our previous database formalization of binary datasets, formal concept analysis

[Ganter and Wille, 1999] treats data records and items in a symmetric fashion by modeling the data as binary relation. As the Markov chains we introduce here rely on freely switching the roles of items and data records, this symmetric formalization is better suited to describe our sampling algorithm. Hence, in this section, we take on the formal concept analysis perspective. That is, we present an algorithm for sampling a formal concept from a concept lattice given by an input context, which is equivalent to sampling a closed set of a binary dataset.

In the remainder of this section, after recapping formal concept analysis, we first show how the state space, i.e., the concept lattice of a given context, can be connected in a way that allows to efficiently select a random element from the neighborhood of some concept. We achieve this by using the closure operators induced by the context. Then, in a second step, we apply to this initial idea the Metropolis-Hastings algorithm and show that the resulting Metropolis process is irreducible. It can therefore be used to sample a concept according to any desired strictly positive distribution.

5.3.1. Formal Concept Analysis

A (formal) **context** is a tuple $(A, O, \mathcal{D})$ with finite sets A and O referred to as **attributes** and **objects**, respectively, and a binary relation $\mathcal{D} \subseteq A \times O$. The attributes can be regarded as taking on the role of the ground set and the objects as taking on the role of the data records of our previous binary dataset representation of input databases. The maps $O[\cdot] : \mathcal{P}(A) \to \mathcal{P}(O)$ and $A[\cdot] : \mathcal{P}(O) \to \mathcal{P}(A)$ are defined as

$$\begin{aligned} O[X] &= \{o \in O \colon \forall a \in X, (a, o) \in \mathcal{D}\} \text{ and} \\ A[Y] &= \{a \in A \colon \forall o \in Y, (a, o) \in \mathcal{D}\} \ , \end{aligned}$$

respectively. An ordered pair $C = (I, E) \in \mathcal{P}(A) \times \mathcal{P}(O)$ is called a (formal) **concept**, denoted $\langle I, E\rangle$, if $O[I] = E$ and $A[E] = I$. The set I is called the **intent** of C and E is called its **extent**. The set of all concepts for a given context is denoted $\mathcal{C}(A, O, \mathcal{D})$ or just $\mathcal{C}$ when there is no ambiguity. It is partially ordered by the **hierarchical order**, i.e., the binary relation $\preceq$ defined by $\langle I, E\rangle \preceq \langle I', E'\rangle$ if and only if $I \subseteq I'$ (or equivalently $E \supseteq E'$). For the minimal respectively maximal elements of $\mathcal{C}$ with respect to $\preceq$ we write "$\perp$" for $\langle A[O], O\rangle$ and "$\top$" for $\langle A, O[A]\rangle$. The set $\mathcal{C}(A, O, \mathcal{D})$ ordered by $\preceq$ forms a complete lattice called the **concept lattice** of $(A, O, \mathcal{D})$. Due to contexts being isomorphic to binary datasets and the symmetry of attributes and objects, we already know that

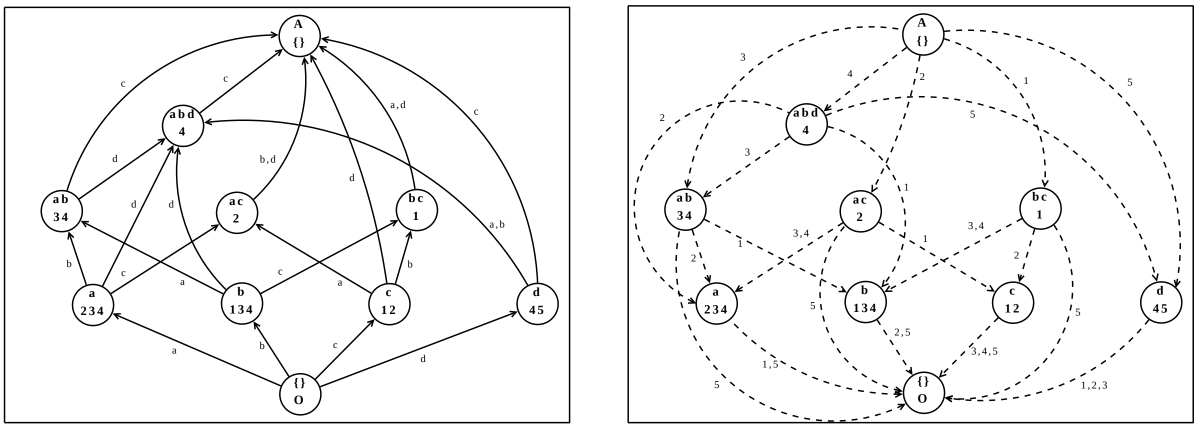
a) the maps $A[\cdot]$ and $O[\cdot]$ form an (order-reversing) Galois connection, i.e., for all $X \subseteq A$ and $Y \subseteq O$ it holds that
$$X \subseteq A[Y] \iff Y \subseteq O[X] \ ,$$

b) their compositions $\phi = A[O[\cdot]]$ and $\psi = O[A[\cdot]]$ form closure operators on $\mathcal{P}(A)$ and $\mathcal{P}(O)$, respectively,

c) and all concepts $C = \langle I, E\rangle \in \mathcal{C}$ are of the form $C = \langle \phi(X), O[\phi(X)]\rangle$ for some $X \subseteq A$ respectively $C = \langle A[\psi(Y)], \psi(Y)\rangle$ for some $Y \subseteq O$, i.e., the intents of all concepts are the fixpoints of ϕ and the extents the fixpoints of ψ.

We refer to ψ as the **object closure operator** and to ϕ as the **attribute closure operator**, the latter of which corresponds to the standard support closure operator of the binary dataset setting. Clearly, there is a one to one correspondence between concept intents (i.e., the fixpoints of the attribute closure operator) and the closed sets of the binary dataset representation (i.e., the fixpoints of the standard support closure operator). Sampling closed sets is, hence, equivalent to sampling formal concepts.

5.3.2. Generating Elements



(a) (b)

Figure 5.4.: Generator graphs $\mathbf{G}_\phi$ and $\mathbf{G}_\psi$ (drawn without self-loops) for the context from Table 5.1.

In order to construct a stochastic process on the concept lattice of a given context we will exploit its associated closure systems. More specifically, one can "move" from one closed set of a closure operator to another by a single element augmentation and a subsequent closure operation. This is exactly the underlying principle of all closed set enumeration algorithms of Chapters 2, 3, and 4. In this context, we refer to the involved single elements as *generating elements* that can be defined as follows.

Definition 5.1 (Generating Element)**.** Let ρ be a closure operator on $\mathcal{P}(X)$ and $C, C' \in \rho(\mathcal{P}(X))$ be fixpoints of ρ. We say that an element $x \in X$ **generates** C' from C with respect to ρ if $\rho(C \cup \{x\}) = C'$. The set of all such generating elements is denoted by $G_\rho(C, C')$. $\diamond$

These generating elements can be used to define a directed graph on the family of closed sets in which two vertices C, C' are joined by an arc if there is at least one element generating C' from C. This graph is implicitly traversed, e.g., by the lcm algorithm (Sec. 2.3.4, p. 29) and by the breadth-first minimum generator listing algorithm (Sec. 2.4.2, p. 34)—see also Section 3.6.2 (p. 55) for a comparison to the divide-and-conquer approach.

Definition 5.2 (Generator Graph)**.** Let ρ be a closure operator on $\mathcal{P}(X)$. The **generator graph** $\mathbf{G}_\rho = (\mathbf{C}_\rho, \mathbf{E}_\rho, \mathbf{l}_\rho)$ of ρ is the directed labeled graph on the fixpoints of ρ as vertices, i.e, $\mathbf{C} = \rho(\mathcal{P}(X))$, with edges

$$\mathbf{E}_\rho = \{(C, C') \in \mathbf{C}_\rho \times \mathbf{C}_\rho \colon G_\rho(C, C') \neq \emptyset\} \ ,$$

and with edge labels $\mathbf{l}_\rho : \mathbf{E}_\rho \to \mathcal{P}(X)$ corresponding to the generating elements, i.e., $\mathbf{l}_\rho(C, C') = G_\rho(C, C')$. $\Diamond$

It is easy to see that $\mathbf{G}_\rho$ is a) acyclic except for self-loops and b) rooted in $\rho(\emptyset)$: statement a) follows by observing that $(C, C') \in \mathbf{E}_\rho$ implies $C \subseteq C'$, i.e., $\mathbf{E}_\rho$ is compatible with a partial order, and for b) note that for all closed sets $C = \{x_1, \ldots, x_k\}$ the sequence $x_1, \ldots, x_k$ corresponds to an edge progression (walk) from $\rho(\emptyset)$ to C due to extensivity and monotonicity of ρ. A random walk on the generator graph can be performed by the following procedure: in a current closed set $Y \in \rho(\mathcal{P}(X))$ draw an element $x \sim u(X)$, and then move to $\rho(Y \cup \{x\})$. The transition probability from a set Y to a set Z is then directly proportional to the number of generating elements $|G_\rho(Y, Z)|$, which can be computed efficiently by at most $|X|$ closure computations (assuming that ρ can be computed efficiently).

	a	b	c	d
1	0	1	1	0
2	1	0	1	0
3	1	1	0	0
4	1	1	0	1
5	0	0	0	1

Table 5.1.: Example context with attributes $\{a, b, c, d\}$ and objects $\{1, 2, 3, 4, 5\}$.

Now, if one identifies concepts with their intents respectively with their extents, there are *two* associated generator graphs to a given context—the one induced by the attribute closure operator ϕ and the one induced by the object closure operator ψ. For the example context from Table 5.1 these graphs are drawn in Figure 5.4. Note that the arc relation $\mathbf{E}_\phi$ is generally not identical to the transitive reduction of $\preceq$, which is usually used within diagrams illustrating a concept lattice: the concepts $\langle a, 234\rangle$ and $\langle abd, 4\rangle$ are joined by an arc although they are not direct successors with respect to $\preceq$. The basic idea for our sampling algorithm is to perform a random walk on these generator graphs. Taking just one of them, however, does not result in an irreducible chain. Either $\top$ or $\perp$ would be an absorbing state for such a process, in which any random walk will result with a probability converging to 1 for an increasing number of steps.

A first idea to achieve irreducibility might be to take one of the corresponding generator graphs, say $\mathbf{G}_\phi$, add all inverse edges of $\mathbf{E}_\phi$ as additional possible state transitions, and perform a random walk on the resulting strongly connected graph. Unfortunately, with this approach, drawing a neighbor of a given concept (as required for an efficient chain simulation) is as hard as the general problem of sampling a concept.

Proposition 5.2. Given *a context* $(A, O, \mathcal{D})$ *and a concept* $\langle I, E\rangle \in \mathcal{C}$, drawing *a predecessor of* $\langle I, E\rangle$ *in* $\mathbf{G}_\phi$ *uniformly at random, i.e., a concept* $\langle I', E'\rangle \in \mathcal{C}$ *with* $\phi(I' \cup a) = I$ *for some* $a \in A$*, is as hard as generating an arbitrary element of* $\mathcal{C}$ *uniformly at random.*

Proof. A given context $(A, O, \mathcal{D})$ with concepts $\mathcal{C}$ can be transformed into a context $(A', O', \mathcal{D}')$ with concepts $\mathcal{C}'$ such that A' is a successor of all concepts $C \in \mathcal{C}$ with respect to the closure operator ϕ' of $(A', O', \mathcal{D}')$. Moreover, this transformation can be computed in linear time. Hence, drawing a predecessor of A' is equivalent to drawing a concept $C \in \mathcal{C}$. The construction is as follows: set $A' = A \cup \{a^*\}$ with $a^* \notin A$, $O' = O \cup \{o^*\}$ with $o^* \notin O$, and

$$\mathcal{D}' = \mathcal{D} \cup \{(a, o^*) \colon a \in A'\} \ .$$

Then $\mathcal{C} = \mathcal{C}' \setminus \{\langle A', \{o^*\}\rangle\}$ and for all $\langle I, E\rangle \in \mathcal{C}$, it holds that $\phi'(I \cup \{a^*\}) = A'$, i.e., all $C \in \mathcal{C}$ are predecessors of $\langle A', \{o^*\}\rangle$ in $\mathbf{G}_{\phi'}$ as required. □

An alternative approach is to use the a random walk on the *union* of both generator graphs as stochastic process. Technically, this can be realized as follows: in a current state flip a fair coin and then, based on the outcome, choose either $\mathbf{G}_\phi$ or $\mathbf{G}_\psi$ to proceed to the next state as described above. This leads to the state transition probabilities

$$q(C, C') = \begin{cases} |G_\phi(I, I')| / (2\,|A|), & \text{if } C \prec C' \\ |G_\psi(E, E')| / (2\,|O|), & \text{if } C \succ C' \\ |I| / (2\,|A|) + |E| / (2\,|O|), & \text{if } C = C' \ . \end{cases} \tag{5.3}$$

for concepts $C = \langle I, E\rangle$ and $C' = \langle I', E'\rangle$. It is easy to see that the resulting stochastic process is an ergodic Markov chain. Thus, it has a stationary distribution to which it converges. Generally, however, we do not have a closed form of this distribution, and, clearly, it does not correspond to any interestingness measure. In the next subsection we show how the chain can be modified such that it converges to a distribution that is known and desired.

5.3.3. Metropolis-Hastings Algorithm

Let $\pi : \mathcal{C} \to [0, 1]$ be the desired target distribution, according to which we would like to sample concepts. For technical reasons that will become clear shortly we require π to be strictly positive, i.e., $\pi(C) > 0$ for all $C \in \mathcal{C}$. In practice this can be achieved easily (see the exemplary applications in Sec. 5.4). If our ergodic preliminary chain would have symmetric state transition probabilities, we could simply use it as *proposal chain* as follows: in a current concept C, propose a successor C' according to q (as defined in Eq. (5.3)), and then accept the proposal with probability $\pi(C')/\pi(C)$. This is the classic Metropolis algorithm, and for the resulting Markov chain it is easy to check that it satisfies the detailed balance condition for π—granted that q is symmetric as well as ergodic and that π is strictly positive. However, the example in Figure 5.4 shows that the union of the two generator graphs corresponding to a given context does not necessarily induce symmetric state transition probabilities. In fact, the resulting process is in general not even time-reversible—see for instance concepts $\perp$ and $\langle bc, 1\rangle$ in Figure 5.4, for which we have $q(\perp, \langle bc, 1\rangle) = 0$ but $q(\langle bc, 1\rangle, \perp) > 0$. As a solution one can factor the quotient of the proposal probabilities into the acceptance probabilities. The resulting state transitions are

$$p_\pi(C, C') = \begin{cases} q(C, C') \min\{\alpha \frac{\pi(C')}{\pi(C)}, 1\}, & \text{if } q(C, C') > 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.4}$$

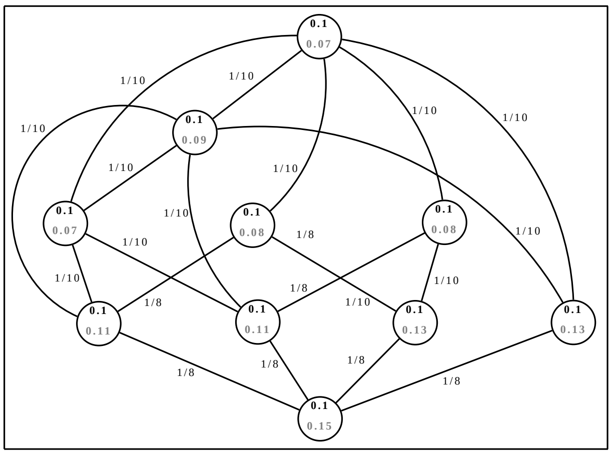


Figure 5.5.: Resulting state transitions for Table 5.1 and $\pi = u(\mathcal{C})$; remaining probabilities are assigned to self-loops; nodes contain stationary probability (black) and apx. probability after five steps with $X_0 = \langle \emptyset, O \rangle$ (gray).

where $\alpha = q(C', C)/q(C, C')$. This is the Metropolis-Hastings algorithm [Hastings, 1970], and the underlying process is called the **Metropolis-process** of q and π. For the example from Table 5.1 its state transition probabilities are drawn in Figure 5.5. It is important to note that, in order to simulate the Metropolis process, one does not need the exact probabilities $\pi(C')$ and $\pi(C)$. As the acceptance probability only depends on their quotient, it is sufficient to have access to an **unnormalized potential** of π, i.e., a mapping $\tilde{\pi} : \mathcal{C} \to \mathbb{R}$ such that there is a constant α with $\tilde{\pi}(C) = \alpha\pi(C)$ for all $C \in \mathcal{C}$. For instance, for the uniform target distribution one can choose any constant function for $\tilde{\pi}$. Algorithm 10 is a Markov chain Monte Carlo implementation of the Metropolis process. It takes as input a context, a number of iterations s, and an oracle for the unnormalized potential $\tilde{\pi}$. Then, after choosing an initial state X_0 uniformly among $\perp$ and $\top$, it simulates the process for s steps, and returns the state it has reached by that time, i.e., its realization of X_s.

It is easy to see that the algorithm indeed uses the state transition probabilities p_π. So far, however, we have omitted an important aspect of its correctness: while π satisfies the detailed balance condition for this chain, this comes at the cost of setting $p_\pi(C, C') = 0$ for some pairs of concepts that have $q(C, C') > 0$. Thus, the irreducibility of p_π is not directly implied by the irreducibility of q. It is, however, guaranteed by the closure properties of ϕ and ψ, as shown in the proof of Theorem 5.3 below. In contrast to the distributional convergence, it is straightforward to show that the single steps of the induced Markov chains can be simulated efficiently: the computation time is dominated by the calculation of the acceptance probability α, which can be computed by $|O| + |A|$ closure computations. However, one can do better than this naive $O((|O| + |A|)\,|\mathcal{D}|)$ solution. In fact, the following summarizing theorem holds.

Theorem 5.3. *On input context $(A, O, \mathcal{D})$, step number s, and strictly positive function $\tilde{\pi}$, Algorithm 10 produces in time $O(s\,|A|\,|O|)$ a concept $C \in \mathcal{C}(A, O, \mathcal{D})$ according to a distribution p_π^s such that*

$$\lim_{s \to \infty} \|p_\pi^s, \pi\|_{tv} = 0$$

Algorithm 10 Metropolis-Hastings Concept Sampling

```
Input  : context $(A, O, \mathcal{D})$, number of iterations $s$, oracle of map $\tilde{\pi} : \mathcal{C} \to \mathbb{R}_+$
Output : concept $\langle I, E \rangle$
1.  init $\langle I, E \rangle \sim u(\{\top, \bot\})$ and $i \leftarrow 0$
2.  $i \leftarrow i + 1$
3.  draw $d \sim u(\{\text{up}, \text{down}\})$
4.  if $d = \text{up}$ then
5.      draw $a \sim u(A)$
6.      $\langle I', E' \rangle \leftarrow \langle \phi(I \cup \{a\}), O[\phi(I \cup \{a\})] \rangle$
7.      $\alpha \leftarrow (|G_\psi(E', E)|\,|A|) \,/\, (|G_\phi(I, I')|\,|O|)$
8.  else
9.      draw $o \sim u(O)$
10.     $\langle I', E' \rangle \leftarrow \langle A[\psi(E \cup \{o\})]\,, \psi(E \cup \{o\}) \rangle$
11.     $\alpha \leftarrow (|G_\phi(I', I)|\,|O|) \,/\, (|G_\psi(E, E')|\,|A|)$
12. draw $x \sim u([0, 1])$
13. if $x < \alpha \tilde{\pi}(I') \,/\, \tilde{\pi}(I)$ then $\langle I, E \rangle \leftarrow \langle I', E' \rangle$
14. if $i = s$ then return $\langle I, E \rangle$ else goto 2
```

where π is the distribution on $\mathcal{C}$ resulting from normalizing $\tilde{\pi}$, i.e., $\pi(\cdot) = \tilde{\pi}(\cdot) / \sum_{C \in \mathcal{C}} \tilde{\pi}(C)$.

Proof. For the time complexity we show that the single step complexity for the case $d = \text{up}$ is in $O(|A|\,|O|)$. The other case has the same complexity by symmetry. The complexity is dominated by the computation of the acceptance probability $\alpha = (|G_\psi(E', E)|\,|A|) \,/\, (|G_\phi(I, I')|\,|O|)$. For self-loops this probability is readily available as $(|E|\,|A|)/(|I|\,|O|)$. Thus, consider the case $\langle I, E \rangle \neq \langle I', E' \rangle$. One can check that in this case it holds that

$$\begin{aligned} G_\phi(I, I') &= \{a \in (I' \setminus I) \colon \forall o \in (E \setminus E'), (a, o) \notin \mathcal{D}\} \\ G_\psi(E', E) &= \{o \in (E \setminus E') \colon \forall a \in (I' \setminus I), (a, o) \notin \mathcal{D}\} \ . \end{aligned}$$

These sets can be counted in time $O(|A|\,|O|)$ by checking for all pairs $(a, o) \in (I' \setminus I) \times (E \setminus E')$ once whether $(a, o) \in \mathcal{D}$.

Regarding the distributional convergence, by construction, the algorithm simulates the Metropolis process governed by the transition probabilities p_π defined in Equation (5.4) of π and q (Eq. (5.3)). It can directly be checked that π and p_π satisfy the detailed balance condition (Eq. 5.1). It remains to show irreducibility, i.e., $p^t_\pi(\langle I, E \rangle, \langle I', E' \rangle) > 0$ for some t and all pairs of concepts $\langle I, E \rangle, \langle I', E' \rangle \in \mathcal{C}$. It is sufficient to consider the case $\langle I, E \rangle \preceq \langle I', E' \rangle$: for other states reachability follows then via $\langle I \cap I', A[I \cap I'] \rangle \preceq \langle I, E \rangle$, $\langle I \cap I', A[I \cap I'] \rangle \preceq \langle I', E' \rangle$, the transitivity of the reachability relation, and the fact that per definition it holds that $p_\pi(C, C') > 0$ if and only if $p_\pi(C', C) > 0$ for concepts $C, C' \in \mathcal{C}$. Moreover, as the considered state spaces $\mathcal{C}$ are finite, it suffices to consider direct successors $\langle I, E \rangle \,\dot{\preceq}\, \langle I', E' \rangle$. For such concepts it is easy to show that I is a maximal proper subset of I' in $\phi(\mathcal{P}(A))$ and E' is a maximal proper subset of E in $\psi(\mathcal{P}(O))$. Let $a \in I' \setminus I$ and $o \in E \setminus E'$. It follows by the closure operator properties that $\phi(I \cup \{a\}) = I'$ and $\psi(E' \cup \{o\}) = E$. Thus, the proposal probabilities between $\langle I, E\rangle$ and $\langle I', E'\rangle$ are non-zero in both directions, and together with the fact that $\tilde{\pi}$ is strictly positive, it follows for the transition probability of the Metropolis-process that $p_\pi(\langle I, E\rangle, \langle I', E'\rangle) > 0$ as required. □

That is, our algorithm asymptotically draws samples as desired and a single step can be computed efficiently. Also recall that Markov chain convergence is always monotonic. Hence, we are getting closer to the desired target distribution the greater we choose the parameter s. Still it remains to discuss for how many steps one has to simulate the process until the distribution is "close enough"—Theorem 5.3 does not answer this question.

5.3.4. Number of Iterations

The minimum number of simulation steps getting us ϵ-close to the stationary distribution is the mixing time $\tau(\epsilon)$ of the chain. Unfortunately, in the worst case the mixing can be infeasible slow, i.e., it requires a number of steps that can grow exponentially in the input size for any fixed $\epsilon > 0$. The following statement on the mixing time holds.

Proposition 5.4. *Let $\epsilon > 0$ be fixed and $\tau_n(\epsilon)$ denote the worst-case mixing time of Algorithm 10 for the uniform target distribution and an input context with n attributes and n objects (hence, of size at most n^2). Then $\tau_n(\epsilon) \in \Omega(2^{n/2})$.*

Proof. We prove the claim by showing that for all even n there is a context $(A, O, \mathcal{D})$ with $A = O = \{1, \ldots, n\}$ (and $|\mathcal{D}| = n^2/2 - n$) such that $\tau(\epsilon) > 2^{n/2-3}\log(1/(2\epsilon))$. Similar to the example in Section 5.2.3 this can be achieved using the conductance technique. Choose

$$\mathcal{D} = \{(i,j) \colon i \neq j, (i,j \leq n/2 \vee i,j > n/2)\} .$$

Then the set of concepts is of size $2^{n/2+1} + 2$ and can be expressed as a disjoint union $\mathcal{C}(A, O, \mathcal{D}) = \mathcal{C}_1 \dot{\cup} \mathcal{C}_2$ with

$$\begin{aligned}\mathcal{C}_1 =& \{\langle I, O[I]\rangle \colon I \subseteq \{1, \ldots, n/2\}\} \\ \mathcal{C}_2 =& \{\langle I, O[I]\rangle \colon \emptyset \neq I \subseteq \{n/2+1, \ldots, n\}\} \cup \{\top\}\end{aligned}$$

with $|\mathcal{C}_1| = |\mathcal{C}_2| = |\mathcal{C}|/2$. Hence, for the uniform target distribution $\pi = u(\mathcal{C})$, it holds that $\pi(\mathcal{C}_1) = 1/2$. Moreover, the only state pairs that contribute to the conductance $\Phi_{\mathcal{C}_1}$ are the n pairs $\{\bot, C\}$ and $\{C', \top\}$ of the form

$$\begin{aligned}C &\in \{\langle I, E\rangle \colon I \in \{\{n/2+1\}, \ldots, \{n\}\}\} \\ C' &\in \{\langle A \setminus \{a\}, E\rangle \colon a \in \{1, \ldots, n/2\}\} .\end{aligned}$$

For all such pairs one can check that the proposal probability is $2/n$ whereas the return proposal probability is $(n/2+1)/2n$, and, hence, the corresponding transition probabilities are less than $2/n$. This implies for the conductance

$$\begin{aligned}\Phi \leq \Phi_{\mathcal{C}_1} &= \frac{1}{\pi(\mathcal{C}_1)} \sum_{C \in \mathcal{C}_1, C' \in \mathcal{C}_2} \underbrace{\pi(C)}_{1/|\mathcal{C}|} \underbrace{p_\pi(C, C')}_{<2/n} \\ &< 2n \frac{1}{2^{n/2+1} + 2} \frac{2}{n} \\ &< \frac{1}{2^{n/2-1}}\end{aligned}$$

Using Theorem 5.1 it follows for the mixing time that $\tau(\epsilon) \geq 2^{n/2-3} \log(1/(2\epsilon))$ as required. $\square$

Thus, even the strictest theoretical worst-case bound on the mixing time for general input contexts would not lead to an efficient algorithm. The proof of the proposition is based on the observation that the chain can have a very small conductance, i.e., a relatively large portion of the state space (growing exponentially in n) can only be connected via a linear number of states to the rest of the state space. We assume that real-world datasets do not exhibit this behavior. In our applications we therefore use the following polynomially bounded heuristic function for assigning the number of simulation steps:

$$\text{steps}((A, O, \mathcal{D}), \epsilon) = 4n \ln(n) \ln(\epsilon) \tag{5.5}$$

where $n = \min\{|A|, |O|\}$. The motivation for this function is as follows. Assume without loss of generality that $|A| < |O|$. As long as only "$d =$ up" is chosen in line 3 of Algorithm 10 the expected number of steps until all elements of A have been drawn at least once is $n \ln(n) + O(n)$ with a variance that is bounded by $2n^2$ (coupon collector's problem). It follows that asymptotically after $2n \ln(n)$ all elements have been drawn with probability at least $3/4$. Consequently, as "$d =$ up" is chosen with probability $1/2$, we can multiply with an additional factor of 2 to know that with high probability there was at least one step upward and one step downward for each attribute $a \in A$—if one assumes a large conductance this suffices to reach every state with appropriate probability. The final factor of $\ln(\epsilon)$ is motivated by the fact that the total variation distance must decay exponentially in the number of simulation steps for a fixed problem instance.

This can also be observed in the experimental results presented in Figure 5.6. They include four real-world datasets [1] and two target distributions: the uniform distribution and the distribution proportional to the area function $\pi(\cdot) = q_{\text{area}}(\cdot)/Z$. Note that the plots differ in their semantics. While for both target distributions a point (x, y) means that the x-step distribution has a total variation distance of y from the target distribution, for the heuristic the inverse of the step-heuristic is shown, i.e., the value of ϵ on the y-axis reflects the desired accuracy that leads to a particular number of steps. Thus, the heuristic is correct if its corresponding plot dominates the two total variation distance plots. As we can see, this is the case for both distributions on all four datasets. In fact, the heuristic is rather conservative for three of the four datasets ("mushroom", "retail", and "chess"). Note that the y-axis has a logarithmic scale.

The effect of the target distribution on the mixing time is unclear. On the one hand, deviation from the uniform distribution can create bottlenecks, i.e., low conductance sets, where there have been non before. On the other hand, it can also happen that bottlenecks of the uniform distribution are softened. This can be observed for "mushroom" and "chess". For that reason we do not reflect the target distribution in the heuristic.

[1] Databases as before from UCI Machine Learning Repository [Asuncion and Newman, 2007]. In order to allow explicit computation of the state transition matrices only a sample of the transactions is used for three of the databases. The sample size is given in brackets.

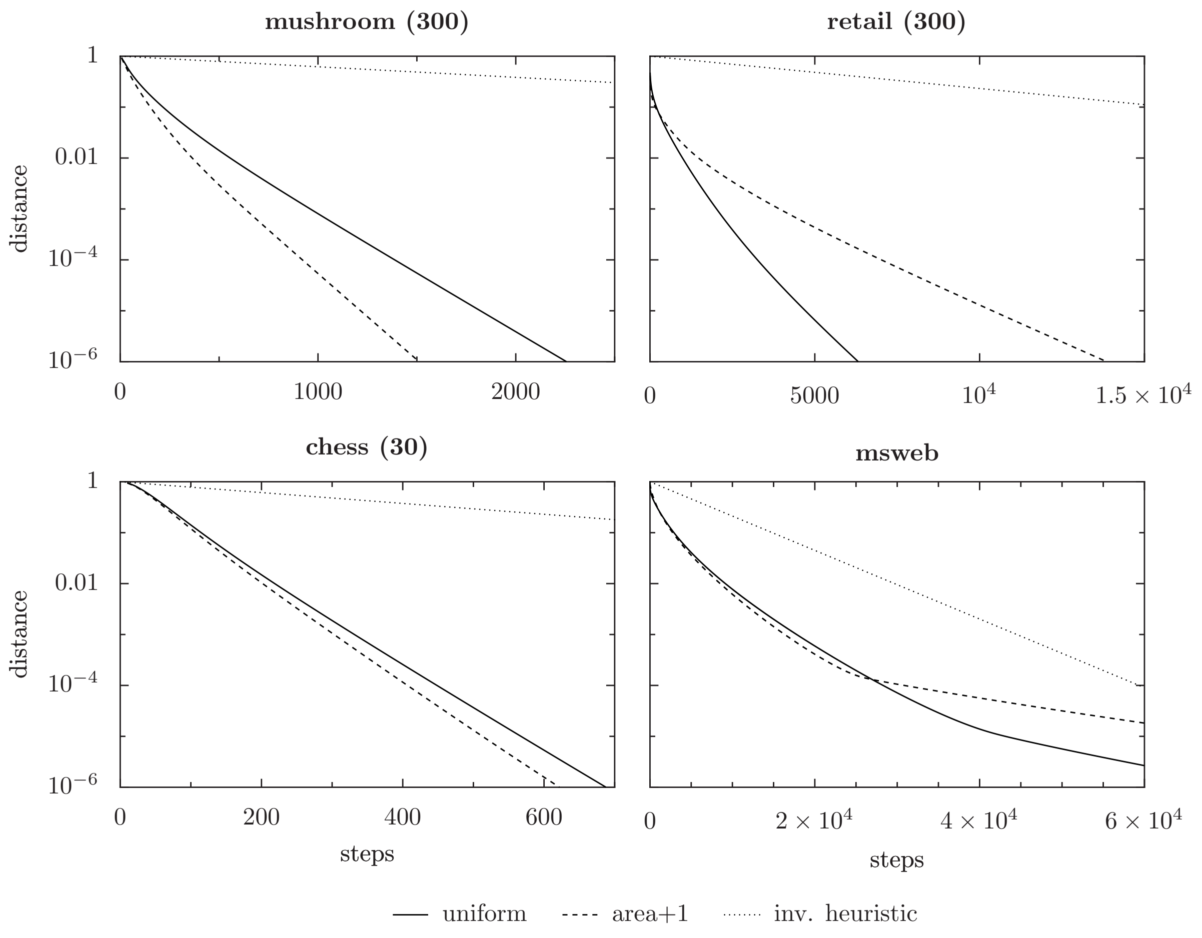


Figure 5.6.: Distribution convergence and the inverse heuristic function, i.e., the closeness to stationarity that the heuristic "assumes" for a given number of steps. The heuristic is feasible as long as the inverse heuristic dominates the total variation distance of the t-step distributions.

5.4. Exemplary Applications

In this section, we present exemplary applications of the closed set sampling algorithm to local pattern discovery. This revisits the motivations from Section 5.1 on controlled repeated pattern sampling. Specifically, we attack the problem of minimal non-redundant association rule discovery on the relatively demanding datasets from the introductory Chapter 1 (p. 4) as well as the non-redundant subgroup discovery problem with minimum generators (p. 29) for the most challenging among the datasets of Chapter 2.

5.4.1. Association Rule Discovery

We start with the presentation of the application of discovering minimal non-redundant association rules. Just as in Chapter 1, for the sake of simplicity, we consider exact association rules, i.e., rules with a confidence of 1. Sampling a rule with confidence below 1 can be done,

	Election	Questio.	Census
sampling wrt. q_{supp}	2.1 sec.	21 min.	7 min.
sampling wrt. q_{area}	2.2 sec.	21 min.	7 min.
JClose...	59 min.	7500 m.	1729 min.
... with threshold	0.35	0.7	0.7

Table 5.2.: Time for randomized association rule generation compared to exhaustive generation using the frequency thresholds from the motivational experiment from page 4.

for instance, by sampling the consequent of the rule in a first step; and then sampling the antecedent according to a distribution that is proportional to the confidence and support of the resulting rule. One can generate exact minimal non-redundant association rules by first sampling a closed set C, and then calculating a minimal generator G of C, i.e., a minimal set with $\sigma(G) = C$. By Proposition 2.5 the resulting rule $\sigma(G) \to C \setminus \sigma(G)$ is a minimal non-redundant rule. For the calculation of the minimal generators, one can use the simple greedy approach from Chapter 2 (Algorithm 3) that is essentially equivalent to the greedy algorithm for the set cover problem. In fact, with this algorithm one generates an approximation to a *shortest* generator and not only a *minimal* generator with respect to set inclusion. Thus, we are aiming for particularly short, i.e., general, rule antecedents.

With this method we generate association rules for the three datasets "Election", "Questionnaire", and "Census" from Section 1.1 (see Table 1.2) for two different target distributions: the one proportional to the rule's *support* and the one proportional to their *area* (adding the constant 1 to ensure a strictly positive distribution). That is, we used $\tilde{\pi}(\cdot) = q_{\mathrm{supp}}(\cdot)+1$ respectively $\tilde{\pi}(\cdot) = q_{\mathrm{area}}(\cdot) + 1$ (see Section 3.1) as parameter for Algorithm 10. As desired closeness to the target distribution we choose $\epsilon = 0.01$. Note that, in contrast to exhaustive listing algorithms, for the sampling method we do not need to define any threshold like minimum support. Table 5.2 shows the time required to generate a single rule according to q_{supp} and q_{area}. In addition, the table shows the runtime of JClose for exemplary support thresholds. Comparing the computation times of sampling with that of JClose, we can see that the uniform generation of a rule from the Election dataset takes about 1/1609 of the time needed to exhaustively list non-redundant association rules with threshold 0.35. For Questionnaire and threshold 0.7, the ratio is 1/340; finally for Census with threshold 0.7, the ratio is 1/247. As discussed in Section 1.1, all exhaustive listing algorithms exhibit a similar asymptotic performance behavior. Thus, for methods that outperform JClose on these particular datasets, we would end up with similar ratios for slightly smaller support thresholds. Moreover, it is not uncommon that one is interested in a small set of rules having a substantially lower support than the thresholds considered above. For such cases the ratios would increase dramatically, because the time for exhaustive listing increases exponentially whereas the time for sampling remains constant. Thus, for such applications sampling has the potential to provide a significant speedup. In fact, it is applicable in scenarios where exhaustive listing is hopeless.

5.4.2. Supervised Descriptive Rule Induction

As second application we consider *supervised descriptive rule induction*, i.e., local pattern mining considering labeled data such as emerging pattern mining, contrast set mining, or subgroup discovery (see Sec. 2.1; p. 17). Specifically, we reconsider the non-redundant subgroup discovery problem asking to find minimum generators of high binomial test quality that are pairwise non-equivalent. This can be done by sampling a closed set C with Algorithm 10 choosing the binomial test quality function q_{bino} as parameter, and, subsequently, by computing an approximated minimum generator of C using Algorithm 3 as above for the association rules. In fact, one has to cap q_{bino} from below by a small constant $c > 0$ to ensure a strictly positive target distribution, i.e., use $\tilde{\pi}(\cdot) = \max\{q_{\text{bino}}(\cdot), c\}$. With the binomial test quality function we have an evaluation metric that has shown to adequately measure interestingness in several applications. Thus, in addition to the computation time, in this experiment we can also evaluate the quality of the sampled patterns.

We consider the three labeled datasets "sick", "soybean", and "lung-cancer" from the UCI machine learning repository that turned out to be the most time demanding in the experiment on non-redundant subgroup discovery (see Table 2.6; p. 35). As baseline method we use the exhaustive breadth-first minimum generator listing approach (Algorithm 4). As discussed in Chapter 2, this algorithm enumerates potentially interesting support equivalence classes based on an optimistic estimator $\hat{q}_{\text{bino}}$ and outperforms the (non equivalence class based) depth-first set listing algorithms on the considered datasets. Table 5.3 gives the time needed to generate a pattern by the randomized algorithm compared to the time for exhaustively listing all patterns with a binomial quality of at least 0.1. This threshold is approximately one third of the binomial quality of the best patterns in all three datasets. As observed in the unsupervised setting, the randomized approach allows to generate a pattern in a fraction of the time needed for the exhaustive computation.

	sick	soyb.	lung.
sampling time	3	1	0.02
listing time	1460	705	69

Table 5.3.: Time (in seconds) of random pattern generation according to q_{bino} compared to exhaustive listing.

In order to evaluate the pattern quality, we compare the best patterns within a set of 100 samples to the globally best patterns (Table 5.4). Although the quality of the randomly generated patterns do not reach the optimum, we observe the anticipated and desired result: the collected sets form a high quality sample relatively to the complete pattern space, from which they are drawn.

5.5. Approximate Counting and Complexity

In this section we highlight the connection of concept sampling to approximate concept counting. In particular, we show how concept sampling can be used to design a randomized approximation scheme for the number of concepts of a given context. That is, we give an algorithm for the following problem.

Problem 5.2 (COUNT-CLOSED-SETS)**.** *Given* a dataset $\mathcal{D}$ over E, *compute* the cardinality $|\mathcal{C}(\mathcal{D})|$ of the corresponding closed set family. $\Diamond$

The connection between pattern sampling and approximate pattern counting is important for at least two reasons: an efficient counting algorithm can be used to quickly check the feasibility of a potentially exponential time exhaustive pattern listing algorithm, because exhaustive listing takes time at least proportional to the number of patterns and for amortized linear time enumeration algorithms the listing time is also at most proportional to that number. Moreover, the connection of sampling and counting ties the computational complexities of these two tasks together.

Thus, following the presentation of an approximation scheme for the number of closed sets, we investigate the computational complexity of closed and frequent set counting problems. In particular, the following problem turns out to be exceptionally hard in the sense that even approximating the order of magnitude of a solution is intractable.

Problem 5.3 (COUNT-FREQUENT-SETS)**.** *Given* a dataset $\mathcal{D}$ over E and a frequency threshold f, *compute* the cardinality $|\mathcal{F}(\mathcal{D}, f)|$ of the corresponding frequent set family. $\Diamond$

By a straightforward reduction we can see that this implies the same hardness result for counting closed frequent sets, i.e., for adding a minimum frequency parameter to Problem 5.2.

5.5.1. Preliminaries

Prior to the technical discussion, we need to fix some notions and notation related to the randomized approximation of counting problems and their computational complexity. For further reading on the reducibility between counting problems we refer to Dyer et al. [2004]. A general introduction to computational complexity, specifically to the hardness of approximation, can be found in the textbook of Arora and Barak [2009].

A **bounded probability (BP) algorithm** for a computational problem with problem instances X and possible solutions Y specified by a correctness relation $R \subseteq X \times Y$ is a probabilistic algorithm $\mathfrak{A}$ such that it holds for all $x \in X$ that $\mathbb{P}_{\mathfrak{A}(x)}[(x, \mathfrak{A}(x)) \in R] \geq 3/4$ where $\mathfrak{A}(x)$ denotes the randomized output of $\mathfrak{A}$ on input x. For a function $f : \mathbb{N} \to \mathbb{N}$ the class **BPTIME**$(f(n))$ denotes all decision problems, i.e., problems with $Y = \{0, 1\}$, that can be solved by a BP-algorithm with a time complexity in $O(f(n))$. Now we consider the case where Y is the set of natural numbers $\mathbb{N}$. A **randomized approximation scheme (RAS)** for a mapping $g : X \to \mathbb{N}$ is a BP-algorithm $\mathfrak{A}$ taking arguments $x \in X$ and $\epsilon \in (0, 1/2]$

	sick	soybean	lung.
best quality	0.177	0.223	0.336
sampled quality	0.167	0.187	0.263

Table 5.4.: Best binomial test quality among all patterns versus best quality among sampled sets.

satisfying the relaxed correctness predicate $R_\epsilon = \{(x,y)\colon (1-\epsilon)g(x) \leq y \leq (1+\epsilon)g(x)\}$ (this is sometimes just paraphrased "y is ϵ-close to $g(x)$"), i.e., $\mathfrak{A}$ satisfies

$$\mathbb{P}[(1-\epsilon)g(x) \leq \mathfrak{A}(x,\epsilon) \leq (1+\epsilon)g(x)] \geq 3/4 \ . \tag{5.6}$$

Such an algorithm is called **fully polynomial RAS (FPRAS)** if its time complexity is bounded polynomially in size(x) and $1/\epsilon$. The constant 3/4 appearing in the definition has no significance other than being strictly between 1/2 and 1. Any two success probabilities from this interval can be reached from one another by a small number of repetitions of the corresponding algorithm and returning the median of the results (see Jerrum et al. [1986]). A weaker notion of approximation is given by the following definition: An algorithm $\mathfrak{A}$ is called an **α-factor approximation** of g (or said to "approximate g within α") if it satisfies the relaxed correctness relation $R_\alpha = \{(x,y)\colon g(x)/\alpha(x) \leq y \leq \alpha(x)g(x)\}$ where $\alpha\colon X \to \mathbb{R}$ is a function that may grow in the size of x. Clearly, an efficient approximation scheme can act as an efficient c-factor approximation algorithm for all constants $c > 1$.

When considering counting problems, a central complexity class is **#P** [Valiant, 1979], which, intuitively, consists of all problems asking to count the number of solutions where it is possible in polynomial time to verify a single solution. Formally, a function $f\colon X \to \mathbb{N}$ is in **#P** if there is a relation $R \subseteq X \cup Y$ that is polynomially balanced, i.e., $\max_{y \in R[x]} \text{size}(y) \leq \text{poly}(\text{size}(x))$, that satisfies $|R[x]| = f(x)$ for all $x \in X$, and that is verified by a polynomial time algorithm $\mathfrak{A}$, i.e., for all $(x,y) \in X \times Y$ it holds that $\mathfrak{A}(x,y) = 1$ if and only if $(x,y) \in R$. Although the existence of an exact polynomial time solver for a **#P**-hard problem implies **P** = **NP**,[2] there are fully polynomial time approximation schemes for some of them, e.g., for the problem of computing the number of satisfying assignments of a DNF formula [Karp et al., 1989] or for computing the permanent of matrices with non-negative entries [Jerrum et al., 2004]. Hence, proving **#P**-hardness for a counting problem is not sufficient to rule out the existence of a corresponding FPRAS.

However, no FPRAS is known for a problem **#P**-hard with respect to reductions that are approximation-preserving (see Dyer et al. [2004]; definition is also given below). Moreover, Zuckerman [1996] has shown that the existence of an FPRAS for a problem **#P**-hard under AP-reduction implies **NP** = **RP**, and it is widely believed that this is not the case. An **approximation-preserving (AP) reduction** from a function $f : X \to \mathbb{N}$ to a function $g : X' \to \mathbb{N}$ is a probabilistic algorithm $\mathcal{R}$ with possible inputs from $X \times (0, 1/2]$ having access to an oracle $\mathcal{O}$ with possible inputs from $X' \times (0, 1/2]$ such that the following holds. There is a polynomial p such that for all inputs (x, ϵ) to $\mathcal{R}$ all performed calls $\mathcal{O}(x', \epsilon')$ satisfy $1/\epsilon' \leq p(1/\epsilon, \text{size}(x))$, and $\mathcal{R}$ is a RAS for f whenever $\mathcal{O}$ is a RAS for g. Note that parsimonious reductions [Simon, 1977], i.e., reductions that preserve the number of solutions, are a special case of AP-reductions. In particular, f has an FPRAS if g has one and f is AP-reducible to g.

5.5.2. Counting Concepts

A simple Monte Carlo approach for counting the number of closed sets is the following: uniformly generate an element $F \subseteq E$, return 1 if F is closed, and return 0 otherwise. The

[2]In fact, polynomial time algorithms equipped with an oracle for a **#P**-complete problem are able to solve any problem in the polynomial hierarchy (see Arora and Barak [2009, pp. 352-353]).

expected value of this experiment is $|\mathcal{C}|/2^{|E|}$. Thus, taking the mean of sufficiently many independent repetitions and multiplying it by $2^{|E|}$ is a correct randomized approximation scheme. It is, however, not polynomial (assume a fixed accuracy parameter). This is due to the fact that $|\mathcal{C}|/2^{|E|}$ can be as small as $1/2^n$ for an instance of size n. For such instances the expected number of trials before the first 1-turn-out appears is not bounded by a polynomial in n. But as long as all returned results are 0, the combined solution is also 0, which does not satisfy any relative approximation guarantee; in particular not Equation (5.6). The standard solution to this problem is to express the counting result by a number of factors, each of which having a reasonable lower bound (see Jerrum and Sinclair [1997]). In our case, such a factorization can be done as follows.

Let $(A, O, \mathcal{D})$ be a context and $o_1 \ldots, o_m$ some ordering of the elements of O. For $i \in \{0, \ldots, m\}$ define the context $\mathbb{C}_i = (A, O_i, \mathcal{D}_i)$ with $O_i = \{o_j : j \leq i\}$, $\mathcal{D}_i$ as the restriction of $\mathcal{D}$ to O_i, ϕ_i as the corresponding attribute closure operator, and $\mathcal{I}_i = \phi_i(\mathcal{P}(A))$ the concept intents. Note that in general a concept intent $I \in \mathcal{I}_{i+1}$ is not an intent of a concept with respect to the context $\mathbb{C}_i$, i.e., not a fixpoint of ϕ_i—for instance for the context of Table 5.1, $\{d\}$ is a fixpoint of ϕ_5, but not of ϕ_4. The following two properties, however, hold.

Lemma 5.5. *For all contexts $(A, O, \mathcal{D})$ and all $i \in \{0, \ldots, |O|\}$ it holds that (i) $\mathcal{I}_i \subseteq \mathcal{I}_{i+1}$ and (ii) $1/2 \leq |\mathcal{I}_i| / |\mathcal{I}_{i+1}|$.*

Proof. For (i) let $I \in \mathcal{I}_i$. In case $o_{i+1} \notin I[O_{i+1}]$, it is $I[O_{i+1}] = I[O_i]$. Otherwise, per definition we know that all $a \in A$ that satisfy $(a, o) \in \mathcal{D}$ for all $o \in O_i$ are also satisfying $(a, o) \in \mathcal{D}$ for all $o \in O_{i+1}$. Thus, in both cases $\phi_{i+1}(I) = A[O_{i+1}[I]] = A[O_i[I]] = \phi_i(I) = I$ as required.

We prove (ii) by showing that the restriction of the closure operator ϕ_i to $\mathcal{I}_{i+1} \setminus \mathcal{I}_i$ is an injective map into $\mathcal{I}_i$, i.e., $|\mathcal{I}_{i+1} \setminus \mathcal{I}_i| \leq |\mathcal{I}_i|$. Together with (i) this implies the claim. Note that per definition, $\phi_i(\mathcal{I}_{i+1} \setminus \mathcal{I}_i) \subseteq \phi_i(\mathcal{P}(A)) = \mathcal{I}_i$. Regarding injectivity, assume for a contradiction that there are distinct intents $X, Y \in \mathcal{I}_{i+1} \setminus \mathcal{I}_i$ such that $\phi_i(X) = \phi_i(Y)$. Then $O_i[X] = O_i[Y]$, and, as X and Y are both closed with respect to ϕ_{i+1} but not closed with respect to ϕ_i, it must hold that $o_{i+1} \in O_{i+1}[X] \cap O_{i+1}[Y]$. It follows that $O_{i+1}[X] = O_i[X] \cup \{o_{i+1}\}$ as well as $O_{i+1}[Y] = O_i[Y] \cup \{o_{i+1}\}$. This implies $O_{i+1}[X] = O_{i+1}[Y]$ and in turn $X = Y$ contradicting the assumption that the intents are distinct. □

Thus, the $\mathcal{I}_i$ define an increasing sequence of closed set families with $|\mathcal{I}_0| = |\{A\}| = 1$ and $|\mathcal{I}_m| = |\mathcal{C}(A, O, \mathcal{D})|$. These families allow to express the number of concepts of the complete context by the "telescoping product"

$$|\mathcal{C}(A, O, \mathcal{D})| = \left(\frac{1}{|\mathcal{I}_o|} \prod_{i=1}^{m} \frac{|\mathcal{I}_{i-1}|}{|\mathcal{I}_i|} \right)^{-1} \tag{5.7}$$

such that each factor $\mathcal{I}_i/\mathcal{I}_{i+1}$ is lower bounded by 1/2. This expression reduces the task of approximating $|\mathcal{C}|$ to estimating the ratios $|\mathcal{I}_i| / |\mathcal{I}_{i+1}|$. Similarly to the naive approach described above, this can be done by sampling concepts corresponding to the context $\mathbb{C}_{i+1}$ and checking whether they are also concepts with respect to $\mathbb{C}_i$.

Formally, for $i \in \{1, \ldots, m\}$ let $Z_i(I)$ denote the binary random variable on probability space $(\mathcal{I}_i, u'_i)$ that takes on value 1 if $I \in \mathcal{I}_{i-1}$ and 0 otherwise. If the distributions u'_i are close to uniform, independent simulations of the random variables Z_i can be used to count the number of concepts via Equation 5.7. Namely, the number of concepts is approximated by the product estimator $Z = \prod_1^m \bar{Z}_i$ where $\bar{Z}_i$ is defined by

$$\bar{Z}_i = (Z_i^{(1)} + \cdots + Z_i^{(t)})/t \tag{5.8}$$

with $Z_i^{(j)}$ denoting independent copies of Z_i. Using standard reasoning (see, e.g., Jerrum and Sinclair [1997]) and Lemma 5.5 one can show that Z^{-1} is ϵ-close to $|\mathcal{C}(A, O, \mathcal{D})|$ with probability at least 3/4 if

(i) for the total variation distance between the distributions u'_i and the uniform distributions on $\mathcal{I}_i$ it holds that $\|u'_i, u(\mathcal{I}_i)\|_{\text{tv}} \leq \epsilon/(12\,|O|)$ and

(ii) for the number of trials t in Equation (5.8), i.e., the number of independent copies of the Z_i that are averaged, it holds that $t \geq 12\,|O|\,/\epsilon^2$.

Granted that the step heuristic (Eq. (5.5), p. 99) is correct for the input context $\mathbb{C}$ and all its sub-contexts $\mathbb{C}_i$, we can use this heuristic within the Markov chain Monte Carlo sampler (Alg. 10) for simulating the counting estimator Z. See Algorithm 11 for a pseudocode.

Algorithm 11 Monte Carlo Concept Counting

```
Input   : context $\mathbb{C} = (A, O, \mathcal{D})$, accuracy $\epsilon \in (0, \frac{1}{2}]$
Require: $\|p^s_{i,u(\mathcal{C}_i)}, u(\mathcal{C}_i)\|_{\text{tv}} \leq \epsilon'$ for all $i \in \{1, \dots, |O|\}$ and $\epsilon' \in (0, \frac{1}{2}]$ where $s = \text{steps}(\mathbb{C}_i, \epsilon')$
         and $p_i$ the Metropolis transition probabilities (Eq. (5.4), p. 95) for context $\mathbb{C}_i$
Output : random real $a$ with $\mathbb{P}[(1-\epsilon)\,|\mathcal{C}| \leq a \leq (1+\epsilon)\,|\mathcal{C}|] \geq 3/4$

1. $t \leftarrow 12\,|O|\,/\epsilon^2$
2. for $i = 1, \dots, |O|$ do
3.    $r_i \leftarrow 0$
4.    for $k = 1, \dots, t$ do
5.       $\langle I, E\rangle \leftarrow$ sample$(\mathbb{C}_i, \text{steps}(\mathbb{C}_i, \epsilon/(12\,|O|)), C \mapsto 1)$ //see Alg. 10 (p. 97)
6.       if $\phi_{i-1}(I) = I$ then $r_i \leftarrow r_i + 1$
7.    $r_i \leftarrow r_i/t$
8. return $\prod_{i=1}^{|O|} r_i^{-1}$
```

Theorem 5.6 below emphasizes the general reducibility of counting to sampling independent of any heuristic (or the concrete sampling algorithm). Hence, the time required for sampling is an explicit parameter of this result. The precise time complexity stated below follows from the observation that the roles of A and O are interchangeable. That is, the number of concepts of $(A, O, \mathcal{D})$ is equal to the number of concepts of the "transposed" context $(O, A, \mathcal{D}^{-1})$. Hence, we can conclude:

Theorem 5.6. *There is a randomized approximation scheme for the number of concepts $|\mathcal{C}|$ of a given context $(A, O, \mathcal{D})$ with time complexity $O\big(n\epsilon^{-2}T_S(\epsilon/(12n))\big)$ for accuracy ϵ where $n = \min(|A|, |O|)$ and $T_S(\epsilon')$ is the time required to sample a concept almost uniformly, i.e., according to a distribution with a total variation distance of at most ϵ' from uniform.*

In order to evaluate the accuracy of this randomized counting approach, consider the results of a series of computations for the dataset "chess" presented in Figure 5.7. The randomized counting estimates are compared with the exact number of concepts computed by exhaustive enumeration. In order to be able to exhaustively compute the exact number of concepts, samples of different sizes were used instead of the whole dataset. Moreover, the accuracy was set to $\epsilon = 1/2$. The x-axis gives the size of the sample, while the y-axis shows the exact number

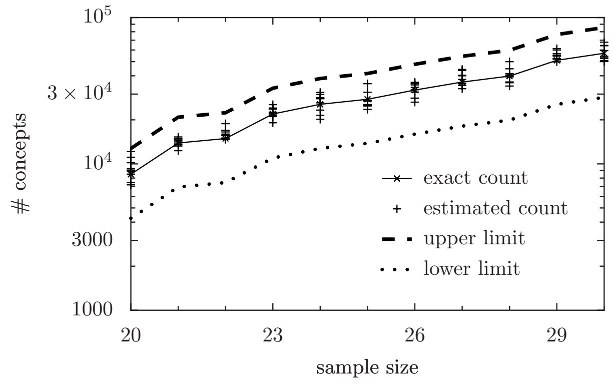


Figure 5.7.: Estimated number of concepts on samples of the 'chess' dataset.

of concepts ("exact count"), the upper and lower $1 \pm 1/2$ deviation limits ("upper limit" and "lower limit"), as well as the result of the randomized algorithm in a series of ten runs per sample ("estimated count"); all in logarithmic scale. One can observe that the approximated result lies within the desired deviation bounds in all randomized computations. This shows that the positive evaluation of the step heuristic for the chess dataset (see Fig. 5.6) transfers to all sub-contexts it induces.

Remark 5.1. A very similar approximate counting approach can be designed for Problem 5.3, i.e., for approximating the number of frequent sets $\mathcal{F}(\mathcal{D}, f) = \{F \subseteq E \colon q_{\text{freq}}(\mathcal{D}, F) \geq f\}$ of a dataset $\mathcal{D}$ over $E = \{e_1, \ldots, e_n\}$ with respect to some frequency threshold f. Again it is possible to express the counting result by a number of factors, each of which corresponding to size ratios of certain sub-families of $\mathcal{F}$. In contrast to the concept case where the sequence is defined via the data records, here it is defined via the items—otherwise the ratios would not have a reasonable lower bound.

Formally, for $i \in \{1, \ldots, n\}$ let $E_i = \{e_1, \ldots, e_i\}$ and $\mathcal{F}_i = \{F \in \mathcal{F} \colon F \subseteq E_i\}$ be the family of frequent sets containing only elements from the first i items. Just as we did with the number of closed sets in Equation 5.7, we can rewrite $|\mathcal{F}| = |\mathcal{F}_n|$ as the product

$$|\mathcal{F}_0| \prod_{i=s+1}^{n} \frac{|\mathcal{F}_i|}{|\mathcal{F}_{i-1}|} = |\mathcal{F}_n| \quad . \tag{5.9}$$

It follows directly from the definition that for all $i \leq n$ it holds that $\mathcal{F}_{i-1} \subseteq \mathcal{F}_i$. Moreover, a constant non-zero lower bound for the ratios $|\mathcal{F}_{i-1}| \,/\, |\mathcal{F}_i|$ is implied by the following observation: For distinct sets $F \neq F'$ with $F, F' \in \mathcal{F}_i \backslash \mathcal{F}_{i-1}$ the sets $F \backslash \{e_i\}$ and $F' \backslash \{e_i\}$ are distinct elements of $\mathcal{F}_{i-1}$. Hence, $\mathcal{F}_i \setminus \mathcal{F}_{i-1}$ can be mapped injectively into $\mathcal{F}_{i-1}$ implying $|\mathcal{F}_i \setminus \mathcal{F}_{i-1}| \leq |\mathcal{F}_{i-1}|$, and thus it holds that

$$1 \geq |\mathcal{F}_{i-1}| \,/\, |\mathcal{F}_i| \geq 1/2 \ . \tag{5.10}$$

Having access to an almost uniform sampling procedure for the frequent set families $\mathcal{F}_i$, one can again estimate the individual factors of the product in Equation 5.9. While the Markov chain from Example 5.1 has no efficient worst-case mixing time, it mixes well in practice using a heuristic simulation step number similar to the one presented in Section 5.3.4. Figure 5.8 presents the performance of the resulting counting algorithm for the dataset "chess" in terms of both accuracy and computation time. $\diamondsuit$

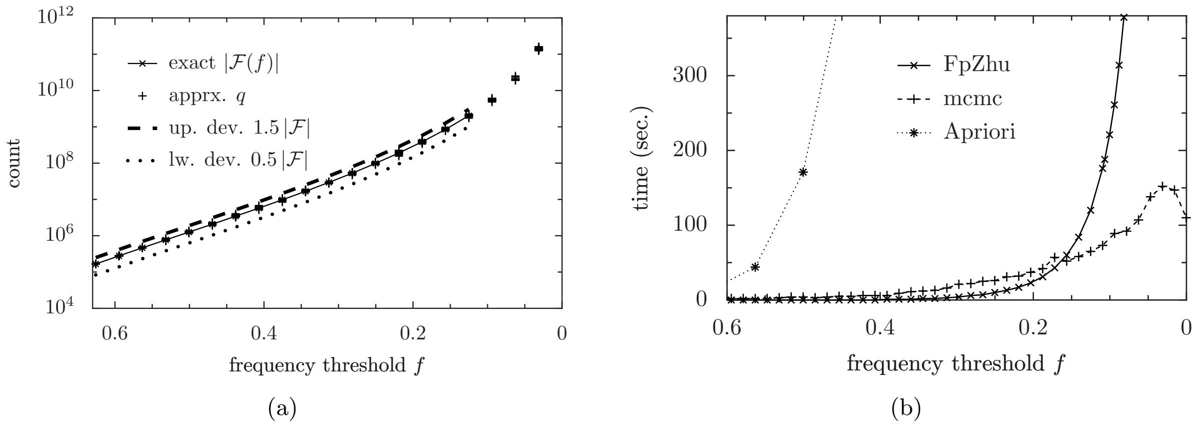


Figure 5.8.: Results of performance experiments for Markov chain Monte Carlo frequent set counting (as described in Rem. 5.1) with respect to accuracy (a) and computation time (b). Accuracy was within bounds for 170 trials. For the remaining 30 trials using very low frequency thresholds no comparison could be made due to prohibitive computation time required for exact exhaustive baseline methods: in contrast to randomized counting, the computation times of exhaustive methods grow exponentially for decreasing frequency thresholds.

5.5.3. Complexity

As shown in Proposition 5.4, the Markov chain Monte Carlo approach used in Algorithm 10 for sampling closed sets requires super-polynomially many steps for some input datasets. Although the heuristic polynomial step number from Section 5.3.4 leads to sufficient experimental results, clearly one would prefer a polynomial sampling algorithm with provable worst-case guarantee. There is, however, some evidence that no such algorithm exists or that at least designing one is a difficult problem.

As we have shown in the previous section, an algorithm that can sample concepts almost uniformly in polynomial time can be used to design a fully polynomial randomized approximation scheme (FPRAS) for counting the number of concepts for a given context. While Kuznetsov [2001] proves that exact concept counting is **#P**-hard, its analysis does not rule out the existence of a corresponding FPRAS, because the proof relies on a non AP reduction[3] (see Sec. 5.5.1). Nevertheless the following considerations give some insight in the difficulty of approximate concept counting. Counting concepts is (parsimoniously) equivalent to counting the number of maximal bipartite cliques of a given bipartite graph, which is as hard as the complexity class $\#\mathbf{RH\Pi_1}$ (hardness result and complexity class are both introduced by Dyer et al. [2004]). Importantly, equivalence as well as hardness are here defined with respect to approximation-preserving reductions.

The class $\#\mathbf{RH\Pi_1}$ is a subclass of $\#\mathbf{P}$, which is characterized logically as the class of counting problems that can be expressed by a logical first-order formula in a certain restricted Horn form extended with only universal quantification (hence, the name $\#\mathbf{RH\Pi_1}$). The problems known to be complete for $\#\mathbf{RH\Pi_1}$ lie between the $\#\mathbf{P}$-complete problems and those for which an FPRAS is known (completeness again with respect to AP-reductions). Despite much interest for algorithms that approximate an $\#\mathbf{RH\Pi_1}$-hard problem, there is no known FPRAS for any of them. Thus, albeit only hard for a subclass of $\#\mathbf{P}$, there is some evidence that there is no FPRAS for counting the closed sets of a given input database and, hence, no worst-case efficient algorithm for sampling closed sets almost uniformly.

All of these considerations do not take into account frequency constraints. However, one motivation for considering approximate pattern counting, stated in the beginning of this section, is that it leads to efficient feasibility tests for exhaustive pattern enumeration. Since exhaustive closed set enumeration is usually combined with a frequency constraint, it is interesting to investigate the complexity of this combination. Clearly, the problem of approximately counting closed frequent sets is at least as hard as approximate closed set counting. In addition, it is also at least as hard as counting frequent sets, i.e., as solving Problem 5.3 (see Rem. 5.1 for a corresponding worst-case inefficient Markov chain Monte Carlo RAS): if a given dataset $\mathcal{D}$ over E is transformed into a dataset $\mathcal{D}'$ consisting of $|E|$ copies of $\mathcal{D}$ plus once the data record $E \setminus \{e\}$ for each $e \in E$ then all sets that are f-frequent in $\mathcal{D}$ are $(f\,|E|)$-frequent in $\mathcal{D}$ and vice versa. In addition all sets $F \subseteq E$ are closed in $\mathcal{D}'$. So counting closed $(f\,|E|)$-frequent sets in $\mathcal{D}'$ means counting f-frequent sets in $\mathcal{D}$.

Below we provide evidence that, in fact, the introduction of a frequency constraint makes the problem of counting closed sets substantially harder. Interestingly, the result already holds for Problem 5.3, i.e., for considering only minimum frequency and neglecting closedness. We can make a much more profound statement regarding the hardness of approximating the number of frequent sets $|\mathcal{F}|$ than regarding the approximate counting of all closed sets. Namely, we can show that under reasonable complexity assumptions there is not even a bounded probability algorithm for approximating the logarithm of $|\mathcal{F}|$.

Gunopulos et al. [2003] proves $\#\mathbf{P}$-hardness of COUNT-FREQUENT-SETS by giving a reduction from the problem of counting the satisfying assignments of a monotone 2-CNF formula (see also the discussion in Section 5.6). However, this reduction is not approximation-preserving. Hence, it only implies that there is no *exact* algorithm for that problem (unless $\mathbf{P} = \mathbf{NP}$). For ruling out approximation algorithms we have to choose a different starting point, namely the hardness of approximating a frequent set of maximum cardinality. The following result holds

[3] Alternatively, considering only the second step of the proof of Kuznetsov [2001], it starts from the problem of counting the number of satisfying assignments of a DNF, which is believed not to be $\#\mathbf{P}$-hard with respect to AP reductions because it admits an FPRAS.

for that problem.

Lemma 5.7. *Unless for all $\epsilon > 0$ and for all problems in* ***NP*** *there is a BP-algorithm that runs in time 2^{n^ϵ} for instances of size n, i.e., unless $\boldsymbol{NP} = \bigcap_{\epsilon>0} \boldsymbol{BPTIME}(2^{n^\epsilon})$, the following holds: There is a constant $\delta_{BC} > 0$ such that there is no polynomial time BP-algorithm that, given a dataset $\mathcal{D}$ of size n and a frequency threshold f, computes a frequent set $F \in \mathcal{F}(\mathcal{D}, f)$ with $|F| \geq (\max_{F' \in \mathcal{F}} |F'|)/n^{\delta_{BC}}$.*

A proof of this lemma can be found in Appendix A.3. It is based on Khot's seminal inapproximability result for the problem of approximating a maximum balanced bipartite clique, which in fact rules out BP-algorithms under the above assumption (see Khot [2004] also for information about the magnitude of δ_{BC}). As the number of frequent sets induces an upper bound as well as a lower bound on the maximum cardinality of a frequent set (see the proof below), Lemma 5.7 can be used to show the following result.

Theorem 5.8. *Unless $\boldsymbol{NP} = \bigcap_{\epsilon>0} \boldsymbol{BPTIME}(2^{n^\epsilon})$, the following holds: There is a constant $\delta_{\#F} > 0$ such that there is no polynomial time BP-algorithm that, given a dataset $\mathcal{D}$ of size n and a frequency threshold f, approximates $\log |\mathcal{F}(\mathcal{D}, f)|$ within $n^{\delta_{\#F}}$.*

Proof. It is easy to show that approximating only the maximum number k such that there is a frequent set of size k is polynomially equivalent to the actual construction of a corresponding set (iteratively delete an $e \in E$ and check whether e is part of the last remaining maximum frequent set by checking whether maximum size has dropped to $k-1$; repeat with $k' = k - 1$ until complete solution constructed). Combined with Lemma 5.7, for our claim it is sufficient to show that if for all $\delta > 0$ there is a BP-algorithm $\mathfrak{A}_\delta$ approximating $\log |\mathcal{F}|$ within n^δ then there is also for all $\delta' > 0$ a BP-algorithm $\mathfrak{A}'_{\delta'}$ approximating the maximum frequent set cardinality k within $n^{\delta'}$.

Let $\delta' > 0$. Choose δ with $0 < \delta < \delta'$ and observe that there is an $n^* \in \mathbb{N}$ such that $n^\delta \log n < n^{\delta'}$ for all $n \geq n^*$. We construct a polynomial time BP-algorithm $\mathfrak{A}'_{\delta'}$ for approximating the maximum frequent set cardinality based on a potential polynomial time BP-algorithm $\mathfrak{A}_\delta$ for approximating the logarithm of the number of frequent sets as follows. For all (finitely many) instances of size less than n^* the algorithm $\mathfrak{A}'_{\delta'}$ looks up the true result in a hard-coded table. Let $\mathcal{D}$ be an instance with $\mathrm{size}(\mathcal{D}) = n \geq n^*$. Then $\mathfrak{A}'_{\delta'}$ checks whether the maximum frequent set size k is less or equal to 1 (which can easily be done in polynomial time). If this is not the case then $\mathfrak{A}'_{\delta'}$ returns $\mathfrak{A}_\delta(\mathcal{D}, f)$, i.e., the result of the assumed polynomial time n^δ-approximation of the number of frequent sets. We claim that $\mathfrak{A}'_{\delta'}$ is a polynomial time BP-algorithm approximating the logarithm of the maximum cardinality of a frequent set within $n^{\delta'}$ as required.

The time complexity as well as correctness for the cases $n < n^*$ and $k \leq 1$ follow from the definitions. Let $\mathcal{D}$ be a dataset over E of size $n \geq n^*$ such that $|F| = k > 1$ for a maximum frequent set $F \in \mathcal{F}$. By the anti-monotonicity of $\mathcal{F}$, the number $|\mathcal{F}|$ induces an upper bound on k. Namely, as all of the 2^k subsets of F must also be an element of $\mathcal{F}$, it holds that $|\mathcal{F}| \geq 2^k$ and, hence, $\log |\mathcal{F}| \geq k$. On the other hand, since all frequent set are of size at most k, it is possible to map $\mathcal{F}$ injectively into the set of all k-element strings E^k over symbols E, e.g., by

$$\{e_{i_1}, \ldots, e_{i_l}\} \mapsto \begin{cases} e_{i_1} e_{i_2} \ldots e_{i_l} \underbrace{e_{i_l} \ldots e_{i_l}}_{k-l}, & \text{if } l \geq 1 \\ e_n e_{n-1} \ldots e_1, & \text{otherwise} \end{cases}$$

which is injective for $k > 1$. Thus, it holds that $|\mathcal{F}| \leq \left|E^k\right| = |E|^k = n^k$ implying $\log|\mathcal{F}| \leq k \log n$. By the approximation guarantee of $\mathfrak{A}_\delta$, it follows that

$$\begin{aligned}
& & \log|\mathcal{F}| \,/\, \log n \leq k & & \leq \log|\mathcal{F}| \\
&\Leftrightarrow & \mathfrak{A}_\delta(\mathcal{D}, f)/(n^\delta \log n) \leq k & & \leq n^\delta \mathfrak{A}_\delta(\mathcal{D}, f) \\
&\Leftrightarrow & \mathfrak{A}'_{\delta'}(\mathcal{D}, f)/n^{\delta'} \leq k & & \leq n^{\delta'} \mathfrak{A}'_{\delta'}(\mathcal{D}, f)
\end{aligned}$$

as required for the correctness of $\mathfrak{A}'_{\delta'}$. □

Although the complexity assumption of this theorem is stronger than $\mathbf{P} \neq \mathbf{NP}$ it is still a widely believed standard assumption. Moreover, non-existence of an α-approximation of the logarithm of a number implies non-existence of an 2^α-approximation to the actual number. Thus, we have strong evidence that there is no reasonable worst-case correct approximation algorithm for the general COUNT-FREQUENT-SETS problem and in particular no fully polynomial approximation scheme.

5.6. Summary and Discussion

This chapter introduced an alternative paradigm to local pattern discovery called controlled repeated pattern sampling, which involves the generation of single patterns according to a user-specified probability distribution. In order to solve this sampling problem, we presented a Metropolis-Hastings algorithm with an underlying Markov chain that operates directly on the closed sets. In two exemplary applications we demonstrated how this algorithm can be used for knowledge discovery. Also we highlighted the connection between closed set sampling and counting and used it to derive negative complexity results that justify why our algorithm exhibits no provable worst-case accuracy guarantee.

It is important to note that although this sampling algorithm is described with respect to closed sets of binary datasets, the approach is not limited to this scenario. In fact it can be applied to all pattern classes having a similar Galois connection between the pattern language $\mathcal{L}$ (partially ordered by a relation $\preceq$) and the family of data records $\mathcal{P}(\mathcal{D})$. More precisely, it can be applied wherever one is interested in sampling from the set of fixpoints of composed mappings $\mathrm{msd}(\mathrm{supp}(\cdot))$ respectively $\mathrm{supp}(\mathrm{msd}(\cdot))$ defined by

$$\begin{aligned}
\mathrm{supp}\colon \mathcal{L} &\to \mathcal{P}(\mathcal{D}) \\
\mathrm{msd}\colon \mathcal{P}(\mathcal{D}) &\to \mathcal{L}
\end{aligned}$$

that form an order-reversing Galois connection, i.e.,

$$L \preceq \mathrm{msd}(\mathcal{D}') \iff \mathcal{D}' \subseteq \mathrm{supp}(L)$$

for all $L \in \mathcal{L}$ and $\mathcal{D}' \subseteq \mathcal{D}$. Furthermore, the method can be combined with any anti-monotone or monotone constraint. It is for instance easy to see that the algorithm remains correct when the chain is restricted to the set of all frequent concepts.

5.6.1. Further Remarks on Problem Complexities

As mentioned in Section 5.5.3, Gunopulos et al. [2003] shows #**P**-hardness of counting the number of frequent sets by a proof relying on a non-AP-reduction. Although this approach does not lead to an inapproximability result for frequent set counting, it can be used to derive other interesting observations.

The reduction starts from the problem of computing the number of satisfying truth assignments of a given monotone 2-CNF formula, i.e., a conjunctive normal form formula containing only two positive literals per clause. It was shown by Zuckerman [1996] that this number and in fact even its logarithm is hard to approximate within a factor of n^{ϵ} for instances of size n. Gunopulos et al.'s construction transforms a 2-CNF formula into a binary dataset over a ground set of size n such that the number of satisfying truth assignments corresponds to the number of sets that are *not* 1-frequent (then it uses the fact that the number of infrequent sets is equal to 2^n minus the number of frequent sets). Hence, the construction is non-parsimonious, i.e., the results of original and transformed problem instances are not equal—in fact their difference is not polynomially bounded. As a consequence, relative approximation guarantees are not preserved.

Still, the two aforementioned results together do imply the interesting fact that there is no efficient approximation algorithm for counting the number of infrequent sets even if the (absolute) frequency threshold is fixed to 1. This is an interesting difference to the same restriction for COUNT-FREQUENT-SETS: Restricted to support threshold 1 approximating the number of frequent sets becomes equivalent to approximating the number of satisfying assignments of a given DNF-formula. For this problem there is a well-known fully polynomial randomized approximation scheme [Karp et al., 1989].

5.6.2. Is Pattern Sampling a Theoretical Dead-End?

Our investigations on the computational complexity of closed and frequent set sampling might suggest that this approach is a dead-end at least with respect to theoretical guarantees. Of course, one can always try to identify well-behaved subclasses of the general problem that provide rapidly mixing Markov chains even in the worst case. In fact, there is also a much broader potential research direction that appears to be a logical next step after investigating Markov chain Monte Carlo sampling: in the absence of feasible a priori bounds on the mixing time, there are several techniques that can be used to draw *perfect samples*, i.e., generated exactly according to the stationary distribution (see Huber [2008] and references therein).

With this idea, one replaces Monte Carlo algorithms, i.e., randomly varying result quality but deterministic computation time by Las Vegas algorithms, i.e., guaranteed result quality but randomly varying computation time. This is an important shift because of the increased amount of control the latter approach provides: Markov chain Monte Carlo methods with step heuristic may be likely to be accurate on a given real-world dataset, but one never knows for certain. Markov chain Las Vegas algorithms have the disadvantage that they may be slow on a given database, but, in contrast to the drawback of heuristic Markov chain Monte Carlo algorithms, this event is completely observable.

The most popular variant of these perfect sampling techniques is coupling from the past. It can be applied efficiently if the chain is monotone in the following sense: the state space is partially ordered, contains a global maximal as well as a global minimal element, and if two simulations of the chain that use the same source of random bits are in state x and y with $x \leq y$ then also the successor state of x must be smaller than the successor state of y. Indeed, at first glance it appears that coupling from the past may be applied to our chain because its state space is partially ordered by $\preceq$ and always contains a global minimal element $\perp$ as well as a global maximal element $\top$. Even for the uniform target distribution, however, the chain is not monotone. This can be observed in the example of Figure 5.4: denote by $\mathrm{succ}_{d,5,0}(\langle I, E\rangle)$ the successor of $\langle I, E\rangle$ when the random bits used by the computation induce the decisions $d =$ down, $o = 5$, and $p = 0$. Then $\langle bc, 1\rangle \preceq \langle A, \emptyset\rangle$ whereas $\mathrm{succ}_{d,5,0}(\langle bc, 1\rangle) = \langle bc, 1\rangle \not\preceq \langle d, 45\rangle = \mathrm{succ}_{d,5,0}(\langle A, \emptyset\rangle)$. It is an open problem, how to define a Markov chain on the concept lattice that has monotone transition probabilities.

6. Conclusion

This chapter gives a final high-level summary of the thesis and discusses the character and the value of the main results (for a complete list of all formal results see Section 1.2 of the introduction). After this discussion, a concluding section presents directions for possible follow-up research.

6.1. Summary

The goal of this thesis is to provide the algorithmic foundations of efficient local pattern discovery algorithms for listing small and interesting collections of closed sets that can be applied whenever there are too many closed (frequent) sets to be listed effectively. This is motivated by the observation that state-of-the-art methods can exhibit infeasible computation times already for small to mid-sized input datasets, and that, moreover, the reason for this phenomenon is a potentially overwhelming and uncontrolled output size (outweighing the excellent input size scalability of state-of-the-art methods).

In particular, as specified in the introductory chapter, we are interested in algorithms that provide control over the produced output and exhibit an efficient time complexity (see requirements (R1) and (R2) of Section 1.1). Regarding the efficiency requirement, Chapter 2 gave a concise formal introduction to local pattern discovery that included output sensitive efficiency notions and explained why traditional closed set discovery can be performed adhering to this standard. Subsequent chapters then introduced different modified closed set discovery approaches—closed constrained set discovery, strongly closed set discovery, and controlled repeated closed set sampling—resulting in output families that, albeit incorporating advanced reduction techniques, retain efficient enumerability. This is a non-trivial property, as several output reduction techniques proposed in the literature do not allow an efficient enumeration of their induced result family (unless $\mathbf{P} = \mathbf{NP}$). Examples are maximal frequent sets (following readily from results of Boros et al. [2003]) or top-k interesting sets with respect to different measures (see Section 1.1).

In particular, it turned out that the closed constrained sets are induced by a partially defined closure operator if the constraints induce a confluent set system and, moreover, that divide-and-conquer closed set listing can be used to enumerate the results with polynomial delay if the set system is also strongly accessible. In contrast, listing closed constrained sets in general cannot be done even with output polynomial time, and listing fixpoints of a partially defined closure operator is intractable if one allows accessible set systems. Moreover, the divide-and-conquer approach is not limited to closed sets, i.e., fixpoints of a support closure operator; it is applicable to general closure operators (having a strongly accessible domain). Hence, by constructively showing that the strongly closed sets are always induced by an efficiently computable closure operator, it is possible to also design a polynomial delay algorithm for listing that pattern class. While closed constrained sets as well as strongly closed sets can be listed by the same algorithmic approach (and can even be combined), for controlled repeated closed set sampling we utilized a completely different technique, namely the Metropolis-Hastings algorithm applied to the state space formed by the complete closed set lattice. Again, by focusing on datasets inducing a state space with a high conductance, we end up with a polynomial delay algorithm (with respect to the total family of sampled sets). Similar to closed constrained set listing, the problem appears to be intractable for general instances due to the intrinsic connection between sampling closed sets and counting them.

6.2. Discussion

In order to assess the relevance and importance of the results summarized above, it is necessary to recognize the theoretical character of this work: it motivates and introduces closed set discovery variants and then analyzes them rigorously with respect to the formal output sensitive efficiency notions. That is, for all discussed algorithms, tight asymptotic performance bounds are provided in terms of their time, space, and delay. In addition, the thesis maintains a strict separation of algorithms and computational problems—acknowledging that they stand in a many-to-many relationship. While this course of action follows the tradition of algorithmic research on the original closed set listing problem of Boros et al. [2003] and Uno et al. [2004], it stands in contrast to the majority of the remaining literature on pattern discovery algorithms, which mainly relies on empirical results. Clearly, this preference of a theoretical over a practical approach has advantages as well as disadvantages.

The dual theoretical perspective (algorithm/problem) enables technology transfers of algorithmic approaches from one problem to another one that shares some relevant structural property. In particular, this applies to the divide-and-conquer set listing approach, which is generalized in this thesis from standard closed set listing to fixpoint listing of arbitrary, even partially defined, closure operators. Moreover, the dual perspective motivates the consideration of problem complexities in its own right: for several computational problems the thesis provides precise lower bounds on their computational complexity respectively discusses other evidence of their intractability. Examples are the exponential lower bound for listing the fixpoints of closure operators defined on an accessible set systems or the hardness of counting, and thus uniformly sampling, frequent sets. Such results should not be interpreted as a show stopper for certain pattern discovery applications. Instead they are an important guidance for future research efforts: they motivate a more careful requirement analysis for pattern discovery as well as the revelation of additional structural assertions of the input datasets. Both directions can lead to more specialized but tractable subproblems with corresponding efficient algorithms.

While a theoretical analysis driven by a focus on the asymptotic worst-case performance can lead to problem insights in principle, it can also neglect aspects of practical importance: proving formal correctness and performance results requires all problems and algorithms to be modeled and presented in the most simple form possible. Consequently, this thesis mostly ignores implementation techniques and data structures that do not affect the asymptotic worst-case performance, such as for instance Apriori's candidate generation mechanism [Agrawal et al., 1996] or prefix trees for input data compression [Han et al., 2004]. Such techniques are, however, indispensable for real-world applications with large scale input data. Hence, the highly streamlined and idealized presentation can be considered incomplete from a practical point of view.

In summary, while this thesis does not present readily engineered data analysis systems, it analyzes their algorithmic cores, gives precise performance guarantees, and explicitly lists the formal assumptions under which they hold. In addition, the thesis provides novel insights into the computational complexity of several pattern discovery problems that constitute guidelines for further algorithmic research.

6.3. Outlook

Having discussed the contributions of this thesis, we now turn to potential research topics that might emerge from it. Note that several open research questions that specifically relate to certain parts of this thesis are listed at the end of the different chapters. In addition, it should be mentioned that the computational complexity of many of the interestingness measures used in local pattern discovery is still open. However, here we conclude with two major directions of potential future research that go beyond specific algorithmic issues.

Taking up the discussion of Section 6.2, a natural follow-up question of this thesis is how one can improve the input size scalability of our closed set discovery algorithms. While basic techniques like successive database reduction can apparently be integrated easily in both listing and sampling, more sophisticated methods are necessary in large scale applications where the complete database does not fit into main memory. An interesting approach for such scenarios that, if applicable, leads to sublinear scaling is sequential input sampling [Scheffer and Wrobel, 2002]. Here, pattern interestingness is only evaluated on a subset of the available data records. Starting from a small sample, the sample size is iteratively increased until a family of approximately best patterns can be identified with high confidence. In order to control the error probability, this method relies on a priori knowing the total number of patterns to be evaluated. Generally, it is an interesting question how this can be handled in a constraint-based environment where we do not know the number of candidate patterns in advance. An important further question, specifically for the case of closed patterns, is what does happen to the probability of a pattern being closed if only a data subset is considered? As we have seen in Section 4.1.2, this probability increases with the strength of closedness, but precise statements would be desirable in order to (probabilistically) characterize the result family. Note that we face similar problems in the setting of a streaming environment (see, e.g., Chi et al. [2006]), which is another model aimed to attack large-scale input data.

Finally, let us consider another major branch of potential follow-up research that leaves the purely algorithmic area and can be regarded as a further development of the controlled repeated pattern sampling paradigm. It emerges from challenging two of our implicit ground assumptions. Namely, we assumed that we know (a) an interestingness measure and (b) a precise interestingness threshold that together define factual patterns in a given input database. In Chapter 5 we already abandoned assumption (b). In fact, however, even (a) is often violated because interestingness can be subjective (see Jaroszewicz et al. [2009]), i.e., depend on hidden user-specific parameters. Formally, the interestingness measure is not only a function of the pattern (and the database) but also depends on a parameter vector θ were the optimal setting θ^* is user-specific and, hence, initially unknown to the system. In order to find these parameters one could design an *interactive pattern discovery process* that proceeds as follows: produce a pattern, ask the user for one click of feedback about this pattern (out of a small set of possible feedbacks F), produce a new pattern with an adjusted notion of interestingness, and so on. Thus, such a process essentially operates in an active learning environment. Formally, a discovery process P maps feedback histories to parameter settings of the quality function, i.e., $P(h) = \theta$ where a feedback history $h \in (\mathcal{L} \times F)^*$ is a sequence of patterns and corresponding user feedbacks. After t steps of interaction between the user and the discovery process we have a feedback history h_t that is defined by $h_0 = \epsilon$ and $h_{i+1} = h_i \circ (x_{i+1}, f_{i+1})$ where $x_{i+1} \in \mathcal{L}$ is a pattern generated according to $q(P(h_i), \cdot)$ and $f_{i+1} \in F$ the feedback the user gave after inspecting pattern x_{i+1}. While the controlled repeated pattern sampling approach can naturally be used to produce the required individual patterns, it is an interesting open question how to design a discovery process P such that the resulting sequence of interestingness measures $q(P(h_i), \cdot)$ for $i = 1, 2, \ldots$ is likely to "rapidly approach" the optimal quality measure $q(\theta^*, \cdot)$.

A. Additional Proofs

A.1. Analysis of Breadth-First Minimum Generator Listing

This section presents a formal proof of correctness of the breadth-first mininum generator listing algorithm. The proof is based on the fact that the breadth-first enqueuing/dequeueing process is compatible with the modifed lexicographical order $\prec_M$ and, moreover, on the guarantee of Lemma 2.11 that every non-empty minimum generator has a certain predecessor with respect to that order.

Proof of Theorem 2.12. Exactness and non-redundancy follow from showing that, in modified lexicographical order, for each $[F] \in \mathcal{S}^{\equiv}$ a tuple $(\mu(F), \sigma(F), A)$ is enqueued and only tuples of this form are enqueued, i.e., that the following three properties hold:

(i) If a tuple (G, C, A) is enqueued before (G', C', A') then $G \prec_M G'$.

(ii) For all $[F] \in \mathcal{S}^{\equiv}$ a tuple (G, C, A) with $G = \mu(F)$ and $C = \sigma(F)$ is enqueued, and $A \supseteq \{e \in E \colon (G \cup \{e\}) \in \mathcal{S}^{\equiv}, e < \min G\}$.

(iii) All enqueued tuples (G, C, A) are of the form $G = \mu(F)$ and $C = \sigma(F)$ for some $[F] \in \mathcal{S}^{\equiv}$.

Property (i) is implied by the breadth-first strategy and the following observation: if $G_1 \prec_M G_2$ then all sets G'_1 generated from G_1 are modified lexicographically smaller than all sets G'_2 generated from G_2.

Assume that (ii) is violated for some $[F]$. Then choose a class $[F]$ that violates (ii) with a minimal $G' = \mu(F)$. As $(\emptyset, \sigma(\emptyset), E)$ is enqueued in line 2, it holds that $G' \neq \emptyset$. By Lemma 2.11, $G = G' \setminus \{\min G'\}$ is lexicographically minimal in $[G]$. The anti-monotonicity of the enumeration space and $G \subset G'$ imply that (ii) holds for $[G]$. In particular a tuple (G, C, A) is enqueued with $(\min G') \in A$ because $\min G' < \min G$ (for the same reason and because of the anti-monotonicity of $\mathcal{S}^{\equiv}$, the augmentation set A' satisfies $A' \supseteq \{e \in E : (G' \cup \{e\}) \in \mathcal{S}^{\equiv}, e < \min G'\}$). Thus, G' is generated subsequently in line 8. Then $\sigma(G')$ does not pass the visited check in line 10. This implies that $[G']$ has already been visited, say via $G'' \in [G']$. It follows from (i) that $G'' \prec_M G'$ contradicting $G' = \mu(G')$.

For (iii) observe that $C = \sigma(G)$ for all enqueued tuples by the generation of C in line 2. Now assume that $G \neq \mu(C)$ for an enqueued tuple (G, C, A). Then there is an $G' \in [C]$ with $G' \prec_M G$. By the anti-monotinicity of the enumeration space and (ii) a tuple (G', C', A') is enqueued, and by (i) it is enqueued before (G, C, A). In the same iteration $C' = C$ is added to $\mathcal{V}$. Consequently, (G, C, A) can not be enqueued as it does not pass the visited check in line 10—a contradiction.

Finally, regarding the performance note that for each dequeued tuple at most $|E|$ augmentations are evaluated involving a computation of σ and a visited check. The prefix-tree lookup is performed in time $|C| \leq |E|$ and σ is computed in time size$(\mathcal{D})$. Also the space dominant data structure $\mathcal{V}$ contains at most one element of size at most $|E|$ for each dequeued tuple. Hence, the performance claims follow from the correctness, i.e., from the fact that there is exactly one enqueued tuple per potentially interesting equivalence class. □

A.2. Analysis of the Zero-Elimination Algorithm

This section gives a proof of the correctness and the time complexity of Algorithm 8 as stated in Theorem 4.7. The correctness part is splitted into to Lemmas showing soundness respectively completeness, i.e., that all elements added to the input set are part of the Δ-closure and that all elements of the Δ-closure are eventually added. These results are then combined with an analysis of the time complexity in the proof of Theorem 4.7 below.

Lemma A.1. *Let a dataset $\mathcal{D}$ over E, an integer $\Delta \geq 0$, and a set $F \subseteq E$ be an input to Algorithm 8. After the execution of line 4 the following invariants hold for all $e \in E \setminus F$ throughout the remaining computation:*

$$\begin{aligned}\Sigma(e) \geq &\left|\bar{\mathcal{D}}[\{e\}] \cap \mathcal{D}[\sigma_\Delta(F)]\right| + \\ &\left|\{D \in \bar{\mathcal{D}}[\{e\}] \colon D \in \bar{\mathcal{D}}[\sigma_\Delta(F)] \wedge D \notin X\}\right| \enspace .\end{aligned}$$

Moreover, for all $e \in E$ that Algorithm 8 adds to C it holds that $e \in \sigma_\Delta(F)$.

Proof. After line 4 the values $\Sigma(e)$ are initialized to $|\{D \in \mathcal{D}[F] \colon e \notin D\}|$. Thus the invariants hold at this point and no e has been added to C that could violate $e \in \sigma_\Delta(F)$. Subsequently Σ and C are only modified during deleteColumn and deleteRow. We prove correct behavior of these procedures by induction on their calling order. Consider a call deleteRow(D). By induction before the execution of line 1 the invariant holds. Hence, after the addition of D to X for all $e \notin D$, $\Sigma(e) - 1$ is greater or equal to

$$\begin{aligned}&\left|\bar{\mathcal{D}}[\{e\}] \cap \mathcal{D}[\sigma_\Delta(F)]\right| + \\ &\left|\{D \in \bar{\mathcal{D}}[\{e\}] \colon D \in \bar{\mathcal{D}}[\sigma_\Delta(F)] \wedge D \notin X\}\right| \enspace .\end{aligned}$$

Since in line 3 the number $\Sigma(e)$ is decremented by only one for such an e, this step does not violate any invariant.

Furthermore, for a call deleteColumn(e) we know that $\Sigma(e) \leq \Delta$ and thus by induction that $\left|\bar{\mathcal{D}}[\{e\}] \cap \mathcal{D}[\sigma_\Delta(F)]\right|$ is not greater than Δ. It follows that $e \in \sigma_\Delta(F)$. □

Lemma A.2. *Let a dataset $\mathcal{D}$ over E, an integer $\Delta \geq 0$, and a set $F \subseteq E$ be an input to Algorithm 8. Moreover, suppose $e \in (\sigma_\Delta(F) \setminus F)$. Then Algorithm 8 adds e to C eventually.*

Proof. Since $e \in \sigma_\Delta(F)$ there is a k such that $e \in \hat{\sigma}^k_\Delta(F) \setminus \hat{\sigma}^{k-1}_\Delta(F)$. We prove the claim by induction on k. In case $k = 1$, deleteColumn(e) is called in Line 6 of the main procedure and subsequently added to C. Otherwise by induction there is a state of the algorithm in which $C \supseteq \hat{\sigma}^{k-1}_\Delta(F)$ and so there is a subsequent state in which deleteRow(D) has been called for all $D \in \mathcal{D}\left[\hat{\sigma}^{k-1}_\Delta(F)\right]$. Since

$$\left|\mathcal{D}\left[\hat{\sigma}^{k-1}_\Delta(F)\right]\right| - \left|\mathcal{D}\left[\hat{\sigma}^{k-1}_\Delta(F) \cup \{e\}\right]\right| < \Delta$$

the counter $\Sigma(e)$ will have been set to a value smaller than Δ at this state and consequently e will be added to C. $\square$

Proof of Theorem 4.7. We show that Algorithm 8 can be used to compute σ_Δ with the claimed complexity. For the running time we treat each of the three procedures separately. The first loop in line 2 can be realized by one pass through the data in time $O\big(\overline{\text{size}}(\mathcal{D}[F])\big)$. The same holds for the second for-loop constituting a total time of $O\big(\overline{\text{size}}(\mathcal{D}[F])\big)$ spend in the main procedure.

For the deleteColumn procedures observe that it is called at most once for each $e \in E \setminus F$ and contains a loop over all data records $D \in \mathcal{D}[F]$ with $e \notin D$ realized by one traversal of the list L_e in time $O(|L_e|)$. So the total time spend within this procedure is bounded by

$$O\left(\sum_{e \in E \setminus F} |L_e|\right) = O\big(\overline{\text{size}}(\mathcal{D}[F])\big) \quad .$$

Similarly, the deleteRow procedure is called at most once for each element $D \in \mathcal{D}[F]$ and contains a loop over all $e \in \bar{D}$. Thus the time spend in this procedure and thus the overall time complexity of the whole algorithm is bounded by $O\big(\overline{\text{size}}(\mathcal{D}[F])\big)$.

For the correctness, Lemma A.1 implies that always $C \subseteq \sigma_\Delta(F)$. Together with Lemma A.2 it follows that C as returned in line 7 of the main procedure is equal to $\sigma_\Delta(F)$. $\square$

A.3. Hardness of Approximating a Maximum Size Frequent Set

In this section, we give the proof of Lemma 5.7, which is based on the following seminal result of Khot [2004] for approximating a maximum balanced bipartite clique of a given bipartite graph: Unless there are probabilistic algorithms with an arbitrary small exponential time complexity for all problems in **NP** there is no polynomial approximation scheme for maximum balanced biclique problem, i.e., the infimum of all constants k such that there is a k-approximation algorithm for that problem is bounded away from 1. It was known before the publication of Khot's paper that such a result, once achieved, can be boosted via derandomized graph products (introduced by Alon et al. [1995]). So that the result of Khot implies in fact:

Theorem A.3 (Khot [2004]). *Unless $\boldsymbol{NP} = \bigcap_{\epsilon>0} \boldsymbol{BPTIME}(2^{n^\epsilon})$, the following holds: there is a constant $\delta_{BC} > 0$ such that there is no algorithm approximating a maximum balanced biclique within a factor of* $\text{size}(x)^{\delta_{BC}}$ *for instances x.*

With this we can prove the hardness of approximating a frequent set of maximum size by giving an approximation-preserving reduction from the maximum balanced biclique problem.

Proof of Lemma 5.7. Suppose there is an algorithm $\mathfrak{A}$ approximating a maximum frequent set within a factor of $\alpha(\text{size}(x))$ for instances x. Then one can construct a dataset $\mathcal{D}$ over V from a given bipartite graph $G = (V, U, E)$ by setting $\mathcal{D} = \{\{v \in V \colon \{v, u\} \in E\} \colon u \in U\}$ and find t_{APX} the maximum $t \in \{1, \dots, |\mathcal{D}|\}$ such that $|\mathfrak{A}(\mathcal{D}, t)| \geq t$ by running $\mathfrak{A}$ at most $|\mathcal{D}|$ times. Let (X, Y) be a maximum balanced bipartite clique in G with size t_{OPT}. Any set of data records corresponding to a subset $Y' \subseteq Y$ contains the t_{OPT} items corresponding to X—in particular those with $|Y'| = t_{\mathrm{OPT}}/\alpha(\mathrm{size}(D)) = t^*$. This implies for the maximum cardinality of a t^*-frequent set in $\mathcal{D}$, denoted as $\mathrm{mfs}(\mathcal{D}, t^*)$,

$$\mathrm{mfs}(\mathcal{D}, t^*) \geq t_{\mathrm{OPT}} \Rightarrow |\mathfrak{A}(D, t^*)| \geq t_{\mathrm{OPT}}/\alpha(\mathrm{size}(\mathcal{D})) = t^* \ .$$

But then $t_{\mathrm{APX}} \geq t^* = t_{\mathrm{OPT}}/\alpha(\mathrm{size}(\mathcal{D})) \geq t_{\mathrm{OPT}}/\alpha(\mathrm{size}(G))$, because the transformed instance is of equal or smaller size. Since all necessary computations can be performed in polynomial time, we have a polynomial algorithm approximating a maximum balanced biclique within a factor of $\alpha(\mathrm{size}(x))$ for instances x, which completes our approximation-preserving reduction. □

Bibliography

F. N. Afrati, A. Gionis, and H. Mannila. Approximating a collection of frequent sets. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2004)*, pages 12–19. ACM, 2004.

R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI/MIT Press, 1996.

N. Alon, U. Feige, A. Wigderson, and D. Zuckerman. Derandomized graph products. *Computational Complexity*, 5(1):60–75, 1995.

H. Arimura and T. Uno. Polynomial-delay and polynomial-space algorithms for mining closed sequences, graphs, and pictures in accessible set systems. In *Proceedings of the SIAM International Conference on Data Mining (SDM 2009)*, pages 1087–1098. SIAM, 2009.

S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

A. Asuncion and D. Newman. UCI machine learning repository, 2007. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Y. Bastide, N. Pasquier, R. Taouil, G. Stumme, and L. Lakhal. Mining minimal non-redundant association rules using frequent closed itemsets. In *Proceedings of the 1st International Conference on Computational Logic (CL 2000)*, volume 1861 of *Lecture Notes in Computer Science*, pages 972–986. Springer, 2000a.

Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal. Mining frequent patterns with counting inference. *SIGKDD Explorations Newsletter*, 2(2):66–75, 2000b.

S. D. Bay and M. J. Pazzani. Detecting group differences: Mining contrast sets. *Data Mining and Knowledge Discovery*, 5(3):213–246, 2001.

J. Besson, C. Robardet, J.-F. Boulicaut, and S. Rome. Constraint-based concept mining and its application to microarray data analysis. *Intelligent Data Analysis*, 9(1):59–82, 2005.

S. Bistarelli and F. Bonchi. Soft constraint based pattern mining. *Data and Knowledge Engineering*, 62(1):118–137, 2007.

M. Boley. On approximating minimum infrequent and maximum frequent sets. In *Proceedings of the 10th International Discovery Science Conference (DS 2007)*, volume 4755 of *Lecture Notes in Computer Science*, pages 68–77. Springer, 2007a.

M. Boley. Intelligent pattern mining via quick parameter evaluation. In *NSF Symposium on Next Generation of Data Mining and Cyber-Enabled Discovery for Innovation (NGDM '07)*, 2007b.

M. Boley and T. Gärtner. On the complexity of constraint-based theory extraction. In *Proceedings of the 12th International Discovery Science Conference (DS 2009)*, volume 5808 of *Lecture Notes in Computer Science*, pages 92–106. Springer, 2009.

M. Boley and H. Grosskreutz. A randomized approach for approximating the number of frequent sets. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008)*, pages 43–52. IEEE Computer Society, 2008.

M. Boley and H. Grosskreutz. Approximating the number of frequent sets in dense data. *Knowledge and Information Systems*, 21(1):65–89, Oct. 2009a.

M. Boley and H. Grosskreutz. Non-redundant subgroup discovery using a closure system. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2009), Part I*, volume 5781 of *Lecture Notes in Computer Science*, pages 179–194. Springer, 2009b.

M. Boley, T. Horváth, A. Poigné, and S. Wrobel. Efficient closed pattern mining in strongly accessible set systems (extended abstract). In *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2007)*, volume 4702 of *Lecture Notes in Computer Science*, pages 382–389. Springer, 2007a.

M. Boley, T. Horváth, A. Poigné, and S. Wrobel. Efficient closed pattern mining in strongly accessible set systems. In *Proceedings of the 5th International Workshop on Mining and Learning with Graphs (MLG 2007)*, 2007b.

M. Boley, T. Horváth, A. Poigne, and S. Wrobel. New results on listing closed sets of strongly accessible set systems. In *Proceedings of the 7th International Workshop on Mining and Learning with Graphs (MLG 2009)*, 2009a.

M. Boley, T. Horváth, and S. Wrobel. Efficient discovery of interesting patterns based on strong closedness. In *Proceedings of the SIAM International Conference on Data Mining (SDM 2009)*, pages 1002–1013. SIAM, 2009b.

M. Boley, T. Horváth, and S. Wrobel. Efficient discovery of interesting patterns based on strong closedness. *Statistical Analysis and Data Mining*, 2(5–6):346–360, 2009c.

M. Boley, H. Grosskreutz, and T. Gärtner. Formal concept sampling for counting and threshold-free local pattern mining. In *Proceedings of the SIAM International Conference on Data Mining (SDM 2010)*. SIAM, 2010a.

M. Boley, T. Horváth, A. Poigné, and S. Wrobel. Listing closed sets of strongly accessible set systems with applications to data mining. *Theoretical Computer Science*, 411(3):691–700, 2010b.

F. Bonchi and C. Lucchese. On closed constrained frequent pattern mining. In *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM 2004)*, pages 35–42. IEEE Computer Society, 2004.

F. Bonchi and C. Lucchese. Extending the state-of-the-art of constraint-based pattern discovery. *Data and Knowledge Engineering*, 60(2):377–399, 2007. Web data and schema management, Papers selected from the XSDM 2005 workshop.

E. Boros, V. Gurvich, L. Khachiyan, and K. Makino. On maximal frequent and minimal infrequent sets in binary matrices. *Annals of Mathematics and Artificial Intelligence*, 39(3): 211–221, 2003.

J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Free-sets: A condensed representation of boolean data for the approximation of frequency queries. *Data Mining and Knowledge Discovery*, 7 (1):5–22, 2003.

J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI 1998)*. Morgan Kaufmann, 1998.

D. Bridge and A. Ferguson. Diverse product recommendations using an expressive language for case retrieval. *Advances in Case-Based Reasoning*, pages 291–298, 2002.

T. Brijs, G. Swinnen, K. Vanhoof, and G. Wets. Using association rules for product assortment decisions: A case study. In *Knowledge Discovery and Data Mining*, pages 254–260, 1999.

C. Bucila, J. Gehrke, D. Kifer, and W. White. DualMiner: A dual-pruning algorithm for itemsets with constraints. *Data Mining and Knowledge Discovery*, 7(3):241–272, 2003.

T. Calders, C. Rigotti, and J. françois Boulicaut. A survey on condensed representations for frequent sets. In *Revised Selected Papers of the European Workshop on Inductive Databases and Constraint Based Mining*, volume 3848 of *Lecture Notes in Computer Science*, pages 64–80. Springer, 2005.

L. Cerf, J. Besson, C. Robardet, and J.-F. Boulicaut. Closed patterns meet n-ary relations. *ACM Transactions on Knowledge Discovery from Data*, 3(1):1–36, 2009.

J. Cheng, Y. Ke, and W. Ng. δ-tolerance closed frequent itemsets. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006)*, pages 139–148. IEEE Computer Society, 2006.

Y. Chi, H. Wang, P. S. Yu, and R. R. Muntz. Catch the moment: maintaining closed frequent itemsets over a data stream sliding window. *Knowledge and Information Systems*, 10(3): 265–294, 2006.

L. De Raedt and A. Zimmermann. Constraint-based pattern set mining. In *Proceedings of the 7th SIAM International Conference on Data Mining (SDM 2007)*, 2007.

G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '99)*, pages 43–52. ACM, 1999.

M. Dyer, L. A. Goldberg, C. Greenhill, and M. R. Jerrum. The relative complexity of approximate counting problems. *Algorithmica*, 38(3):471–500, 2004.

T. Fawcett and F. J. Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1(3):291–316, 1997.

U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

T. A. Feo and M. G. C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2):67 – 71, 1989.

C. Fox, A. Levitin, and T. Redman. The notion of data and its quality dimensions. *Information Processing & Management*, 30(1):9–19, 1994.

W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus. Knowledge discovery in databases: an overview. *AI Magazine*, 13(3):57–70, 1992.

B. Ganter and K. Reuter. Finding all closed sets : A general approach. *Order*, 8:283–290, 1991.

B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer Verlag, 1999.

F. Geerts, B. Goethals, and T. Mielikäinen. Tiling databases. In *Proceedings of the 7th International Discovery Science Conference*, volume 3245 of *Lecture Notes in Computer Science*, pages 278–289. Springer, 2004.

A. Gély. A generic algorithm for generating closed sets of a binary relation. In *Proceedings of 3rd International Conference on Formal Concept Analysis (ICFCA 2005)*, volume 3403 of *Lecture Notes in Computer Science*, pages 223–234. Springer, 2005.

K. Geurts, G. Wets, T. Brijs, and K. Vanhoof. Profiling high frequency accident locations using association rules. In *Proceedings of the 82nd Annual Transportation Research Board*, page 18, 2003.

B. Goethals and M. J. Zaki. Advances in frequent itemset mining implementations: report on fimi'03. *SIGKDD Explorations*, 6(1):109–117, 2004.

L. A. Goldberg. *Efficient algorithms for listing combinatorial structures*. Cambridge University Press, 1993.

H. Grosskreutz, S. Rüping, and S. Wrobel. Tight optimistic estimates for fast subgroup discovery. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD 2008), Part I*, volume 5211 of *Lecture Notes in Computer Science*, pages 440–456. Springer, 2008.

D. Gunopulos, R. Khardon, H. Mannila, S. Saluja, H. Toivonen, and R. S. Sharma. Discovering all most specific sentences. *ACM Transactions on Database Systems*, 28(2):140–174, 2003.

J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8(1):53–87, 2004.

D. J. Hand. Pattern detection and discovery. In *Proceedings of the ESF Exploratory Workshop on Pattern Detection and Discovery*, volume 2447 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2002.

D. J. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining.* MIT Press, 2001.

M. A. Hasan and M. J. Zaki. MUSK: Uniform sampling of k maximal patterns. In *Proceedings of the SIAM International Conference on Data Mining (SDM 2009)*, pages 650–661. SIAM, 2009.

M. A. Hasan, V. Chaoji, S. Salem, J. Besson, and M. J. Zaki. ORIGAMI: Mining representative orthogonal graph patterns. In *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007)*, pages 153–162. IEEE Computer Society, 2007.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* Springer, 2009.

W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

M. Huber. Perfect simulation with exponential tails on the running time. *Random Structures and Algorithms*, 33(1):29–43, 2008.

S. Jaroszewicz, T. Scheffer, and D. A. Simovici. Scalable pattern mining with bayesian networks as background knowledge. *Data Mining and Knowledge Discovery*, 18(1):56–100, 2009.

C. Jermaine. Finding the most interesting correlations in a database: how hard can it be? *Information Systems*, 30(1):21–46, 2005.

M. R. Jerrum and A. Sinclair. The markov chain monte carlo method: an approach to approximate counting and integration. In *Approximation algorithms for NP-hard problems*, pages 482–520. PWS Publishing, 1997.

M. R. Jerrum, L. G. Valiant, and V. V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43(2-3):169–188, 1986.

M. R. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM*, 51(4):671–697, 2004.

D. S. Johnson. Approximation algorithms for combinatorial problems. In *Proceedings of the 5th Annual ACM Symposium on Theory of Computing (STOC '73)*, pages 38–49. ACM, 1973.

D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123, 1988.

R. M. Karp, M. Luby, and N. Madras. Monte-carlo approximation algorithms for enumeration problems. *Journal of Algorithms*, 10(3):429–448, 1989.

M. Kaytoue-Uberall, S. Duplessis, S. O. Kuznetsov, and A. Napoli. Two FCA-based methods for mining gene expression data. In *Proceedings of the 7th International Conference on Formal Concept Analysis (ICFCA 2009)*, volume 5548 of *Lecture Notes in Computer Science*, pages 251–266. Springer, 2009.

S. Khot. Ruling out PTAS for graph min-bisection, densest subgraph and bipartite clique. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2004)*, pages 136–145. IEEE Computer Society, 2004.

A. J. Knobbe and E. K. Y. Ho. Pattern teams. In *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2006)*, volume 4213 of *Lecture Notes in Computer Science*, pages 577–584, 2006.

A. J. Knobbe, B. Crémilleux, J. Fürnkranz, and M. Scholz. From local patterns to global models: the lego approach to data mining. In *From Local Patterns to Global Models: Proceedings of the ECML/PKDD 2008 Workshop (LEGO '08)*, 2008.

B. Korte and L. Lovász. Relations between subclasses of greedoids. *Mathematical Methods of Operations Research*, 29(7):249–267, 1985.

R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the web for emerging cyber-communities. *Computer Networks*, 31(11-16):1481–1493, 1999.

S. Kuznetsov. On computing the size of a lattice and related decision problems. *Order*, 18: 313–321, 2001.

N. Lavrac, B. Kavsek, P. Flach, and L. Todorovski. Subgroup discovery with CN2-SD. *Journal of Machine Learning Research*, 5(Feb):153–188, 2004.

E. L. Lawler. Matroid intersection algorithms. *Mathematical Programming*, 9:31–59, 1975.

E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Generating all maximal independent sets: NP-hardness and polynomial-time algorithms. *SIAM Journal of Computing*, 9(3): 558–565, 1980.

D. A. Levin, Y. Peres, and E. L. Wilmer. *Markov Chains and Mixing Times.* American Mathematical Society, 2008.

H. Mannila. Local and global methods in data mining: Basic techniques and open problems. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming (ICALP 2002)*, volume 2380 of *Lecture Notes in Computer Science*, pages 57–68. Springer, 2002.

B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Effective personalization based on association rule discovery from web usage data. In *Proceedings of the 3rd International Workshop on Web Information and Data Management (WIDM '01)*, pages 9–15. ACM, 2001.

S.-I. Nakano and T. Uno. Constant time generation of trees with specified diameter. In *Revised Papers of 30th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2004)*, volume 3353 of *Lecture Notes in Computer Science*, pages 33–45, 2004.

M. Nanni, B. Kuijpers, C. Körner, M. May, and D. Pedreschi. Spatiotemporal data mining. In *Mobility, Data Mining and Privacy - Geographic Knowledge Discovery*. Springer, 2008.

J. G. Oxley. *Matroid Theory*. Oxford University Press, 2006.

N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1):25–46, 1999.

N. Pasquier, R. Taouil, Y. Bastide, G. Stumme, and L. Lakhal. Generating a condensed representation for association rules. *Journal of Intelligent Information Systems*, 24(1):29–60, 2005.

J. Pei and J. Han. Can we push more constraints into frequent pattern mining? In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2000)*, pages 350–354. ACM, 2000.

D. Randall. Rapidly mixing markov chains with applications in computer science and physics. *Computing in Science and Engineering*, 8(2):30–41, 2006.

G. J. Russell and A. Petersen. Analysis of cross category dependence in market basket selection. *Journal of Retailing*, 76(3):367–392, 2000.

J. Sawada. Generating bracelets in constant amortized time. *SIAM Journal of Computing*, 31 (1):259–268, 2001.

T. Scheffer and S. Wrobel. Finding the most interesting patterns in a database quickly by using sequential sampling. *Journal of Machine Learning Research*, 3:833–862, 2002.

L. Schietgat, F. Costa, J. Ramon, and L. D. Raedt. Maximum common subgraph mining: A fast and effective approach towards feature generation. In *Proceedings of the 7th International Workshop on Mining and Learning with Graphs (MLG 2009)*, 2009.

J. Simon. On the difference between one and many (preliminary version). In *Proceedings of 4th International Colloquium on Automata, Languages and Programming (ICALP 1977)*, volume 52 of *Lecture Notes in Computer Science*, pages 480–491. Springer, 1977.

A. Sinclair and M. R. Jerrum. Approximate counting, uniform generation and rapidly mixing markov chains. In *Proceedings of the International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 314 of *Lecture Notes in Computer Science*, pages 134–148, 1987.

P. Slavík. A tight analysis of the greedy algorithm for set cover. *Journal of Algorithms*, 25(2): 237–254, 1997.

R. Srikant, Q. Vu, and R. Agrawal. Mining association rules with item constraints. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining (KDD-97)*, pages 67–73. AAAI Press, 1997.

P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the right interestingness measure for association patterns. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '02)*, pages 32–41. ACM, 2002.

N. Tatti. Maximum entropy based significance of itemsets. *Knowledge and Information Systems*, 17(1):57–77, 2008.

J. W. Tukey. We need both exploratory and confirmatory. *The American Statistician*, 34(1): 23–25, 1980.

T. Uno, T. Asai, Y. Uchida, and H. Arimura. An efficient algorithm for enumerating closed patterns in transaction databases. In *Proceedings of the 7th International Discovery Science Conference (DS 2004)*, volume 3245 of *Lecture Notes in Computer Science*, pages 16–31. Springer, 2004.

L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8 (2):189–201, 1979.

J. Wang, J. Han, Y. Lu, and P. Tzvetkov. TFP: An efficient algorithm for mining top-k frequent closed itemsets. *IEEE Transactions on Knowledge and Data Engineering*, 17(5): 652–664, 2005a.

L. Wang, H. Zhao, G. Dong, and J. Li. On the complexity of finding emerging patterns. *Theoretical Computer Science*, 335(1):15–27, 2005b. Pattern Discovery in the Post Genome.

S. Wrobel. An algorithm for multi-relational discovery of subgroups. In *Proceedings of the 1st European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD '97)*, volume 1263 of *Lecture Notes in Computer Science*, pages 78–87. Springer, 1997.

D. Xin, J. Han, X. Yan, and H. Cheng. Mining compressed frequent-pattern sets. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB '05)*, pages 709–720. VLDB Endowment, 2005.

C. Yang, U. Fayyad, and P. S. Bradley. Efficient discovery of error-tolerant frequent itemsets in high dimensions. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2001)*, pages 194–203. ACM, 2001.

F. Zhu, X. Yan, J. Han, P. S. Yu, and H. Cheng. Mining colossal frequent patterns by core pattern fusion. In *Proceedings of the 23rd International Conference on Data Engineering (ICDE 2007)*, pages 706–715, 2007.

D. Zuckerman. On unapproximable versions of NP-complete problems. *SIAM Journal of Computing*, 25(6):1293–1304, 1996.

Curriculum Vitae

Mario Boley

Born 1981 in Leverkusen, Germany; childhood in Kerpen (Rheinl.); currently living in Bonn

Contact via e-mail `mario.boley@iais.fraunhofer.de` or via telephone +49-2241-14-2789

Education and Experience

since 2010 *Full research position* at the Knowledge Discovery department of Fraunhofer IAIS and the Computer Science Institute III of the University of Bonn; involved in projects of German Research Foundation (constructive machine learning, well-behaved pattern mining) as well as the European Commission project LIFT

2011 *Research visit* at University of Antwerp hosted by Prof. Dr. Bart Goethals

2007 – 2011 *PhD-Student* at Fraunhofer IAIS and University of Bonn; Writing this dissertation under Prof. Dr. Stefan Wrobel; teaching and student thesis supervision

2006 *Research visit* at University of Illinois in Chicago hosted by Prof. Dr. Bhaskar DasGupta funded by the German National Academic Foundation

2003 - 2007 *Student employee* at Fraunhofer IAIS in the departments Spatial Decision Support and Knowledge Discovery; research, software development, GIS

2002 - 2007 *CS studies* at University of Bonn (minor Mathematics); Diploma with distinction

2001 - 2002 *Military service*; army media division in Mayen

1988 - 2001 *School* in Kerpen

Awards

2008 *Best-Student-Paper-Award* of the IEEE International Conference on Data Mining (ICDM) for: "A Randomized Approach For Approximating the Number of Frequent Sets" with H. Grosskreutz

2007 *Distinguished-Paper-Award* of the International Workshop on Mining and Learning with Graphs (MLG) for: "Efficient Closed Pattern Mining in Strongly Accessible Set Systems" with T. Horváth, A. Poigné und S. Wrobel

2003-2007 *Scholarship* of the German National Academic Foundation