Discovering Skylines of Subgroup Sets

Matthijs van Leeuwen[1] and Antti Ukkonen[2]

[1] Department of Computer Science, KU Leuven, Belgium
[2] Helsinki Institute for Information Technology HIIT, Aalto University, Finland
matthijs.vanleeuwen@cs.kuleuven.be, antti.ukkonen@aalto.fi

Abstract. Many tasks in exploratory data mining aim to discover the top-k results with respect to a certain interestingness measure. Unfortunately, in practice top-k solution sets are hardly satisfactory, if only because redundancy in such results is a severe problem. To address this, a recent trend is to find *diverse sets of high-quality patterns*. However, a 'perfect' diverse top-k cannot possibly exist, since there is an inherent trade-off between quality and diversity.

We argue that the best way to deal with the quality-diversity trade-off is to *explicitly consider the Pareto front, or skyline, of non-dominated solutions*, i.e. those solutions for which neither quality nor diversity can be improved without degrading the other quantity. In particular, we focus on k-pattern set mining in the context of Subgroup Discovery [6]. For this setting, we present two algorithms for the discovery of skylines; an exact algorithm and a levelwise heuristic.

We evaluate the performance of the two proposed skyline algorithms, and the accuracy of the levelwise method. Furthermore, we show that the skylines can be used for the objective evaluation of subgroup set heuristics. Finally, we show characteristics of the obtained skylines, which reveal that different quality-diversity trade-offs result in clearly different subgroup sets. Hence, the discovery of skylines is an important step towards a better understanding of 'diverse top-k's'.

1 Introduction

"Find me the k highest scoring solutions to this problem." This phrase describes the goal of many common tasks in the fields of both exploratory data mining and information retrieval. Consider for example document retrieval, where the task is to return the top-k documents that are most relevant for a given query. Also in pattern mining, it is quite common to ask for the top-k patterns with respect to a given interestingness or quality measure. As a result, many efficient algorithms for finding top-k's have been proposed in the literature.

Unfortunately, in practice top-k solution sets are hardly satisfactory to the user, which is due to two reasons. First, it is hard to formalise interestingness and therefore it is unlikely that a used quality measure completely matches perceived interestingness. Second, the top-k results are often very redundant, which can have different causes. Focusing on pattern mining, the main cause lies in the use of expressive pattern languages in which many different patterns describe the same structure in the data. Consequently, there are many patterns with almost the same interestingness or quality. When ranking the complete set of patterns according to interestingness and taking the top-k, this results in a clearly redundant result set: the top-k contains many variations of the same theme, while many potentially interesting patterns fall outside the top-k and are thus completely ignored.

Acknowledging this problem, a trend in recent years has been to move away from mining *individual patterns*, and towards mining *pattern sets* [1]. The main idea of pattern set mining is that one should mine *a diverse set of high-quality patterns*, where quality and diversity depend on the specific task. In other words, pattern set mining aims at finding a diverse top-k rather than the top-k. Note that result diversification is also very common in e.g. document retrieval.

An important observation is that a single, 'perfect' diverse top-k cannot possibly exist: whenever we replace elements from the top-k with other elements to improve diversity, it is no longer the top-k with regard to quality. This is inherent to the problem, and can be compared to the risk-return trade-off that forms the core of modern portfolio theory [11]. Consequently, many instances of pattern set mining have a trade-off between quality and diversity, even if this is often obfuscated by parameters that need tuning. However, this also implies that there exists a *Pareto front*, or *skyline*, of *non-dominated pattern sets*. In other words, there must be a set of pattern sets such that no other pattern sets exist that have both higher quality and diversity.

Subgroup Discovery Although most contributions can be generalised to other instances of pattern set mining, in the remainder of this paper we focus on Subgroup Discovery (SD) [6]. SD is an instance of pattern mining that is concerned with finding regions in the data that stand out with respect to a particular target variable. As an example, consider a dataset from the medical domain with patient information, in which we treat hemoglobin concentration as the target. By performing subgroup discovery, we could identify patterns such as $sex = male \rightarrow high$, implying that men tend to have a higher hemoglobin concentration than the overall population. Compared to other pattern mining tasks, a notable advantage is that it is pattern type agnostic and can thus deal with almost any type of data.

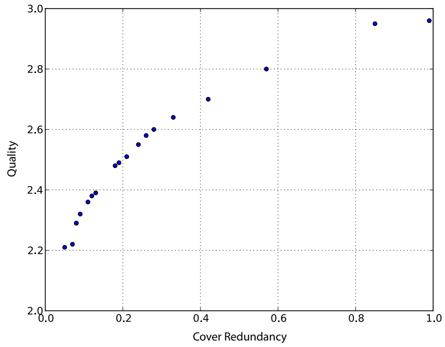

Fig. 1. The effect of varying the diversity parameter of cover-based subgroup selection [10] on quality and cover redundancy ('inverse diversity') (Car).

We recently proposed several heuristic methods for selecting *diverse top-k subgroup sets* from large sets of candidate subgroups [10]. Diversity and quality can be balanced by the user by setting several parameters. To demonstrate this effect and the resulting trade-off between quality and diversity, we conducted a series of selection experiments in which we varied the 'trade-off' parameter. The qualities and diversities of the resulting subgroup sets are plotted in Figure 1. A higher quality implies that individual subgroups have a higher quality on average, and a lower cover redundancy implies that the subgroups cover more diverse parts of the data. The figure clearly shows the trade-off: higher quality comes at the cost of less diversity and vice versa.

Approach and contributions We argue that the best way to deal with the quality-diversity trade-off is by *explicitly considering the Pareto front of optimal solutions*, i.e. those solutions for which neither quality nor diversity can be improved without degrading the other quantity. For this we have the following three arguments. First, maximising only quality or diversity clearly does not give satisfactory results, but neither is it possible to determine in advance what trade-off is desirable for a user. Second, existing heuristic methods yield a single subgroup set, but one cannot possibly know if and where it resides on the Pareto front. Neither theoretical nor empirical arguments have addressed whether the discovered subgroup sets are on or even near the Pareto front. Third and last, the trade-off is usually implicit and hidden by parameters that are hard to tune. This tuning is dataset-dependent and small parameter changes can have a big impact on the resulting trade-off. Hence, only by explicitly considering subgroup set skylines, we can learn more about its characteristics and properties, and find out what subgroup sets are preferred by users.

The approach and contributions of this paper can be summarised as follows:

1. We introduce the concept of the quality-diversity skyline for k-pattern sets, and argue that it is important to investigate its shape. Such analysis helps in making more principled choices with regard to the quality-diversity trade-off, e.g. interactively or by generalising frequent behaviour, but may also result in insights that lead to novel heuristics for pattern set mining.
2. For developing our theory and methodology, we focus on instances of the k-pattern set mining problem for which quality of the pattern set is the *sum of the individual qualities*, and diversity is quantified using *joint entropy*.
3. For this setting, we present two algorithms that compute the skyline given a set of candidate patterns.
   (a) The first is a branch-and-bound algorithm that computes the exact Pareto front. Properties of joint entropy are used to prune the search space, which results in considerable speedups and makes skyline computation feasible for modestly sized candidate sets.
   (b) The second is a heuristic that takes a levelwise approach; the Pareto front of size i is approximated by augmenting the Pareto front of subgroup sets of size $i - 1$. Although this method is a heuristic, it approximates the results of the exact method very well and is much faster.
4. We perform experiments to investigate how the methods perform in the context of subgroup sets. We study runtimes of the two skyline algorithms, and investigate how accurate the approximations of the heuristic method are. In addition, we show that the skylines can be used for the objective evaluation of

subgroup set heuristics. Finally, we show characteristics of the obtained skylines, which reveals that subgroup sets with very different quality-diversity trade-offs exist. Hence, explicitly computing and considering skylines is an important step towards a better understanding of 'diverse top-k's'.

After discussing related work in Section 2, we introduce notation and preliminaries in Section 3. Section 4 introduces Pareto optimal subgroup sets, after which Sections 5 and 6 introduce the skyline discovery algorithms. Section 7 presents the experiments and we round up with conclusions in Section 8.

2 Related work

Subgroup Discovery has been around since the late 90s [6], and is closely related to Contrast Set Mining and Emerging Pattern Mining. Subgroup Discovery is probably the most generic of these, as the general task makes very few assumptions about the data.

Diverse subgroup set discovery, introduced in [10], resembles supervised pattern set selection methods [12,2,1], but quantifying quality separately from diversity makes it substantially different. This decoupling has clear advantages, as the two can be independently varied and it becomes more apparent how each of the two perform. Supervised pattern set mining methods often aim to achieve good classification, which does not match the *exploratory* goal of Subgroup Discovery.

Entropy was used to quantify diversity of k-sized pattern sets by Knobbe and Ho [8], but our overall approach and aims are rather different. Knobbe and Ho's *maximally informative k-itemsets* maximize entropy and are therefore maximally diverse, but no other interestingness or quality measure is taken into account. Skylines consisting of individual patterns were previously studied in [13].

Our problem can also be regarded as an instance of a bicriteria or multiobjective combinatorial optimisation (MOCO) problem. Quite some approaches to MOCO problems have been proposed, both exact [4] and heuristic [5]. The main property that distinguishes our problem is that given a point in decision space, it is very hard to determine whether a feasible solution in criteria space exists, let alone to compute this solution. This is due to the large size and complex structure of the criteria space, i.e. the set of all possible pattern sets. Consequently, searching through criteria space is required to find a skyline, and this cannot be accomplished with standard MOCO approaches.

3 Preliminaries

We assume that the tuples to be analysed are described by k (≥ 1) *description attributes* A and a target attribute Y. All attributes A_i (and Y) have a domain of possible values $\text{Dom}(A_i)$ (resp. $\text{Dom}(Y)$). A dataset $\mathcal{D}$ is a bag of tuples t over the complete set of attributes $\{A_1, \ldots, A_k, Y\}$. The central concept is the *subgroup*, which consists of a *description* and a corresponding *cover*. A *subgroup cover* is a bag of tuples $G \subseteq \mathcal{D}$ and $|G|$ denotes its size, also called *subgroup size* or *coverage*. A *subgroup description* is a formula s, consisting of a conjunction of conditions on the description attributes, and its corresponding subgroup cover is the set of tuples that satisfy the formula, i.e. $G_s = \{t \in \mathcal{D} \mid t \vDash s\}$.

A *subgroup quality measure* is a function $\varphi : 2^Y \mapsto \mathbb{R}$ that assigns a numeric value to any subgroup based on its target values $\pi_Y(G)$. The traditional Subgroup Discovery (SD) task is to find the top-k ranking subgroups according to φ. Depending on the data and task, either exhaustive search or beam search can be used. Several parameters influence the search, e.g. a minimum coverage threshold requires subgroup covers to consist of at least *mincov* tuples, and the maximum depth parameter *maxdepth* imposes a maximum on the pattern length.

***k*-subgroup sets** A *k-subgroup set* S is a set consisting of exactly k subgroups. Subgroup set quality, denoted by $q(S)$, is simply the sum of the individual qualities of the k subgroups, i.e. $q(S) = \sum_{G \in S} \varphi(G)$. Any suitable quality measure can be used for $\varphi(G)$; in this paper we use Weighted Relative Accuracy [6], probably the best-known quality measure for subgroups.

To quantify diversity among the subgroup covers of a subgroup set, we use *joint entropy.* Joint entropy, denoted by H, is obtained by computing the entropy over the binary features defined by the subgroups in the set.

Definition 1 (Joint Entropy). *Given a k-subgroup set* $S = \{G_1, \ldots, G_k\}$, *and let* $B = (b_1, \ldots, b_k) \in \{0,1\}^k$ *be a tuple of binary values. Let* $p(s_{G_1} = b_1, \ldots, s_{G_k} = b_k)$ *denote the fraction of tuples* $t \in \mathcal{D}$ *such that* $s_{G_1}(t) = b_1 \wedge \ldots \wedge s_{G_k}(t) = b_k$. *The* joint entropy *of* S *is defined as:*

$$H(S) = - \sum_{B \in \{0,1\}^k} p(s_{G_1} = b_1, \ldots, s_{G_k} = b_k) \log_2 p(s_{G_1} = b_1, \ldots, s_{G_k} = b_k).$$

H is measured in bits, and each subgroup contributes at most 1 bit of information, so that $H(S) \leq |S|$. A higher entropy indicates higher diversity. Note that we decided not to use Cover redundancy [10], because joint entropy is more widely known and used, and can be used to prune the search space.

4 Skylines of Pareto optimal *k*-subgroup sets

In this section we formally state the problem that we address in this paper, i.e. that of finding skylines of Pareto optimal k-subgroup sets. Before discussing the problem in more detail, we define the notion of *dominance.*

Definition 2. *Given points* x *and* y *from some set* Ω, *and a set of functions* $\mathcal{F} = \{f_1, \ldots, f_n\}$, $f_i : \Omega \to \mathbb{R}$ *for all* $f_i \in \mathcal{F}$, *we say that point* x dominates *point* y *in terms of* $\mathcal{F}$, *denoted* $x \succ_{\mathcal{F}} y$, *if and only if* $f_i(x) \geq f_i(y)$ *for all* $f_i \in \mathcal{F}$ *and there exists at least one function* $f_i \in \mathcal{F}$ *for which* $f_i(x) > f_i(y)$.

The Pareto front, or skyline, of a set of points are those points that are not being dominated by any other point. Finding the Pareto front is in general not hard. For a given set of n points it can be computed in time $O(n^2)$ by a brute-force method, and in time $O(n \log n)$ by using an improved algorithm [9]. However, in our case n is very large, because the set of points we want to compute the Pareto front of is in fact the *power set* of the set C of candidate subgroups, denoted 2^C. That is, we are addressing the following problem:

Problem 1 (k-subset skyline). Given a discrete set C, an integer k, a set of functions $\mathcal{F} = \{f_1, \dots, f_n\}$, with $f_i : 2^C \rightarrow \mathbb{R}$ for all $f_i \in \mathcal{F}$, find the set

$$P_k = \{S \in 2^C : |S| = k \text{ and } \nexists S' \in 2^C \text{ st. } S' \succ_{\mathcal{F}} S\}.$$

Note that we only consider subsets of C of a fixed size k. The brute-force approach to Problem 1 simply materialises all k-sized subsets of C, and then runs a standard Pareto front algorithm on this. Clearly this is not going to work unless C is very small.

In our k-subgroup set application the set $\mathcal{F}$ contains two functions, one for subgroup set quality, and another for diversity. For the rest of the paper we let $\mathcal{F} = \{q, d\}$, where q is a quality measure of the set S, defined as $q(S) = \sum_{G \in S} \varphi(G)$, while d is joint entropy as given in Definition 1. Although in principle d could be any diversity measure, the exact method presented in the next section exploits properties of joint entropy for pruning the search space.

Problem 2 (k-subgroup set skyline). Given a set of subgroups C, an integer k, the set of functions $\mathcal{F} = \{q, d\}$, with $q(S) = \sum_{G \in S} \varphi(G)$ and $d(S) = H(S)$, find the k-subset skyline (as defined by Problem 1).

5 Exact algorithm

In this section we present an exact algorithm for solving Problem 2 that combines an efficient subset enumeration scheme with results from [7] to prune a substantial part of the search space.

As a first step in designing the exact algorithm, suppose that we have a list L with all k-sized subsets of C sorted in non-increasing order of $q(S)$. Clearly the first element of L must belong to the skyline as it has the highest quality of all subsets. Assign its diversity to $d_{\max}$. A simple algorithm to construct the skyline is to scan over L until we find a subset with diversity larger than $d_{\max}$. This subset is added to the skyline, we set $d_{\max}$ equal to the diversity of this subset, and continue scanning L. When the algorithm reaches the end of L we have found the exact skyline.[3]

Our exact algorithm works in the same way, but it does this *without materialising the complete list L*. Instead, we enumerate the subsets in decreasing order of quality in an online fashion with *polynomial delay*, meaning that the computation required to obtain the next subset in the sequence is polynomial in

[3] To be precise, this is only true in the absence of ties in $q(S)$. If some subsets all have the same quality, and ties were broken at random when sorting the subsets, this algorithm may include some subsets in the skyline that are dominated. However, removing these is a simple post-processing step.

Algorithm 1 Outline of an algorithm that enumerates all $S \subset C$ with $|S| = k$ in non-increasing order of $q(S)$.

```
1. $Q \leftarrow$ empty priority queue, insert $X = \{1, 2, \dots k\}$ into $Q$ with priority $q(C(X))$.
2. While $Q$ is not empty:
    (a) Pop the highest priority index set $X$ from $Q$, and output $C(X)$.
    (b) Insert every $X' \in N(X)$ into $Q$ with $q(C(S'))$ as the priority *unless* $X'$ has
        already been inserted into $Q$.
```

$|C|$ and k. In Subsections 5.1 and 5.2 we describe how to make the enumeration efficient and how to prune the search space, we now continue with the main idea.

Without loss of generality, assume that the candidate subgroups $G \in C$ are sorted in non-increasing order of $\varphi(G)$, and let $C(i)$ denote the subgroup at position i. That is, we have $\varphi(C(i)) \geq \varphi(C(j))$ for every $i < j$. Let $X \subset \{1, \dots, |C|\}$, $|X| = k$, denote a set of indices to C that induce the k-subgroup set $C(X)$. To simplify notation, we sometimes write $f(X)$ in place of $f(C(X))$ for $f \in \{q, d\}$.

Definition 3. *The* i-neighbour *of* X*, denoted* $n_i(X)$*, is a copy of* X *with the index at position* i *incremented by one. More formally:*

$$n_i(X) = \begin{cases} \{X_1, \dots, X_i + 1, \dots X_k\} & \textit{if } i < k \textit{ and } X_i + 1 < X_{i+1}, \\ \{X_1, \dots, X_k + 1\} & \textit{if } i = k \textit{ and } X_k + 1 \leq |C|, \\ \emptyset & \textit{otherwise.} \end{cases}$$

The neighbourhood *of* X*, denoted* $N(X)$*, is the set* $\{n_i(X) \mid i = 1, \dots, k\}$*. Finally, the set* X *is a* parent *of the set* X' *whenever* $X' \in N(X)$*.*

For example, if $X = \{1, 3, 5\}$, its neighbourhood contains the sets $\{2, 3, 5\}$, $\{1, 4, 5\}$, and $\{1, 3, 6\}$, while for $X = \{1, 2, 3\}$ we have $N(X) = \{\{1, 2, 4\}\}$[4]. Observe that the neighbourhood $N(X)$ of a set X *only contains sets having at most the same quality as* X, i.e. we have $q(X) \geq q(X')$ for every $X' \in N(X)$.

Using this, we can generate the list L on the fly by following the procedure shown in Algorithm 1. The algorithm starts from $\{1, \dots, k\}$ and maintains a priority queue of subsets with $q(S)$ as the priority value.

Proposition 1. *Algorithm 1 is both* complete *and* correct *in the sense that it enumerates all* k*-sized subsets in non-increasing order of* $q(S)$*.*

Proof. Correctness: We show that a subset output by the algorithm can not have a *larger* quality than any subset that was output before. By the mechanics of the algorithm, every index set X' in priority queue Q must belong to the neighbourhood $N(X)$ of some $C(X)$ that was already output. And by definition, all $X' \in N(X)$ have quality at most $q(X)$. Hence Q can only contain subsets with qualities that are upper bounded by qualities of the already generated subsets. This implies that the subsets are output in non-increasing order of $q(S)$.

[4] The empty sets in $N(X)$ are not considered.

Completeness: The algorithm outputs every k-sized subset of C. Clearly every subset that enters Q is eventually output. Algorithm 1 fails to output a subset S if and only if none of its parents is output, because whenever a parent of S is output, S is put into Q. Consider one possible chain of parents from any index set X until we reach the set $\{1,\ldots,k\}$. Because Q is initialised with $\{1,\ldots,k\}$, this chain must exist, meaning that every subset in the chain has at least one parent in Q. This implies that X has at least one parent in Q, and thus $S = C(X)$ is found. □

Notice that the amount of computation needed between two subsets is $O(k|C|)$. Size of the priority queue Q is trivially upper bounded by $2^{|C|}$, which means that insertions and extractions to Q are linear in the size of C using e.g. a binomial heap to implement Q. The size of $N(X)$ is upper bounded by k, meaning we need at most k insertions and one extraction.

5.1 Efficient subset enumeration

A problem with the enumeration scheme given in Algorithm 1 is that it generates the entire neighbourhood $N(X)$ for every X, and these neighbourhoods are partly overlapping. Consider for example the set $\{1,3,5\}$, which is both the 2-neighbour of $\{1,2,5\}$, and the 3-neighbour of the set $\{1,3,4\}$. Algorithm 1 would thus generate the set $\{1,3,5\}$ twice. To avoid this, it must keep track of every set that was inserted into Q at some point. This is clearly undesirable, as the amount of space needed is $O(2^{|C|})$. We thus need an algorithm that generates every subset of size k *once and only once* without additional bookkeeping.

Enumerating subsets once and only once is of course a known problem. However, our situation has the additional challenge that we want to generate the subsets in decreasing order of $q(S)$ with polynomial delay (preferably $O(k|C|)$). An approach that combines the priority queue with a simple subset enumeration scheme, e.g. depth-first traversal, does not satisfy this property. Instead, we use a modification to Algorithm 1 that maintains its computational properties, but avoids duplicates. The idea is to insert only *a subset of* $N(X)$ to Q on every iteration. This subset can be chosen so that Proposition 1 still holds. To this end, we arrange the search space of all k-sized subsets in a directed graph T.

Definition 4. *Given the set C and the integer k, the directed graph T has the node (X,j) for every $X \subset \{1,\ldots,|C|\}$, $|X| = k$. Here $j \in \{1,\ldots,k\}$ is the* position *associated with index set X in the given node. Node (X,j) has neighbours*

$$\{(n_i(X), i) \mid i = 1,\ldots,j\}. \tag{1}$$

That is, the neighbours of (X,j) in T are those i-neighbours of X where the modifications take place in the first j positions of X.

Proposition 2. *The graph T is a tree rooted at $(\{1,\ldots,k\},k)$.*

Proof. First, observe that $(\{1,\ldots,k\},k)$ can have no incoming edges, because by Definitions 3 and 4 such an edge should come from a node with a value smaller

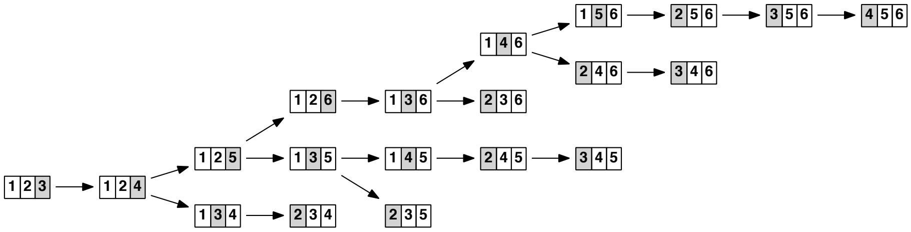

Fig. 2. Example of the subset graph T for 3-sized subsets of the set $\{1, 2, \ldots, 6\}$. The *position* of a node (see Definition 4) is indicated by a light grey background.

than k at position k. Clearly such a node cannot exist, because k is the smallest value that a node may have at position k. Second, every node (X, i) except the root has one, and only one incoming edge. Suppose this edge comes from the node (X', j), with $j \geq i$. By Definition 3, the i-neighbour of a set X' (if it exists) is identical to X' except at position i which is incremented by one. This means that the set X of the node (X, i) must have $X_i = X'_i + 1$ and $X_h = X'_h$ for every $h \neq i$. Clearly there is only one $X' \subset \{1, \ldots, |C|\}$ that satisfies this, and since by definition of T no two nodes of T may contain the same subset, there can be only one node with an outgoing edge to (X, i). □

An example of such a tree for all 3-sized subsets of $\{1, \ldots, 6\}$ is shown in Figure 2. Any algorithm that traverses T will generate every k-sized subset once and only once. We can thus enumerate the subsets in a way that avoids duplicates by making Algorithm 1 traverse the tree T. This is accomplished simply by keeping nodes of T in the priority queue Q, and only inserting those nodes to Q that are neighbours of (X, j) in T (which can be determined without materialising T).

5.2 Entropy-based pruning of the search space

Algorithm 1 scans over a sorted list L of all possible subsets, but in practice L is too long already with small C and k. Next we discuss how to use a property of joint entropy and the tree T to *prune* parts of list L. Recall that at every stage of the algorithm we know that $d_{\max}$ is the largest diversity observed so far. If we can show that no subset below a given node of T can have diversity larger than $d_{\max}$, we can skip the entire subtree. From Figure 2 we can make the following observations:

Observation 1: For any (X, j) of T, the suffix starting at position $(j + 1)$, denoted R^X_{j+1}, is the same in every node of the subtree rooted at (X, j). For example, all nodes below node $(\{1, 3, 6\}, 2)$ contain the element 6 at position 3.

Observation 2: For any (X, j) of T with $j < k$, in every node that appears in the subtree rooted at (X, j), the elements at positions $1, \ldots j$ all have a value less than X_{j+1}. For example, all nodes below $(\{1, 3, 5\}, 2)$ only contain elements that are less than 5 at positions 1 and 2.

Algorithm 2 EXACT(C, k)

```
$X \leftarrow \{1, 2, \ldots, k\}$,  $Q \leftarrow$ empty priority queue,  $d_{\max} \leftarrow d(X)$,  $P_k \leftarrow \emptyset$
push $(X, k)$ into $Q$ with priority $q(X)$
while  $Q$ is not empty  do
    $(X, j) \leftarrow$ highest priority item from $Q$
    if  $d(X) \geq d_{\max}$  then
        $P_k \leftarrow P_k \cup \{C(X)\}$
        $d_{\max} \leftarrow d$
    for  every $(X', i) \in \{(n_i(X), i) \mid i = 1, \ldots, j\}$ do
        if  $\tilde{d}(X') \geq d_{\max}$  then
            insert $(X', i)$ into $Q$ with priority $q(X')$
return $P_k$
```

These observations are general properties of the tree T, and can be shown to follow from Definitions 3 and 4. Together with the following proposition we can use these to prune entire subtrees of T.

Proposition 3. *(Prop. 4 of [7]): Let $S \subset C$ denote a subgroup set, and let H as in Def. 1. Suppose that $\{B_1, \ldots, B_m\}$ is a partition of S. Then $H(S) \leq \sum_{i=1}^{m} H(B_i)$.*

For any (X, j) in T, let $\hat{X}$ denote any index set that appears in the subtree rooted at (X, j). Given (X, j), we must find an upper bound for $d(\hat{X}) = H(\hat{X})$. From Obs. 1 we know that the suffix R^X_{j+1} is the same in X and $\hat{X}$. We have thus $H(R^X_{j+1}) = H(R^{\hat{X}}_{j+1})$ for every $\hat{X}$. Obs. 2 tells us that only certain elements may occur in the first j positions of $\hat{X}$. We can compute the singleton entropies $d(G) = H(G)$ for every subgroup $G \in C$. For $j < k$, let $Z(X, j)$ denote the set of j highest entropy subgroups in the first $X_{j+1} - 1$ positions of C. Now, by construction and Prop. 3, the sum $\sum_{G \in Z(X,j)} H(G)$ must be an upper bound for the entropy of the first j positions of any $\hat{X}$. This means that for all $\hat{X}$ we have

$$d(\hat{X}) \leq d(R^X_{j+1}) + \sum_{G \in Z(X,j)} d(G) = \tilde{d}(X),$$

which gives the desired upper bound. If $\tilde{d}(X) < d_{\max}$, the entire subtree rooted at (X, j) can be pruned.

The algorithm EXACT, shown in Algorithm 2, implements all details discussed in this section. It combines the improved subset enumeration scheme with the pruning results. While this is a substantial improvement over the basic scheme of Alg. 1, the size of the priority queue can still be exponential in $|C|$, as it is proportional to the size of a *cut* of the subset tree T.

6 A greedy levelwise algorithm

The exact algorithm we described above has two drawbacks: 1) it is optimised for the joint entropy diversity function of Definition 1, and 2) it requires exponential

Algorithm 3 LEVELWISE(C, k)

```
$P_2 \leftarrow$SKYLINE$(C \times C)$, $i \leftarrow 3$
while $i \leq k$ do
    $P_i \leftarrow \emptyset$
    for $S \in P_{i-1}$ do
        $P_i \leftarrow$SKYLINE$(P_i \cup \{S \cup c \mid c \in \{C \setminus S\}\})$
    $i \leftarrow i + 1$
return $P_k$
```

space. Of these point 1) makes the method ill-suited for some applications, while point 2) rules out large candidate sets C and subgroup set sizes k. Because of this we also introduce a greedy heuristic for finding the skyline.

In general the main computational bottleneck is caused by the large number of points that must be considered when computing the skyline. However, the resulting skyline itself is very likely going to be orders of magnitude smaller than $\binom{|C|}{k}$. Our algorithm will exploit this property. Moreover, consider the Pareto fronts for sets of size $i-1$ and i, denoted P_{i-1} and P_i, respectively. It seems unlikely that a subset of size $i-1$ that is very far from P_{i-1} in terms of the functions $\mathcal{F}$ would have a superset that belongs to P_i. On the other hand, subsets in P_{i-1} might be more likely to have a superset that belongs to P_i.

We propose an algorithm that constructs an approximate skyline P_k one level at a time starting from P_2. Given the set P_{i-1}, we define P_i as the skyline of the points that are obtained by combining every point in P_{i-1} with every unused candidate in C. More formally, we can define P_i recursively as follows:

$$P_i = \begin{cases} \text{SKYLINE}\left(\{(S, c) \mid S \in P_{i-1}, c \in \{C \setminus S\}\}\right) & \text{if } i > 2, \\ \text{SKYLINE}\left(\{(c_1, c_2) \in \{C \times C\} \mid c_1 \neq c_2\}\right) & \text{if } i = 2, \end{cases} \tag{2}$$

Here SKYLINE is any algorithm that computes skylines in two dimensions.

We put these ideas together in the LEVELWISE algorithm shown in Algorithm 3. It first computes P_2 exactly by considering all subgroup pairs in $C \times C$. In subsequent steps the algorithm applies Equation 2 until it reaches P_k. In practice we obtain better performance by not materialising the entire Cartesian product of P_{i-1} and C in one step, but by incrementally "growing" the set P_i.

7 Experiments

In this section we empirically evaluate the proposed approach and methods.

Datasets: Table 1 presents the datasets that we use, which were all taken from the UCI Machine Learning repository[5]. For each dataset we give the number of tuples, the number of discrete resp. numeric attributes, and the domain size y of the target attribute. Target attributes with more than two classes are treated as binary by considering the majority class as target.

[5] http://archive.ics.uci.edu/ml/

Candidate sets: Candidate sets C of subgroups are generated using exhaustive search. As conditions for the discrete attributes, $A_i = c$ and $A_i \neq c$ for all constants $c \in Dom(A_i)$ are considered. For numeric attributes, conditions $A_i > c$ and $A_i < c$ are considered, where the 'split' values are determined by local binning of occurring values into 6 equal-sized bins.

Table 1. Dataset properties.

dataset	$\lvert\mathcal{D}\rvert$	$\lvert A^{\text{disc}}\rvert$	$\lvert A^{\text{num}}\rvert$	y
Adult	48842	8	6	2
Car	1728	6	0	4
Cmc	1473	7	2	3
Credit-A	690	9	6	2
Credit-G	1000	13	7	2
Mushroom	8124	22	0	2
Pima	768	0	8	2
Tictactoe	958	9	0	2

Search parameters are chosen to result in short (and thus simple) subgroup descriptions, substantial subgroup sizes, and reasonably sized candidate sets. We set $maxdepth = 2$ and $mincov = 5\% \times |\mathcal{D}|$ (except for Adult and Mushroom: $10\% \times |\mathcal{D}|$). For those experiments for which a p-value is given, a permutation test [3] that aims to eliminate false discoveries is used to prune the candidate set. For the remaining experiments, all subgroups found are used as candidates.

Evaluation: We need measures to compare two skylines, P and P'. Intuitively, P is better than P' if there are more points in P that dominate points in P' than vice versa. We denote the fraction of points in P' that are dominated by at least one point in P by $\#\{P \succ P'\}$. For these points, we also measure by how much a skyline dominates another skyline. This is expressed by the quantity $\Delta_f(P \succ P')$, defined as the median of the set $\{(f(S) - f(S'))/f(S') \mid S \in P, S' \in P', S \succ S'\}$, i.e. the median of the relative differences between dominated sets and the sets that dominate it.

7.1 Exact and levelwise skyline discovery

Table 2 presents the results obtained on all datasets with the EXACT and LEVELWISE algorithms. The candidate sets for the first six datasets were pruned using the aforementioned permutation test. Due to long runtimes the exact method was only used with $k = 5$, the levelwise method was also used with $k = 10$. The last two datasets are too large to be used with the exact method, but for the levelwise approach no pruning of the candidate set was needed.

Runtimes greatly vary depending on the dataset and desired subgroup set size. This can be explained by the large variation in the number of 'points' in the search space that need to be explored: for Car with LEVELWISE and $k = 5$ only 4380 subgroup sets are considered, but for Credit-A with EXACT and $k = 5$ a staggering amount of 2.7×10^8 points is considered. Although a large number, this is still only a small fraction of the total search space: 10^{-2}. The exact method explored the same fraction for all datasets, implying that its pruning is effective: 99% of the search space is pruned. Despite this, runtimes are still quite long.

The greedy, levelwise approach explores much smaller parts of the search space: fractions between 10^{-23} and 10^{-4} are reported. The natural question is whether the resulting skylines approximate the exact solutions well. Looking at the skyline sizes, we observe that skylines generally consist of modest numbers of

Table 2. Results with exact and levelwise algorithms. For each experiment, we give dataset, used p-value for the permutation test (if any), candidate set size $|C|$, algorithm, and subgroup set size k. Then follow runtime, the number of points in the search space considered, the fraction of the complete search space considered, the size of the resulting skyline, and corresponding quality and diversity ranges [min, max].

dataset	p	$\|C\|$	method	k	time	#points	fraction	$\|S\|$	$q(S)$	$d(S)$
Car	10^{-3}	71	level	5	< 1s	4380	10^{-4}	55	$[0.23, 0.41]$	$[3.49, 4.90]$
			exact	5	92s	179219	10^{-2}	128	$[0.21, 0.41]$	$[3.49, 5.00]$
			level	10	2s	25211	10^{-8}	99	$[0.56, 0.74]$	$[5.56, 8.43]$
Cmc	10^{-3}	98	level	5	< 1s	10908	10^{-4}	58	$[0.18, 0.26]$	$[1.57, 4.61]$
			exact	5	680s	1545129	10^{-2}	71	$[0.17, 0.26]$	$[1.57, 4.61]$
			level	10	5s	57105	10^{-9}	147	$[0.35, 0.50]$	$[3.39, 7.05]$
Credit-A	10^{-7}	232	level	5	10s	126516	10^{-5}	305	$[0.42, 0.89]$	$[1.41, 4.56]$
			exact	5	20.5h	274696613	10^{-2}	460	$[0.41, 0.89]$	$[1.41, 4.56]$
			level	10	139s	616899	10^{-12}	613	$[0.89, 1.75]$	$[2.13, 6.87]$
Credit-G	10^{-7}	114	level	5	1s	20493	10^{-4}	133	$[0.22, 0.37]$	$[1.61, 4.74]$
			exact	5	3347s	10149837	10^{-2}	137	$[0.22, 0.37]$	$[1.61, 4.74]$
			level	10	34s	145703	10^{-9}	435	$[0.46, 0.73]$	$[1.88, 7.77]$
Pima	10^{-7}	166	level	5	1s	28509	10^{-5}	66	$[0.31, 0.49]$	$[2.62, 4.77]$
			exact	5	1h	12221188	10^{-2}	107	$[0.29, 0.49]$	$[2.62, 4.84]$
			level	10	10s	127252	10^{-11}	248	$[0.60, 0.96]$	$[2.85, 7.75]$
Tictactoe	10^{-2}	90	level	5	2s	22303	10^{-4}	78	$[0.20, 0.37]$	$[3.36, 4.94]$
			exact	5	390s	1107493	10^{-2}	281	$[0.19, 0.37]$	$[3.36, 4.94]$
			level	10	4s	53181	10^{-9}	75	$[0.41, 0.67]$	$[5.68, 7.89]$
Adult		5116	level	5	55h	15870624	10^{-10}	252	$[0.10, 0.48]$	$[1.48, 4.99]$
			level	10	85h	37774186	10^{-23}	2040	$[0.28, 0.96]$	$[1.83, 9.45]$
Mushroom		1617	level	5	785s	2439272	10^{-8}	601	$[0.27, 1.09]$	$[1.85, 4.97]$
			level	10	2940s	9230365	10^{-19}	1310	$[0.50, 2.12]$	$[2.36, 9.06]$

subgroup sets (in particular given the total number of subgroup sets). levelwise tends to find slightly smaller skylines, which is perhaps unsurprising as it explores a relatively small part of the search space. However, are the subgroup sets that it does find on the 'true' skyline?

The minimum-maximum values of subgroup set quality and diversity indicate that the levelwise skylines span almost the same ranges as the exact skylines. Table 3 shows a more elaborate comparison. The second column shows the fraction of points on the levelwise skyline that are dominated by any point on the exact skyline. This reveals that substantial parts of the approximation are on the Pareto front. For those points that are dominated, it is of interest to investigate

Table 3. Comparison of the exact (E) and levelwise (L) skylines with $k = 5$.

dataset	$\#\{E \succ L\}$	$\Delta_q(E \succ L)$	$\Delta_d(E \succ L)$
Car	22.73%	6.48%	1.49%
Cmc	51.72%	0.92%	0.69%
Credit-A	64.59%	1.04%	0.46%
Credit-G	5.31%	0.10%	0.19%
Pima	39.39%	0.78%	0.33%
Tictactoe	0%	–	–

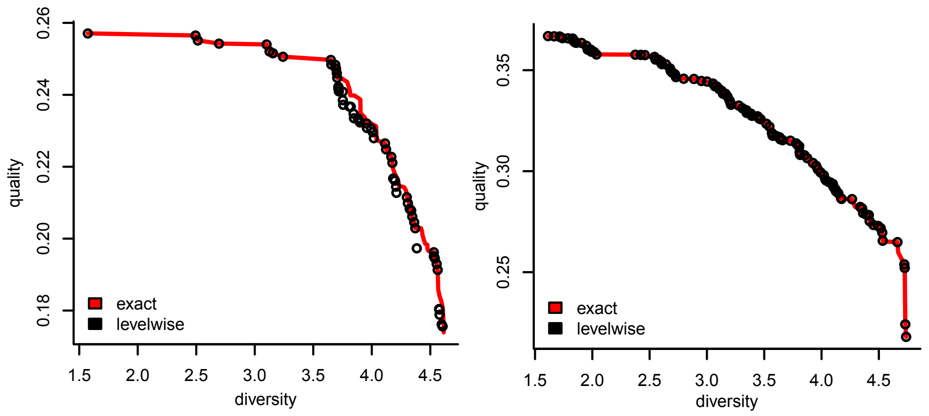

Fig. 3. Comparing exact and levelwise skylines (k=5); Cmc (left) and Credit-G (right).

by how much. This is shown in the rightmost two columns, for both quality and diversity. These numbers are very small and indicate that the levelwise skylines approximate the exact skylines very well. To further illustrate this, consider the skylines plotted in Figure 3. The Credit-G approximate skyline is virtually identical to the exact one, and the Cmc approximation, one of the worst according to Table 3, is still a close approximation for all practical purposes.

The results obtained on Adult and Mushroom, shown at the bottom of Table 2, demonstrate that LEVELWISE can handle moderately sized datasets and candidate sets, although runtimes may increase substantially. However, note that the levelwise approach finds all skylines up to k – *not just for* k. For Adult, for example, 85 hours of runtime gives all skylines for $k = 2$ up to and including $k = 10$. Provisional results can be inspected at any time, and search can be terminated when the user thinks k is large enough. Such an approach is impossible with the exact method, as it only enumerates subgroup sets of exactly size k.

Finally, it is important to note that the quality and diversity ranges spanned by the skylines are often quite large (see Table 2 and Figure 3). For example, for both Cmc and Credit-G diversity ranges from 1.5 up to almost 5 bits – taking into account that 5 bits is the maximum for $k = 5$, this implies large differences between the corresponding subgroup sets. This demonstrates that very different trade-offs between quality and diversity are possible, and hence investigating these skylines is useful. For Cmc, for example, Figure 3 shows that diversity can be increased from 1.5 to 3 bits without affecting quality much. Such knowledge is likely to influence the preference of the end user.

7.2 Evaluating a heuristic subgroup set selection method

In the introduction, we argued that disadvantages of existing heuristic subgroup selection methods are that 1) it is unknown whether the resulting subgroup sets are Pareto optimal, and 2) it is hard to tune the quality-diversity trade-off. To illustrate this, we evaluate a heuristic using LEVELWISE skylines.

Table 4. Comparing LEVELWISE (L) with the entropy-based selection heuristic (H).

dataset	k	$\#\{H \succ L\}$	$\Delta_q(H \succ L)$	$\Delta_d(H \succ L)$	$\#\{L \succ H\}$	$\Delta_q(L \succ H)$	$\Delta_d(L \succ H)$
Car	5	3.45%	0.00%	0.01%	75.86%	16.36%	7.29%
	10	1.79%	0.01%	0.00%	86.21%	12.65%	6.37%
Cmc	5	15.38%	1.67%	1.52%	65.52%	12.06%	2.07%
	10	0.43%	0.00%	0.01%	82.76%	6.03%	1.12%
Credit-A	5	0.00%	-	-	86.21%	1.99%	1.06%
	10	0.09%	0.00%	0.01%	79.31%	2.36%	1.94%
Credit-G	5	0.00%	-	-	68.97%	4.32%	0.27%
	10	0.00%	-	-	82.76%	6.16%	0.71%
Mushroom	5	0.67%	0.00%	0.01%	79.31%	10.92%	5.18%
	10	0.00%	-	-	93.10%	6.35%	4.68%
Pima	5	0.91%	0.00%	0.01%	51.72%	2.65%	0.36%
	10	0.59%	0.00%	0.01%	89.66%	2.36%	0.62%
Tictactoe	5	0.00%	-	-	62.07%	9.03%	1.08%
	10	0.00%	-	-	93.10%	4.73%	1.92%

The recently proposed cover-based subgroup set selection heuristic [10] selects a diverse k-subgroup set from a candidate set, but preliminary experiments revealed that it performed badly in terms of our diversity measure, i.e. joint entropy. We therefore slightly modified it to use entropy as selection criterion:

Given a candidate set C, first order it descending by subgroup quality (φ). Initialise subgroup set S to contain only the highest-quality subgroup and remove it from C. Then, iteratively add subgroups until $|S| = k$. In each iteration, pick that subgroup $G \in C$ that maximises $\alpha\varphi(G)+(1-\alpha)H(S\cup\{G\})$, where $\alpha \in [0,1]$ is a parameter. The selected subgroup is added to S and removed from C.

With this entropy-based heuristic, a single subgroup set can be found. By carefully varying the α parameter, we obtain a 'skyline' consisting of 29 subgroup sets for each dataset (except for Adult, for which running this heuristic many times took too long). Table 4 presents a comparison of this skyline to the one obtained with LEVELWISE. The heuristic rarely finds better solutions than the levelwise method, but most of the solutions found by the heuristic are dominated by the levelwise skyline. Also when considering the relative differences in quality and diversity for those points that are dominated, it is clear that the levelwise method often finds much better skylines than the heuristic.

We conclude that even when carefully tuning the parameter of a pattern set selection method, this does not guarantee that we discover a set of Pareto optimal solutions. Furthermore, this comparison also demonstrates that skyline discovery can be a useful tool in the evaluation of (existing) heuristics.

8 Conclusions

We have argued that whenever there is a quality-diversity trade-off in a k-subset selection task, it is important to explicitly consider the skyline of Pareto optimal solutions. In this paper we focused on the task of pattern set selection in the context of Subgroup Discovery, but many similar 'diverse top-k' tasks exist, not only in exploratory data mining but also for example in information retrieval.

We proposed two algorithms for discovering skylines of k-subgroup sets. If we use joint entropy as diversity measure and use its properties for pruning, the EXACT method can be used with modest candidate sets and k. LEVELWISE performs a greedy, levelwise search and is therefore considerably faster. Furthermore, experiments showed that the obtained skylines closely approximate the exact solutions. Finally, we demonstrated that the skyline discovery algorithms can be used for the objective evaluation of heuristic selection methods.

One might argue that having multiple subgroup sets instead of one only complicates the situation, but observe that this skyline *always exists*; the problem is that users may not be aware of this. Therefore, the explicit discovery of skylines is an important step towards a better understanding of 'diverse top-k's'. Skylines of subgroup sets can be interactively explored, allowing the user to make informed choices based on both the subgroup sets and the shape of the skyline.

Acknowledgements. Matthijs van Leeuwen is supported by a Rubicon grant of the Netherlands Organisation for Scientific Research (NWO).

References

1. B. Bringmann, S. Nijssen, N. Tatti, J. Vreeken, and A. Zimmermann. Mining sets of patterns: Next generation pattern mining. In *Tutorial at ICDM'11*, 2011.
2. B. Bringmann and A. Zimmermann. The chosen few: On identifying valuable patterns. In *Proceedings of the ICDM'07*, pages 63–72, 2007.
3. W. Duivesteijn and A. Knobbe. Exploiting false discoveries – statistical validation of patterns and quality measures in subgroup discovery. In *Proceedings of the ICDM'11*, pages 151–160, 2011.
4. M. Ehrgott and X. Gandibleux. A survey and annoted bibliography of multiobjective combinatorial optimization. *OR Spektrum*, 2000.
5. M. Ehrgott and X. Gandibleux. Approximative solution methods for multiobjective combinatorial optimization. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, 12(1):1–63, 2004.
6. W. Klösgen. *Advances in Knowledge Discovery and Data Mining*, chapter Explora: A Multipattern and Multistrategy Discovery Assistant, pages 249–271. 1996.
7. A. Knobbe and E.K.Y. Ho. Maximally informative k-itemsets and their efficient discovery. In *Proceedings of the KDD'06*, pages 237–244, 2006.
8. A. Knobbe and E.K.Y. Ho. Pattern teams. In *Proceedings of the ECML PKDD'06*, pages 577–584, 2006.
9. H. T. Kung, F. Luccio, and F. P. Preparata. On finding the maxima of a set of vectors. *J. ACM*, 22(4):469–476, 1975.
10. M. van Leeuwen and A. Knobbe. Diverse subgroup set discovery. *Data Mining and Knowledge Discovery*, 25:208–242, 2012.
11. Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.
12. H. Peng, F. Long, and C. Ding. Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, 2005.
13. A. Soulet, C. Raïssi, M. Plantevit, and B. Crémilleux. Mining dominant patterns in the sky. In *Proceedings of the ICDM'11*, pages 655–664, 2011.