Evaluating Pattern Set Mining Strategies in a Constraint Programming Framework

Tias Guns, Siegfried Nijssen, and Luc De Raedt

Katholieke Universiteit Leuven
Celestijnenlaan 200A, B-3001 Leuven, Belgium
{Tias.Guns,Siegfried.Nijssen,Luc.DeRaedt}@cs.kuleuven.be

Abstract. The pattern mining community has shifted its attention from local pattern mining to pattern set mining. The task of pattern set mining is concerned with finding a set of patterns that satisfies a set of constraints and often also scores best w.r.t. an optimisation criteria. Furthermore, while in local pattern mining the constraints are imposed at the level of individual patterns, in pattern set mining they are also concerned with the overall set of patterns. A wide variety of different pattern set mining techniques is available in literature. The key contribution of this paper is that it studies, compares and evaluates such search strategies for pattern set mining. The investigation employs concept-learning as a benchmark for pattern set mining and employs a constraint programming framework in which key components of pattern set mining are formulated and implemented. The study leads to novel insights into the strong and weak points of different pattern set mining strategies.

1 Introduction

In the pattern mining literature, the attention has shifted from local to global pattern mining [1,10] or from individual patterns to pattern sets [5]. *Local* pattern mining is traditionally formulated as the problem of computing $\mathrm{Th}(\mathcal{L}, \varphi, \mathcal{D}) = \{\pi \in L \mid \varphi(\pi, \mathcal{D}) \text{ is true}\}$, where $\mathcal{D}$ is a data set, L a language of patterns, and φ a constraint or predicate that has to be satisfied. Local pattern mining does not take into account the relationships between patterns; the constraints are evaluated *locally*, that is, on every pattern individually, and if the constraints are not restrictive enough, too many patterns are found. On the other hand, in *global* pattern mining or *pattern set mining*, one is interested in finding a small set of relevant and non-redundant patterns. Pattern set mining can be formulated as the problem of computing $\mathbf{Th}(\mathcal{L}, \varphi, \psi, \mathcal{D}) = \{\Pi \subseteq \mathrm{Th}(\mathcal{L}, \varphi, \mathcal{D}) \mid \psi(\Pi, \mathcal{D}) \text{ is true}\}$, where ψ expresses constraints that have to be satisfied by the overall pattern sets. In many cases a function f is used to evaluate pattern sets and one is then only interested in finding the best pattern set Π, i.e. $\arg\max_{\Pi \in \mathbf{Th}(\mathcal{L}, \varphi, \psi, \mathcal{D})} f(\Pi)$.

Within the data mining and the machine learning literature numerous approaches exist that perform pattern set mining. These approaches employ a wide variety of search strategies. In data mining, the *step-wise* strategy is common,

in which first all frequent patterns are computed; they are heuristically post-processed to find a single compressed pattern set; examples are KRIMP [16] and CBA [12]. In machine learning, the *sequential covering* strategy is popular, which repeatedly and heuristically searches for a good pattern or rule and immediately adds this pattern to the current pattern- (or rule-)set; examples are FOIL [14] and CN2 [3]. Only a small number of techniques, such as [5,7,9], search for pattern sets exhaustively, either in a step-wise or in a sequential covering setting.

The key contribution of this paper is that we study, evaluate and compare these common search strategies for pattern set mining. As it is infeasible to perform a detailed comparison on all pattern set mining tasks that have been considered in the literature, we shall focus on one prototypical task for pattern set mining: boolean concept-learning. In this task, the aim is to most accurately describe a concept for which positive and negative examples are given.Within this paper we choose to fix the optimisation measure used to *accuracy*; our focus is on the exploration of a wide variety of search strategies for this measure, from greedy to complete and from step-wise to one-step approaches.

To be able to obtain a fair and detailed comparison we choose to reformulate the different strategies within the common framework of constraint programming. This choice is motivated by [4,13], who have shown that constraint programming is a very flexible and usable approach for tackling a wide variety of local pattern mining tasks (such as closed frequent itemset mining and discriminative or correlated itemset mining), and recent work [9,7] that has lifted these techniques to finding k-pattern sets under constraints (sets containing exactly k patterns). In [7], a global optimization approach to mining pattern sets has been developed and has been shown to work for concept-learning, rule-learning, redescription mining, conceptual clustering as well as tiling. In the present work, we employ this constraint programming framework to compare different search strategies for pattern set mining, focusing on one mining task in more detail.

This paper is organized as follows: in Section 2, we introduce the problem of pattern set mining and its benchmark, concept-learning; in Section 3, we formulate these problems in the framework of constraint programming and introduce various search strategies for pattern set mining; in Section 4, we report on experiments, and finally, in Section 5, we conclude.

2 Pattern Set Mining Task

The benchmark task on which we shall evaluate different pattern set mining strategies is that of finding boolean concepts in the form of k-term DNF expressions. This task is well-known in computational learning theory [8] and is closely related to rule-learning systems such as FOIL [14] and CN2 [3] and data mining systems such as CBA [12] and KRIMP [16]. It is – as we shall now show – a pattern set mining task of the form $\arg\max_{\Pi \in \mathbf{Th}(\mathcal{L},\varphi,\psi,\mathcal{D})} f(\Pi)$.

In this setting, one is given a set of positive and negative examples, where each example corresponds to a boolean variable assignment to the items in $\mathcal{I}$, the set of possible items. Thus each example is an itemset $I_x \subseteq \mathcal{I}$. Positive examples will belong to the set of transactions $\mathcal{T}^+$, negatives ones to $\mathcal{T}^-$. The

pattern language is the set $\mathcal{L} = 2^{\mathcal{I}}$. Hence each pattern corresponds to an itemset $I_p \subseteq \mathcal{I}$ and represents a conjunction of items. The task is then to learn a concept description (a boolean formula) that covers all (or most) of the positive examples and none (or only a few) of the negatives. This can be measured using the accuracy measure, defined as:

$$\text{accuracy}(p, n) = \frac{p + (N - n)}{P + N} \tag{1}$$

where p and n are the number of positive, respectively negative, examples covered, and P and N are the total number of positive, resp. negative, examples present in the database. Concept descriptions are pattern sets, where each pattern set corresponds to a disjunction of patterns (conjunctions). Following [7,15], we shall focus on finding pattern sets that contain exactly k patterns. Thus the pattern sets correspond to k-term DNF formulas. An example is considered covered by the pattern set if the example is a superset of at least one of the itemsets in the pattern set.

Thus the task considered is an instance of the pattern set mining task $\arg\max_{\Pi \in \mathbf{Th}(\mathcal{L},\varphi,\psi,\mathcal{D})} f(\Pi)$, where f is the accuracy, $\mathcal{D} = \mathcal{T} = \mathcal{T}^+ \cup \mathcal{T}^-$, and $\mathcal{L} = 2^{\mathcal{I}}$; φ can be instantiated to a minimum support constraint (requiring that each pattern covers a certain number of examples), a minimum accuracy constraint (requiring that each pattern is individually accurate), or to *true*, a constraint which is always true and allows any pattern to be used which leads to an accurate final set. ψ states that $|\Pi| = k$.

Finding a good pattern set is often a hard task; many pattern set mining tasks, such as the task of k-term DNF learning, are NP complete [8]. Hence, there are no straightforward algorithms for solving such tasks in general, giving rise to a wide variety of search algorithms. The pattern set mining techniques they employ can be categorized along two dimensions.

Two-Step vs One-Step: in the two step approach, one first mines patterns under local constraints to compute the set $\text{Th}(\mathcal{L}, \varphi, \mathcal{D})$; afterwards, these patterns are fed into another algorithm that computes $\arg\max_{\Pi \in \mathbf{Th}(\mathcal{L},\varphi,\psi,\mathcal{D})} f(\Pi)$ using post-processing. In the one step approach, this strict distinction between these two phases can not be made.

Exact vs Approximate: exact methods provide strong guarantees for finding the optimal pattern set under the given constraints, while approximate methods employ heuristics to find good though not necessarily optimal solutions.

In the next section we will consider the instantiations of these settings for the case of concept learning. However, first we will introduce the constraint programming framework within which we will study these instantiations.

3 Constraint Programming Framework

Throughout the remainder of this paper we shall employ the constraint programming framework of [4] for representing and solving pattern set mining problems.

This framework has been shown 1) to allow for the use of a wide range of constraints, 2) to work for both frequent and discriminative pattern mining [13], and 3) to be extendible towards the formulation of k pattern set mining, cf. [7,9]. These other papers provide detailed descriptions of the underlying constraint programming algorithms and technology, including an analysis of the way in which they explore the search tree and a performance analysis. On the other hand, in the present paper – due to space restrictions – we need to focus on the declarative specification of the constraint programming problems; we refer to [4,13,7] for more details on the search strategy of such systems.

3.1 Constraint Programming Notation

Following [4], we assume that we are given a domain of items $\mathcal{I}$ and transactions $\mathcal{T}$, and a binary matrix $\mathcal{D}$. A key insight of the work of [4] is that constraint based mining tasks can be formulated as constraint satisfaction problems over the variables in $\pi = (I, T)$, where a pattern π is represented using the vectors I and T, with a boolean variable I_i and T_t for every item $i \in \mathcal{I}$ and every transaction $t \in \mathcal{T}$. A candidate solution to the constraint satisfaction problem is then one assignment of the variables in π which corresponds to a single itemset. For instance, the pattern represented by $\pi = (< 1, 0, 1 >, < 1, 1, 0, 0, 1 >)$ has items 1 and 3, and covers transactions 1, 2 and 5. Following [7], a pattern set $\mathit{\Pi}$ of size k simply consists of k such patterns: $\mathit{\Pi} = \{\pi_1, \ldots, \pi_k\}, \forall p = 1, \ldots, k : \pi_p = (I^p, T^p)$. We now discuss the different two-step and one-step pattern set mining approaches.

3.2 Two-Step Pattern Set Mining

In two step pattern set mining approaches, one first searches for the set of local patterns $\mathrm{Th}(\mathcal{L}, \varphi, \mathcal{D})$ that satisfy a set of constraints, and then post-processes these to find the pattern sets in $\mathbf{Th}(\mathcal{L}, \varphi, \psi, \mathcal{D})$.

Step 1: Local Pattern Mining. Using the above notation one can formulate many local pattern mining problems, such as frequent and discriminative pattern mining. Indeed, consider the following constraints, introduced in [4,13]:

$$\forall t \in \mathcal{T} : T_t = 1 \leftrightarrow \sum_{i \in \mathcal{I}} I_i(1 - \mathcal{D}_{ti}) = 0. \quad (\textit{Coverage})$$

$$\forall i \in \mathcal{I} : I_i = 1 \leftrightarrow \sum_{t \in \mathcal{T}} T_t(1 - \mathcal{D}_{ti}) = 0. \quad (\textit{Closedness})$$

$$\forall i \in \mathcal{I} : I_i = 1 \rightarrow \sum_{t \in \mathcal{T}} T_t \mathcal{D}_{ti} \geq \theta. \quad (\textit{Min. frequency})$$

$$\forall i \in \mathcal{I} : I_i = 1 \rightarrow \mathrm{accuracy}(\sum_{t \in \mathcal{T}^+} T_t \mathcal{D}_{ti}, \sum_{t \in \mathcal{T}^-} T_t \mathcal{D}_{ti}) \geq \theta. \quad (\textit{Min. accuracy})$$

In these constraints, the *coverage constraint* links the items to the transactions: it states that the transaction set T must be identical to the set of all transactions that are covered by the itemset I. The *closedness constraint* removes redundancy

by ensuring that an itemset has no superset with the same frequency. It is a well-known property that every non-closed pattern has an equally frequent and accurate closed counterpart. The *minimum frequency constraint* ensures that itemset I covers at least θ transactions. It can more simply be formulated as $\sum_{t \in \mathcal{T}} T_t \geq \theta$. The above formulation is equivalent, but posted for each item separately (observe that $\sum_{t \in \mathcal{T}} T_t \mathcal{D}_{ti}$ counts the number of t in column i of binary matrix $\mathcal{D}$ for which $T_t = 1$). This so-called reified formulation results in more effective propagation; cf. [4]. Finally, to mine for all accurate patterns instead of all frequent patterns, the *minimum accuracy constraint* can be used, which ensures that itemsets have an accuracy of at least θ. The reified formulation again results in more effective propagation [13].

To emulate the two step approaches that are common in data mining [12,16,1], we shall employ two alternatives for the first step: 1) using **frequent closed patterns**, which are found with the *coverage*, *closedness* and *minimum frequency* constraints; 2) using **accurate closed patterns**, found with the *coverage*, *closedness* and *minimum accuracy* constraints. Both of these approaches preform the first step in an *exact* manner. They find the set of all local patterns adhering to the constraints.

Step 2: Post-processing the Local Patterns. Once the local patterns have been computed, the two step approach post-processes them in order to arrive at the pattern set. We describe the two main approaches for this.
Post-processing by Sequential Covering (Approximate). The most simple approach to the second step is to perform greedy sequential covering, in which one iteratively selects the best local pattern from $\mathrm{Th}(\mathcal{L}, \varphi, \mathcal{D})$ and removes all of the positive examples that it covers. This continues until the desired number of patterns k has been reached or all positive examples are already covered. This type of approach is most common in data mining systems. Whereas in the first step the set $\mathrm{Th}(\mathcal{L}, \varphi, \mathcal{D})$ is computed exactly in these methods, the second step is often an iterative loop in which patterns are selected greedily from this set.
Post-processing using Complete Search (Exact). Another possibility is to perform a new round of pattern mining as described in [5]. In this case, each previously found pattern in $\mathcal{P} = \mathrm{Th}(\mathcal{L}, \varphi, \mathcal{D})$ can be seen as an item r in a new database; each new item identifies a pattern. One is looking for the set of pattern identifiers $P \subseteq \mathcal{P}$ with the highest accuracy. In this case, the set is not a conjunction of items, but a disjunction of patterns, meaning that a transaction is covered if at least one of the patterns $r \in P$ covers it. This can be formulated in constraint programming after a transformation of the data matrix $\mathcal{D}$ into a matrix $\mathcal{M}$ where the rows correspond to the transactions in $\mathcal{T}$ and the columns to the patterns in $\mathcal{P}$. Moreover $\mathcal{M}_{tr}$ is 1 if and only if pattern r covers transaction t and 0 otherwise. The solution set is now represented using $\Pi = (P, T)$, where P is the vector representation of the pattern set, that is, $P_r = 1$ iff $r \in P$. The formulation of post-processing using complete search is now:

$$\forall t \in \mathcal{T} : T_t = 1 \leftrightarrow \sum_{r \in \mathcal{P}} P_r \mathcal{M}_{tr} \geq 1. \quad (\textit{Disj. Coverage})$$

$$\forall r \in \mathcal{P} : P_r = 1 \rightarrow \mathrm{accuracy}(\sum_{t \in \mathcal{T}^+} \mathcal{L}_{tr}, \sum_{t \in \mathcal{T}^-} \mathcal{L}_{tr}) \geq \theta \quad (\textit{Min. Accuracy})$$

$$\sum_{r \in \mathcal{P}} P_r = k \quad (Set\ Size)$$

To obtain a reified formulation of the accuracy constraint we here use $\mathcal{L}_{tr} = \max(T_t, \mathcal{M}_{tr}) = \mathcal{M}_{tr} + (1 - \mathcal{M}_{tr})T_t$. The column for pattern r in this matrix represents the transaction vector if the pattern r would be added to the set P.

The first constraint is the *disjunctive coverage constraint*. The second constraint is the *minimum accuracy constraint*, posted on each pattern separately and taking the disjunctive coverage into account. Lastly, the *set size constraint* limits the pattern set to size k.

This type of exact two-step approach is relatively new in data mining. Two notable works are [11,5]. In these publications, it was proposed to post-process a set of patterns by using a complete search over subsets of patterns. If an exact pattern mining algorithm is used to compute the initial set of pattern in the first step, this gives a method that is overall exact and offers strong guarantees on the quality of the solution found.

3.3 One-Step Pattern Set Mining

This type of strategy, which is common in machine learning, searches for the pattern set $\mathbf{Th}(\mathcal{L}, \varphi, \psi, \mathcal{D})$ directly, that is, the computation of $\mathbf{Th}(\mathcal{L}, \varphi, \psi, \mathcal{D})$ and $\mathrm{Th}(\mathcal{L}, \varphi, \mathcal{D})$ is integrated or interleaved. This can remove the need to have strong constraints with strict thresholds in φ. There are two approaches to this:

Iterative Sequential Covering (Approximate). In the iterative sequential covering approach that we investigate here, a beam search is employed (with beam width b) to heuristically find the best pattern set. At each step during the search a local pattern mining algorithm is used to find the top-b patterns (with the highest accuracy) and uses these to compute new candidate pattern sets on its beam, after which it prunes all but the best b pattern sets from its beam. This setting is similar to 2-step sequential covering, only that here, at each iteration, the most accurate pattern is mined for directly, instead of selecting it from a set of previously mined patterns. Mining for the most accurate pattern can be done in a constraint programming setting by doing branch-and-bound search over the accuracy threshold θ. In the experimental section, we shall consider different versions of the approach, corresponding to different sizes of the beam. When $b = 1$, one often talks about **greedy sequential covering**.

Examples of one-step greedy sequential covering methods are FOIL and CN2; however, they use greedy algorithms to identify the local patterns instead of a branch-and-bound pattern miner. In data mining, the use of branch-and-bound pattern mining algorithms was recently studied for identifying top-b patterns; see for instance [2].

Global Optimization (Exact). The last option is to specify the problem of finding a pattern set of size k as a global optimization problem. This is possible in a constraint programming framework, thanks to its generic handling of constraints, cf. [7]. The formulation, searching for k patterns $\pi_p = (I^p, T^p)$ directly,

is as follows:

$$\forall p \in \{1,\ldots,k\} : \forall t \in \mathcal{T} : T_t^p \leftrightarrow \sum_{i \in \mathcal{I}} I_i^p (1 - \mathcal{D}_{ti}) = 0, \ (Coverage) \tag{2}$$

$$\forall p \in \{1,\ldots,k\} : \forall i \in \mathcal{I} : I_i^p \leftrightarrow \sum_{t \in \mathcal{T}} T_t^p (1 - \mathcal{D}_{ti}) = 0, \ (Closed) \tag{3}$$

$$T^1 < T^2 < \ldots < T^k \ \ (Canonical) \tag{4}$$

$$\forall t \in \mathcal{T} : B_t = \left[\left(\sum_{p \in \{1..k\}} T_t^p \right) \geq 1 \right], \ (Disj.coverage) \tag{5}$$

$$\textbf{maximize} \ \ \text{accuracy}\left(\sum_{t \in \mathcal{T}^+} B_t, \sum_{t \in \mathcal{T}^-} B_t \right). \ (Accurate) \tag{6}$$

Each pattern has to cover the transactions (Eq. 2) and be closed (Eq. 3). The canonical form constraint in Eq. 4 enforces a fixed lexicographic ordering on the itemsets, thereby avoiding to find equivalent but differently ordered pattern sets. In Eq. 5, the variables B_t are auxiliary variables representing whether transaction t is covered by at least one pattern, corresponding to a disjunctive coverage.

The one-step global optimization approaches to pattern set mining are less common; the authors are only aware of [7,9]. One could argue that some iterative pattern mining strategies will find pattern sets that are optimal under certain conditions. For instance, Tree^2 [2] can find a pattern set with minimal error on supervised training data; however, it neither provides guarantees on the size of the final pattern set nor provides guarantees under additional constraints.

4 Experiments

We now compare the different approaches to boolean concept learning that we presented and answer the following two questions:

- Q1: Under what conditions do the different strategies perform well?
- Q2: What quality/runtime trade-offs do the strategies make?

To measure the quality of a pattern set, we evaluate its accuracy on the dataset. This is an appropriate means of evaluation, as in the boolean concept learning task we consider, the goal is to find a concise description of the training data, rather than a hypothesis that generalizes to an underlying distribution.

The experiments were performed using the Gecode-based system proposed by [4] and performed on PCs running Ubuntu 8.04 with Intel(R) Core(TM)2 Quad CPU Q9550 processors and 4GB of RAM. The datasets were taken from the website accompanying this system[1]. The datasets were derived from the UCI Machine Learning repository [6] by discretising numeric attributes into eight equal-frequency bins. To obtain reasonably balanced class sizes we used the majority class as the positive class . Experiments were run on many datasets, but we here present the findings on 6 diverse datasets whose basic properties are listed in the top 3 rows of Table 1.

[1] `http://dtai.cs.kuleuven.be/CP4IM/datasets/`

	Mushroom	Vote	Hepatitis	German-credit	Austr.-credit	Kr-vs-kp
Transactions	8124	435	137	1000	653	3196
Items	119	48	68	112	125	73
Class distr.	52%	61%	81%	70%	55%	52%
Total patterns	221524	227032	3788342	25M+	25M+	25M+
Pattern poor/rich	poor	poor	poor	rich	rich	rich
frequency ≥ 0.7	12	1	137	132	274	23992
frequency ≥ 0.5	44	13	3351	2031	8237	369415
frequency ≥ 0.3	293	627	93397	34883	257960	25M+
frequency ≥ 0.1	3287	35771	1827264	2080153	24208803	25M+
accuracy ≥ 0.7	197	193	361	2	11009	52573
accuracy ≥ 0.6	757	1509	3459	262	492337	2261427
accuracy ≥ 0.5	11673	9848	31581	6894	25M+	25M+
accuracy ≥ 0.4	221036	105579	221714	228975	25M+	25M+

Table 1: Data properties and number of patterns found for different constraints and thresholds. 25M+ denotes that more than 25 million patterns were found.

4.1 Two-Step Pattern Set Mining

The result of a two-step approach obviously depends on the quality of the patterns found in the first step. We start by investigating the feasibility of this first step, and then study the two-step methods as a whole.

Step 1: Local Pattern Mining. As indicated in Section 3.2, we employ two alternatives: using frequent closed patterns and using accurate closed patterns. Both methods rely on a threshold to influence the number of patterns found.

Table 1 lists the number of patterns found on a number of datasets, for the two alternatives and with different thresholds. Out of practical considerations we stopped the mining process when more than 25 million patterns were found. Using this cut-off, we can distinguish pattern poor data (data having less than 25 million patterns when mining unconstrained) and pattern rich data. In the case of pattern poor data, one can mine using very low or even no thresholds. In the case of pattern rich data, however, one has to use a more stringent threshold in order not be overwhelmed by patterns. Unfortunately, one has to mine with different thresholds to discover how pattern poor or rich an unseen dataset is.

Step 2: Post-processing the Local Patterns. We now investigate how the quality of the global pattern sets is influenced by the threshold used in the first step, and how this compares to pattern sets found by 1-step methods that do not have such thresholds.

Post-processing by Sequential Covering (Approximate). This two-step approach picks the best local pattern from the set of patterns computed in step one. As such, the quality of the pattern set depends on whether the right patterns are in the pre-computed pattern set. We use our generic framework to compare two-step sequential covering to the one-step approach.

For pattern poor data for which the set of all patterns can be calculated, such as the mushroom, vote and hepatitis dataset, using all patterns obviously results

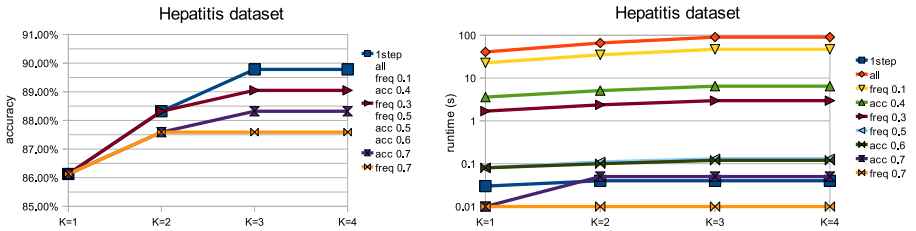

Fig. 1: Quality & runtime for approx. methods, pattern poor hepatitis dataset. In the left figure, algorithms with identical outcome are grouped together.

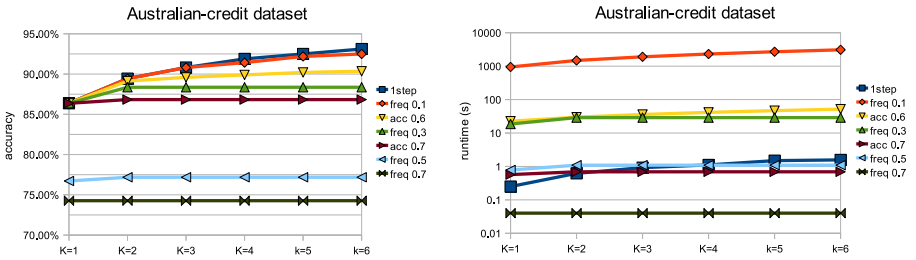

Fig. 2: Quality & runtime for approx. methods, pattern rich australian-credit dataset.

in the same pattern set as found by the one-step approach. Figure 1 shows the prototypical result for such data: low thresholds lead to good pattern sets, while higher thresholds gradually worsen the solution. For this dataset, starting from K=3, no better pattern set can be found. The same is true for the mushroom dataset, while in the vote dataset the sequential covering method continues to improve for higher K. Also note that in Figure 1 a better solution is found when using patterns with accuracy greater than 40%, compared to patterns with accuracy greater than 50%. This implies that a better pattern set can be found containing a local pattern that has a low accuracy on the whole data. This indicates that using accurate local patterns does not permit putting high thresholds in the first step. With respect to question Q2, we can observe that using a lower threshold comes at the cost of higher runtimes. However, for pattern poor datasets such as the one in Figure 1, these times are still manageable. The remarkable efficiency of the one-step sequential covering method is thanks to recent advances in mining top-k discriminative patterns [13].

On pattern rich data such as the german-credit, australian-credit and kr-vs-kp dataset, similar behaviour can be observed. The only difference is that one is forced to use more stringent thresholds. Because of this, the pattern set found by the one-step approach can usually not be found by the two-step approaches. Figure 2 exemplifies this for the australian-credit dataset. Using a frequency threshold of 0.1, the same pattern set as for the one-step method is found for up to K=3, but not so for higher K. When using the highest thresholds, there is a risk of finding significantly worse pattern sets. On the kr-vs-kp dataset, when using high frequency thresholds significantly worse results were found as well,

	Mushroom		Vote		Hepatitis		German-cr.		Austr.-cr.		Kr-vs-kp
	K	sec	K	sec	K	sec	K	sec	K	sec	K
all	–		–		–		-		-		-
freq. ≥ 0.7	6	0.2	only 1 pat		6	0.03	6	2.12	6	0.59	–
freq. ≥ 0.5	6	2.2	6	0.01	2	2650	2	8163	6	14244	–
freq. ≥ 0.3	6	14	6	0.89	–		–		–		-
freq. ≥ 0.1	2	9477	1	1015	–		–		–		-
acc. ≥ 0.7	6	8.6	6	0.12	6	3.05	6	0.01	1	713	–
acc. ≥ 0.6	*4	6714	5	14205	2	6696	6	104	–		–
acc. ≥ 0.5	–		1	391	1	3169	1	696	-		-
acc. ≥ 0.4	–		–		–		–		-		-

Table 2: Largest K (up to 6) and time to find it for the 2-step complete search method. - indicates that step 1 was aborted because more than 25 million patterns were found, – indicates that step 2 did not manage to finish within the timeout of 6 hours. * indicates that no other method found a better pattern set.

while this was not the case for the accuracy threshold. With respect to Q2 we have again observed that lower thresholds lead to higher runtimes for the two-step approaches. Lowering the thresholds further to find even better pattern sets would correspondingly come at the cost of even higher computation times.

Post-processing using Complete Search (Exact). When post-processing a collection of patterns using complete search, the size of that collection becomes a determining factor for the success of the method. Table 2 shows the same datasets and threshold values as in Table 1; here the entries show the largest K for which a pattern set could be found, up to K=6, and the time it took. A general trend is that in case many patterns are found in step 1, e.g. more than 100 000, the method is not able to find the optimal solution. With respect to Q1, only for the mushroom dataset the method found a better pattern set than any other method, when using all accurate patterns with threshold 0.4. For all other sets it found however, one of the 1-step methods found a better solution. Hence, although this method is exact in its second step, it depends on good patterns from its first step. Unfortunately finding those usually requires using low threshold values with corresponding disadvantages.

4.2 One-Step Pattern Set Mining

In this section we compare the different one-step approaches, who need no local pattern constraints and thresholds. We investigate how feasible the one-step exact approach is, as well as how close the greedy sequential covering method brings us to this optimal solution, and whether beam search can close the gap between the two.

When comparing the two-step sequential covering approach with the one-step approach, we already remarked that the latter is very efficient, though it might not find the optimal solution. The one-step exact method is guaranteed to find the optimal solution, but has a much higher computational cost. Table 3 below shows up to which K the exact method was able to find the optimal solution

Mushroom	Vote	Hepatitis	German-credit	Australian-credit	Kr-vs-kp
K=2	K=4	K=3	K=2	K=2	K=3

Table 3: Largest K for which the optimal solution was found within 6 hours.

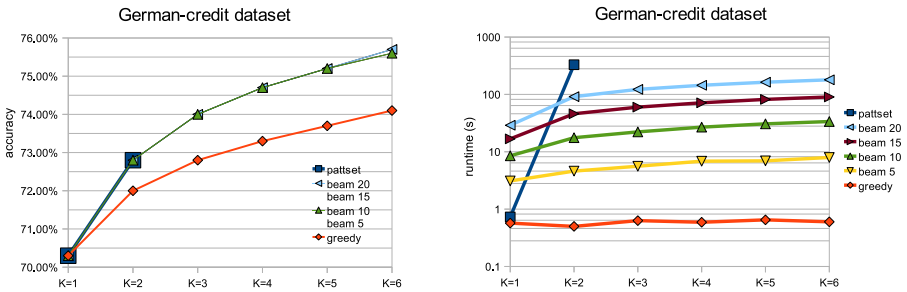

Fig. 3: Quality & runtime for 1-step methods, german-credit dataset. In the left figure, algorithms with identical outcome are grouped together.

within the 6 hours time out. Comparing these results to the two-step exact approach in Table 2, we see that pattern sets can be found without constraints, where the two-step approach failed even with constraints.

With respect to Q1 we observed that only for the kr-vs-kp dataset the greedy method, and hence all beam searches with a larger beam, found the same pattern sets as the exact method. For the mushroom and vote dataset, starting from beam width 5, the optimal pattern set was found. For the german-credit and australian-credit, a beam width of size 15 was necessary. The hepatitis dataset was the only dataset for which the complete method was able to find a better pattern set, in this case for K=3, within the timeout of 6 hours.

Figure 3 shows a representative figure, in this case for the german-credit dataset: while the greedy method is not capable of finding the optimal pattern set, larger beams successfully find the optimum. For K=6, beam sizes of 15 or 20 lead to a better pattern set than when using a lower beam size. The exact method stands out as being the most time consuming. For beam search methods, larger beams clearly lead to larger runtimes. The runtime only increases slightly for increasing sizes of K because the beam search is used in a sequential covering loop that shrinks the dataset at each iteration.

5 Conclusions

We compared several methods for finding pattern sets within a common constraint programming framework, where we focused on boolean concept learning as a benchmark. We distinguished one step from two step approaches, as well as exact from approximate ones. Each method has its strong and weak points, but the one step approximate approaches, which iteratively mine for patterns, provided the best trade-off between runtime and accuracy and do not depend on a threshold; additionally, they can easily be improved using a beam search. The exact approaches, perhaps unsurprisingly, do not scale well to larger and

pattern-rich datasets. A newly introduced approach for one-step exact pattern set mining however has optimality guarantees and performs better than previously used two-step exact approaches. In future work our study can be extended to consider other problem settings in pattern set mining, as well as other heuristics and evaluation metrics; furthermore, even though we cast all settings in one implementation framework in this paper, a more elaborate study could clarify how this approach compares to the pattern set mining systems in the literature.

Acknowledgements. This work was supported by a Postdoc and project "Principles of Patternset Mining" from the Research Foundation—Flanders, as well as a grant from the Agency for Innovation by Science and Technology in Flanders (IWT-Vlaanderen).

References

1. Bringmann, B., Nijssen, S., Tatti, N., Vreeken, J., Zimmermann, A.: Mining sets of patterns. In: Tutorial at ECMLPKDD 2010 (2010)
2. Bringmann, B., Zimmermann, A.: Tree^2 - decision trees for tree structured data. In: Jorge, A., Torgo, L., Brazdil, P., Camacho, R., Gama, J. (eds.) PKDD. LNCS, vol. 3721, pp. 46–58. Springer (2005)
3. Clark, P., Niblett, T.: The CN2 induction algorithm. Machine Learning 3, 261–283 (1989)
4. De Raedt, L., Guns, T., Nijssen, S.: Constraint programming for itemset mining. In: KDD. pp. 204–212. ACM (2008)
5. De Raedt, L., Zimmermann, A.: Constraint-based pattern set mining. In: SDM. SIAM (2007)
6. Frank, A., Asuncion, A.: UCI machine learning repository (2010), `http://archive.ics.uci.edu/ml`
7. Guns, T., Nijssen, S., De Raedt, L.: k-Pattern set mining under constraints. CW Reports CW596, Department of Computer Science, K.U.Leuven (Oct 2010), `https://lirias.kuleuven.be/handle/123456789/278655`
8. Kearns, M.J., Vazirani, U.V.: An introduction to computational learning theory. MIT Press, Cambridge, MA, USA (1994)
9. Khiari, M., Boizumault, P., Crémilleux, B.: Constraint programming for mining n-ary patterns. In: Proceedings of the 16th international conference on Principles and practice of constraint programming. pp. 552–567. CP'10, Springer (2010)
10. Knobbe, A., Crémilleux, B., J. Fürnkranz, M.S.: From local patterns to global models: The lego approach to data mining. In: Fürnkranz, J., Knobbe, A. (eds.) Proceedings of LeGo 2008, an ECMLPKDD 2008 Workshop (2008)
11. Knobbe, A.J., Ho, E.K.Y.: Pattern teams. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD. LNCS, vol. 4213, pp. 577–584. Springer (2006)
12. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In: KDD. pp. 80–86 (1998)
13. Nijssen, S., Guns, T., De Raedt, L.: Correlated itemset mining in ROC space: a constraint programming approach. In: KDD. pp. 647–656. ACM (2009)
14. Quinlan, J.R.: Learning logical definitions from relations. Machine Learning 5, 239–266 (1990)
15. Rückert, U., De Raedt, L.: An experimental evaluation of simplicity in rule learning. Artif. Intell. 172(1), 19–28 (2008)
16. Siebes, A., Vreeken, J., van Leeuwen, M.: Item sets that compress. In: Ghosh, J., Lambert, D., Skillicorn, D.B., Srivastava, J. (eds.) SDM. pp. 395–406. SIAM (2006)