Using Constraints to Discover Sparse and Alternative Subgroup Descriptions

Jakob Bach
Independent researcher*
jakob.bach.ka@gmail.com

Abstract

Subgroup-discovery methods allow users to obtain simple descriptions of interesting regions in a dataset. Using constraints in subgroup discovery can enhance interpretability even further. In this article, we focus on two types of constraints: First, we limit the number of features used in subgroup descriptions, making the latter sparse. Second, we propose the novel optimization problem of finding alternative subgroup descriptions, which cover a similar set of data objects as a given subgroup but use different features. We describe how to integrate both constraint types into heuristic subgroup-discovery methods. Further, we propose a novel Satisfiability Modulo Theories (SMT) formulation of subgroup discovery as a white-box optimization problem, which allows solver-based search for subgroups and is open to a variety of constraint types. Additionally, we prove that both constraint types lead to an $\mathcal{NP}$-hard optimization problem. Finally, we employ 27 binary-classification datasets to compare algorithmic and solver-based search for unconstrained and constrained subgroup discovery. We observe that heuristic search methods often yield high-quality subgroups within a short runtime, also in scenarios with constraints.

Keywords: subgroup discovery, alternatives, constraints, satisfiability modulo theories, explainability, interpretability, XAI

1 Introduction

Motivation The interpretability of prediction models has gained importance in recent years [26, 89]. There are various ways to foster interpretability in machine-learning pipelines. In particular, some machine-learning models are simple enough to be intrinsically interpretable [26], like subgroup descriptions. Subgroup discovery aims to identify 'interesting' subsets of a dataset [5], such as data objects sharing a specific class label, that can be characterized by concise conditions on feature values. Subgroup-discovery methods have recently

*Most of the research for this article was carried out while the author was affiliated with the Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany.

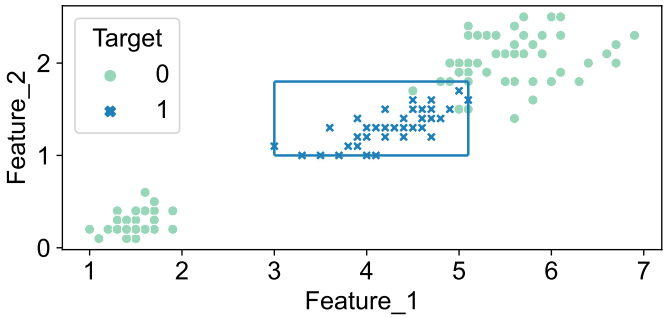


Figure 1: Exemplary subgroup description in the form of a rectangle for a dataset with two real-valued features and a binary prediction target.

been employed in various fields, such as chemistry [75], medicine [36], database engineering [103], decision making [116], and social sciences [59].

Figure 1 displays an exemplary rectangle-shaped subgroup description for a two-dimensional, real-valued dataset with a binary prediction target. This subgroup is defined by $(\mathit{Feature\_1} \in [3.0, 5.1]) \wedge (\mathit{Feature\_2} \in [1.0, 1.8])$ and contains a considerably higher fraction of data objects with $\mathit{Target} = 1$ than the complete dataset. While such subgroup descriptions already tend to be understandable for users, we see further potential to increase interpretability with the help of constraints.

Problem statement This article addresses the problem of constrained subgroup discovery. In particular, we focus on two types of constraints related to the features used in subgroup descriptions:

First, *feature-cardinality constraints* limit the number of selected features, i.e., features used in subgroup descriptions. Thus, the subgroup descriptions become *sparse*, which increases their interpretability at the potential expense of subgroup quality. E.g., in Figure 1, one can use bounds on either feature instead of both to define a subgroup still containing all data objects with $\mathit{Target} = 1$. Such a description is simpler but covers more data objects with $\mathit{Target} = 0$. In general, even intrinsically interpretable models may lose interpretability if they involve too many features [86, 89]. Further, feature selection [50, 76] is common for other machine-learning tasks than subgroup discovery as well.

Second, we formulate constraints to search *alternative subgroup descriptions*: Given an *original* subgroup, an alternative subgroup description should use different features but cover a similar set of data objects. E.g., in Figure 1, one may define a subgroup with an interval on one feature and then try to cover a similar set of data objects with the other feature. With alternative subgroup descriptions, users obtain different explanations for the same subgroup. Such alternative explanations are also popular in other explainable-AI techniques like counterfactuals [91, 105], e.g., to enable users to develop and test multiple hypotheses or foster trust in the predictions [61, 114].

Related work There are various search methods for subgroup discovery, exhaustive [8, 24, 46, 73] as well as heuristic [38, 69, 82, 101] ones. We see research gaps in three aspects: First, all widely used subgroup-discovery methods are algorithmic in nature and only support a limited set of constraints, as the search routines need to be specifically adapted to particular constraint types. Second, the number of features used in a subgroup description is a well-known measure for subgroup complexity [52, 53, 113]. However, there is a lack of systematic evaluations for this constraint type, particularly regarding evaluations with different cardinality thresholds and comparing multiple subgroup-discovery methods. Third, various subgroup-discovery methods yield a diverse set of subgroups rather than only one subgroup, thereby providing alternative solutions [19, 24, 69, 73, 81, 101]. However, this notion of alternatives targets at covering different subsets of data objects from the dataset. In contrast, our notion of alternative subgroup descriptions tries to cover a similar set of data objects as in the original subgroup but with different features in the description.

Contributions Our contribution is fivefold:

First, we formalize subgroup discovery as a Satisfiability Modulo Theories (SMT) optimization problem. This novel white-box formulation admits a solver-based search for subgroups and allows integrating and combining a variety of constraints in a declarative manner.

Second, we formalize two constraint types for this optimization problem, i.e., feature-cardinality constraints and alternative subgroup descriptions. For the latter, we allow users to control alternatives with two parameters, i.e., the number of alternatives and a dissimilarity threshold. We integrate both constraint types into our white-box formulation of subgroup discovery.

Third, we describe how to integrate these two constraint types into three existing heuristic search methods and two novel baselines for subgroup discovery. The latter are faster and simpler than the former, so they may serve as additional reference points for future experimental studies on subgroup discovery.

Fourth, we analyze the time complexity of the subgroup-discovery problem with each of these two constraint types. In particular, we prove several $\mathcal{NP}$-completeness results and thereby show that finding optimal solutions under these constraint types is computationally challenging.

Fifth, we conduct comprehensive experiments with 27 binary-classification datasets from the Penn Machine Learning Benchmarks (PMLB) [96, 104]. We compare solver-based and seven algorithmic subgroup-discovery methods in different experimental scenarios: without constraints, with a feature-cardinality constraint, and for searching alternative subgroup descriptions. In particular, we evaluate the runtime of subgroup discovery and the quality of the discovered subgroups. We also analyze how the subgroup quality in solver-based search depends on the solver's timeout. We publish all code and data (cf. Section 5.6).

Experimental results In our experimental scenario without constraints, the heuristic search methods yield similar subgroup quality as solver-based search.

On the test set, the heuristics may even be better since they show less overfitting, i.e., a lower gap between training-set quality and test-set quality. Additionally, the solver-based search is one to two orders of magnitude slower. Using a solver timeout, a large fraction of the final subgroup quality can be reached in a fraction of the runtime, though this quality is lower than for equally fast heuristics.

With a feature-cardinality constraint, heuristic search methods are still competitive quality-wise compared to solver-based search. Further, subgroups that only use a few features show relatively high quality compared to unconstrained subgroups. I.e., there is a decreasing marginal utility in selecting more features. Additionally, feature-cardinality constraints reduce overfitting.

For alternative subgroup descriptions, heuristics also yield similar quality as solver-based search. Our two user parameters for alternatives control the solutions as expected: The similarity to the original subgroup and the quality of the alternatives decrease for more alternatives and a higher dissimilarity threshold.

Outline Section 2 introduces fundamentals. Section 3 proposes two baselines for subgroup discovery. Section 4 describes and analyzes constrained subgroup discovery. Section 5 outlines our experimental design, while Section 6 presents the experimental results. Section 7 reviews related work. Section 8 concludes and discusses future work. Appendix A contains supplementary materials.

Other versions There is a conference version of this article [14], which retains the most important experimental and theoretical results. It evaluates the same run of the experimental pipeline as this arXiv version (v2).

The dissertation [13] contains a shortened rendition of the first arXiv version, though longer than the conference version. It lacks the comparison against the exhaustive algorithmic search methods *BSD* and *SD-Map*.

Appendix A.1.1, A.1.3, and A.3 appear exclusively in this arXiv version.

2 Fundamentals of Subgroup Discovery

In this section, we describe fundamentals. First, we introduce the optimization problem of subgroup discovery (cf. Section 2.1). Second, we describe common algorithmic search methods to solve this problem (cf. Section 2.2).

2.1 Problem of Subgroup Discovery

Context In general, subgroup discovery involves finding descriptions of interesting subsets of a dataset [5]. There are multiple options to define the type of dataset, the kind of subgroup description, and the criterion of interestingness. In the following, we formalize the notion of subgroup discovery that we tackle in this article. For broader surveys, see [5, 52, 53, 113].

Dataset We focus on tabular, real-valued data. In particular, $X \in \mathbb{R}^{m \times n}$ stands for a dataset in the form of a matrix. Each row is a data object, and each column is a feature. We assume that categorical features have been made numeric, e.g., via a one-hot or an ordinal encoding [84]. There are also subgroup-discovery methods that only process categorical data and require continuous features to be discretized [53, 86]. $X_{i\cdot} \in \mathbb{R}^n$ denotes the values of all features for the i-th data object, while $X_{\cdot j} \in \mathbb{R}^m$ denotes the values of the j-th feature for all data objects. $y \in Y^m$ represents the prediction target with domain Y, e.g., $Y = \{0, 1\}$ for binary classification or $Y = \mathbb{R}$ for regression. To harmonize formalization and evaluation, we focus on binary-classification scenarios in this article. In general, one may also conduct subgroup discovery in multi-class, multi-target, or regression scenarios [5].

Subgroup (description) A subgroup description typically comprises a conjunction of conditions on individual features [86]. For real-valued data, the conditions constitute intervals. Thus, a subgroup description defines a hyper-rectangle. In particular, the subgroup description comprises a lower and upper bound for each feature. The bounds for a feature may also be infinite to leave it unrestricted. A data object resides in the subgroup if all its feature values are in the intervals formed by lower and upper bounds:

Definition 1 (Subgroup (description))**.** Given a dataset $X \in \mathbb{R}^{m \times n}$, a *subgroup* is described by its lower bounds $lb \in \{\mathbb{R} \cup \{-\infty, +\infty\}\}^n$ and upper bounds $ub \in \{\mathbb{R} \cup \{-\infty, +\infty\}\}^n$. Data object $X_{i\cdot}$ is a *member* of this subgroup if $\forall j \in \{1, \ldots, n\} : \ (X_{ij} \geq lb_j) \wedge (X_{ij} \leq ub_j)$.

For categorical features, one may replace the inequality comparisons with equality comparisons against categorical feature values [5].

Throughout this article, we often use the terms *subgroup* and *subgroup description* interchangeably. In a more strict sense, one may use the former term to denote the subgroup's members and the latter for the subgroup's bounds [5].

Subgroup discovery Framing subgroup discovery as an optimization problem requires a notion of subgroup quality, i.e., interestingness of the subgroup. A function $Q(lb, ub, X, y)$ shall return the quality of a subgroup on a particular dataset. Without loss of generality, we assume a maximization problem:

Definition 2 (Subgroup discovery)**.** Given a dataset $X \in \mathbb{R}^{m \times n}$ with prediction target $y \in \{0, 1\}^m$, *subgroup discovery* is the problem of finding a subgroup (cf. Definition 1) with bounds $lb, ub \in \{\mathbb{R} \cup \{-\infty, +\infty\}\}^n$ that maximizes a given notion of subgroup quality $Q(lb, ub, X, y)$.

While this definition refers to one subgroup, some subgroup-discovery methods return a set of subgroups [5].

Subgroup quality For binary-classification scenarios, interesting subgroups should typically contain many data objects from one class but few from the other

class. While traditional classification tries to characterize the dataset globally, subgroup discovery follows a local paradigm, i.e., focuses on the data objects in the subgroup [86]. Without loss of generality, we assume that the class with label '1' is the class of interest, also called *positive* class. Weighted Relative Accuracy (WRAcc) [65] is a popular metric for subgroup quality [86]:

$$\text{WRAcc} = \frac{m_b}{m} \cdot \left(\frac{m_b^+}{m_b} - \frac{m^+}{m} \right) \tag{1}$$

Besides the total number of data objects m, this metric considers the number of positive data objects m^+, the number of data objects in the subgroup m_b, and the number of positive data objects in the subgroup m_b^+. In particular, WRAcc is the product of two factors: m_b/m expresses the generality of the subgroup as the relative frequency of subgroup membership. The second factor measures the relative accuracy of the subgroup, i.e., the difference in the relative frequency of the positive class between the subgroup and the whole dataset. If the subgroup contains the same fraction of positive data objects as the whole dataset, WRAcc is zero. The theoretical maximum and minimum of WRAcc depend on the class frequencies in the dataset. In particular, the maximum WRAcc for a dataset equals the product of the relative frequencies of positive and negative data objects in the dataset [83]:

$$\text{WRAcc}_{\max} = \frac{m^+}{m} \cdot \left(1 - \frac{m^+}{m} \right) \tag{2}$$

This maximum is reached if all positive data objects are in the subgroup and all negative data objects are outside, i.e., $m_b^+ = m_b = m^+$. Depending on the feature values of the dataset, a corresponding subgroup description may not exist. Further, the maximum value of this expression is 0.25 if both classes occur with equal frequency but becomes smaller the more imbalanced the classes are. Thus, it makes sense to normalize WRAcc when working with datasets with different class frequencies. One normalization, which we use in our experiments, is a max-normalization to the range $[-1, 1]$ [83]:

$$\text{nWRAcc} = \frac{\text{WRAcc}}{\text{WRAcc}_{\max}} = \frac{m_b^+ \cdot m - m^+ \cdot m_b}{m^+ \cdot (m - m^+)} \tag{3}$$

Alternatively, one can also min-max-normalize the range to $[0, 1]$ [25, 113].

2.2 Search Methods

To discover subgroups, there are heuristic search methods (cf. Section 2.2.1), like PRIM [38] and Best Interval [82], as well as exhaustive search methods (cf. Section 2.2.2), like SD-Map [6, 8], MergeSD [46], and BSD [71, 73]. In this section, we discuss five search methods that are relevant for our experiments; see [5, 52, 53, 113] for comprehensive surveys on subgroup-discovery methods.

Algorithm 1: *PRIM* for subgroup discovery.

Input: Dataset $X \in \mathbb{R}^{m \times n}$,
Prediction target $y \in \{0,1\}^m$,
Subgroup-quality function $Q(lb, ub, X, y)$,
Peeling fraction $\alpha \in (0,1)$,
Support threshold $\beta_0 \in [0,1]$

Output: Subgroup bounds $lb, ub \in \{\mathbb{R} \cup \{-\infty, +\infty\}\}^n$

for $j \leftarrow 1$ **to** n **do** `// Start with unrestricted subgroup`
  $(lb_j^{\text{opt}},\ ub_j^{\text{opt}}) \leftarrow (-\infty, +\infty)$
$Q^{\text{opt}} \leftarrow Q(lb^{\text{opt}}, ub^{\text{opt}}, X, y)$
$(lb^{\text{peel}}, ub^{\text{peel}}) \leftarrow (lb^{\text{opt}},\ ub^{\text{opt}})$
while $\frac{m_b}{m} > \beta_0$ **do** `// Support threshold satisfied`
  $Q^{\text{cand}} \leftarrow -\infty$
  for $j \in$ *get_permissible_feature_idxs(...)* **do**
    $(lb,\ ub) \leftarrow (lb^{\text{peel}},\ ub^{\text{peel}})$ `// Try peeling lower bound`
    $lb_j \leftarrow \text{quantile}(X_{\cdot j},\ lb,\ ub,\ \alpha)$
    if $Q(lb, ub, X, y) > Q^{cand}$ **then**
      $(lb^{\text{cand}}, ub^{\text{cand}}) \leftarrow (lb, ub)$
    $(lb,\ ub) \leftarrow (lb^{\text{peel}},\ ub^{\text{peel}})$ `// Try peeling upper bound`
    $ub_j \leftarrow \text{quantile}(X_{\cdot j},\ lb,\ ub,\ 1-\alpha)$
    if $Q(lb, ub, X, y) > Q^{cand}$ **then**
      $(lb^{\text{cand}}, ub^{\text{cand}}) \leftarrow (lb, ub)$
  $(lb^{\text{peel}}, ub^{\text{peel}}) \leftarrow (lb^{\text{cand}}, ub^{\text{cand}})$ `// Retain best candidate`
  if $Q(lb^{peel}, ub^{peel}, X, y) > Q^{opt}$ **then** `// Update optimum`
    $Q^{\text{opt}} \leftarrow Q(lb^{\text{peel}}, ub^{\text{peel}}, X, y)$
    $(lb^{\text{opt}},\ ub^{\text{opt}}) \leftarrow (lb^{\text{peel}},\ ub^{\text{peel}})$
$(lb, ub) \leftarrow (lb^{\text{opt}}, ub^{\text{opt}})$
for $j \leftarrow 1$ **to** n **do** `// Reset non-excluding bounds`
  if $lb_j = \min_{i \in \{1,\ldots,m\}} X_{ij}$ **then** $lb_j \leftarrow -\infty$
  if $ub_j = \max_{i \in \{1,\ldots,m\}} X_{ij}$ **then** $ub_j \leftarrow +\infty$
return lb, ub

2.2.1 Heuristic Search Methods

PRIM *Patient Rule Induction Method (PRIM)* [38] is an iterative search algorithm. In its basic form, it consists of a peeling phase and a pasting phase. Peeling restricts the bounds of the subgroup iteratively, while pasting expands them. Algorithm 1 outlines the peeling phase for finding one subgroup, which is the flavor of PRIM we consider in this article and denote as *PRIM*. Pasting may have little effect on the subgroup quality and is often left out [2]. Further, we do not discuss extensions of PRIM like bumping [38, 64], which uses bagging of multiple PRIM runs to improve subgroup quality, or covering [38], which returns a sequence of subgroups covering different data objects.

The algorithm *PRIM* starts with a subgroup containing all data objects, which is the initial solution candidate (Lines 1–4). It continues peeling until the current solution candidate contains at most a fraction β_0 of data objects (Line 5). The support threshold $\beta_0 \in [0, 1]$ is a user parameter. The returned subgroup is the optimal solution candidate over all peeling iterations (Line 20). In our *PRIM* implementation, we add a small post-processing step after peeling: We set *non-excluding bounds* to infinity (Lines 21–23). These are bounds that do not exclude any data objects from the subgroup, i.e., lower/upper bounds that equal the minimum/maximum feature value over all data objects. There are two reasons behind this post-processing: First, we ensure that these bounds remain non-excluding for any new data, where global feature minima/maxima may differ. Second, it becomes easier to see which features are selected in the subgroup description and which are not.

In the iterative peeling procedure (Lines 5–19), the algorithm generates new solution candidates by trying to restrict each *permissible feature* (Lines 7–15). In unconstrained subgroup discovery, each feature is permissible, but the function *get_permissible_feature_idxs(...)* will become useful once we introduce constraints. For each Feature j, the algorithm tests a new lower bound at the α-quantile of feature values in the subgroup and a new upper bound at the $1 - \alpha$-quantile of feature values in the subgroup. The peeling fraction $\alpha \in (0, 1)$ is a user parameter. It describes which fraction of data objects gets excluded from the subgroup in each peeling iteration. Having tested two new bounds for each feature, the algorithm takes the subgroup with the highest associated quality (Line 16) and continues peeling it in the next iteration. Further, if this solution candidate improves upon the optimal solution candidate from all prior iterations, it is stored as the new optimum (Lines 17–19).

Beam Search Beam search is a generic search strategy that is also common in subgroup discovery [9]. It maintains a set of currently best solution candidates, i.e., the beam, which it iteratively updates. The number of solution candidates in the beam is a user parameter, i.e., the beam width $w \in \mathbb{N}$. We outline one way to implement it in Algorithms 2 and 3, which we refer to as *Beam Search* in the following. It is an adapted version of the beam-search implementation in the Python package *pysubgroup* [72].

First, the algorithm *Beam Search* initializes the beam by creating w unre-

Algorithm 2: Generic beam search for subgroup discovery.

Input: Dataset $X \in \mathbb{R}^{m \times n}$,
Prediction target $y \in \{0,1\}^m$,
Subgroup-quality function $Q(lb, ub, X, y)$,
Beam width $w \in \mathbb{N}$

Output: Subgroup bounds $lb, ub \in \{\mathbb{R} \cup \{-\infty, +\infty\}\}^n$

for $l \leftarrow 1$ **to** w **do** `// Initialize beam`
  for $j \leftarrow 1$ **to** n **do**
    $(lb_j^{(\text{beam},\ l)},\ ub_j^{(\text{beam},\ l)}) \leftarrow (-\infty, +\infty)$ `// Unrestricted`
  $cand\_has\_changed^{(l)} \leftarrow$ **true** `// Subgroup should be updated`
  $Q^{(l)} \leftarrow Q(lb^{(\text{beam},\ l)}, ub^{(\text{beam},\ l)}, X, y)$
while $\left(\sum_{l=1}^{w} cand\_has\_changed^{(l)}\right) > 0$ **do** `// Beam has changed`
  $prev\_cand\_changed\_idxs \leftarrow \{l \mid cand\_has\_changed^{(l)}\}$
  for $l \leftarrow 1$ **to** w **do** `// Create temporary solution candidates`
    $(lb^{(\text{cand},\ l)},\ ub^{(\text{cand},\ l)}) \leftarrow (lb^{(\text{beam},\ l)},\ ub^{(\text{beam},\ l)})$
    $cand\_has\_changed^{(l)} \leftarrow$ **false**
  for $l \in prev\_cand\_changed\_idxs$ **do** `// Prepare beam updates`
    for $j \in$ *get_permissible_feature_idxs(...)* **do**
      evaluate_subgroup_updates(...) `// Algorithm 3 or 4`
  for $l \leftarrow 1$ **to** w **do** `// Update beam`
    $(lb^{(\text{beam},\ l)},\ ub^{(\text{beam},\ l)}) \leftarrow (lb^{(\text{cand},\ l)},\ ub^{(\text{cand},\ l)})$
$l \leftarrow \arg\max_{l \in \{1,\ldots,w\}} Q^{(l)}$ `// Select best subgroup from beam`
$(lb,\ ub) \leftarrow (lb^{(\text{beam},\ l)},\ ub^{(\text{beam},\ l)})$
for $j \leftarrow 1$ **to** n **do** `// Reset non-excluding bounds`
  if $lb_j = \min_{i \in \{1,\ldots,m\}} X_{ij}$ **then** $lb_j \leftarrow -\infty$
  if $ub_j = \max_{i \in \{1,\ldots,m\}} X_{ij}$ **then** $ub_j \leftarrow +\infty$
return lb, ub

Algorithm 3: evaluate_subgroup_updates(...) for *Beam Search*.

Input: Parameters and variables from Algorithm 2
Output: None; modifies variables from Algorithm 2 in-place

```
1  (lb, ub) ← (lb^(beam, l), lb^(beam, l))        // Next, update lower bound
2  for b ∈ sort(unique(get_feature_values(X, j, lb^(beam, l), ub^(beam, l)))) do
3      lb_j ← b
4      if (Q(lb, ub, X, y) > min_{l∈{1,...,w}} Q^(l)) and
          (lb, ub) ∉ {(lb^(cand, l), ub^(cand, l)) | l ∈ {1, ..., w}} then
5          l ← arg min_{l∈{1,...,w}} Q^(l)          // Replace worst candidate
6          (lb^(cand, l), ub^(cand, l)) ← (lb, ub)
7          cand_has_changed^(l) ← true
8          Q^(l) ← Q(lb, ub, X, y)
9  (lb, ub) ← (lb^(beam, l), lb^(beam, l))        // Next, update upper bound
10 for b ∈ sort(unique(get_feature_values(X, j, lb^(beam, l), ub^(beam, l)))) do
11     ub_j ← b
12     if (Q(lb, ub, X, y) > min_{l∈{1,...,w}} Q^(l)) and
          (lb, ub) ∉ {(lb^(cand, l), ub^(cand, l)) | l ∈ {1, ..., w}} then
13         l ← arg min_{l∈{1,...,w}} Q^(l)          // Replace worst candidate
14         (lb^(cand, l), ub^(cand, l)) ← (lb, ub)
15         cand_has_changed^(l) ← true
16         Q^(l) ← Q(lb, ub, X, y)
```

stricted subgroups (Lines 1–5). Further, it stores the quality of each of these subgroups. Additionally, it records which subgroups changed in the previous iteration (Lines 6–15) of the search. In particular, it stops once all subgroups in the beam remain unchanged (Line 6). Subsequently, it returns the best subgroup from the beam (Lines 16–21). As for Algorithm 1, we replace all non-excluding bounds with infinity as a post-processing step.

The main loop (Lines 6–15) updates the beam. In particular, for each subgroup that changed in the previous iteration, the algorithm creates new solution candidates by attempting to update the bounds of each feature separately (Lines 11–13). There are different options for this update step. Algorithm 3 outlines the update procedure for *Beam Search*, while *Best Interval* uses a slightly different one (cf. Algorithm 4). For *Beam Search*, the procedure tries to refine the lower bound (Lines 1–8) and the upper bound (Lines 9–16) for a given Feature j separately by replacing it with another feature value from data objects in the subgroup. In particular, it iterates over all these unique feature values. Each solution candidate that improves upon the minimum subgroup quality from the beam replaces the corresponding subgroup, unless it already is part of the beam due to another update action (Lines 4–8 and 12–16).

Algorithm 4: evaluate_subgroup_updates(...) for *Best Interval*.

Input: Parameters and variables from Algorithm 2
Output: None; modifies variables from Algorithm 2 in-place

```
1  (lb, ub) ← (lb^(beam, l), lb^(beam, l))  // Value at index j will change
2  (lb^opt, ub^opt) ← (lb^(beam, l), lb^(beam, l))          // (l, r) in [82]
3  Q^opt ← Q(lb^opt, ub^opt, X, y)                  // WRAcc_max in [82]
4  Q^temp ← −∞                                               // h_max in [82]
5  lb_j^temp ← −∞                                            // t_max in [82]
6  for b ∈ sort(unique(get_feature_values(X, j, lb^(beam, l), ub^(beam, l)))) do
7      lb_j ← b
8      ub_j ← ub_j^(beam, l)
9      if Q(lb, ub, X, y) > Q^temp then
10         lb_j^temp ← b
11         Q^temp ← Q(lb, ub, X, y)
12     lb_j ← lb_j^temp
13     ub_j ← b
14     if Q(lb, ub, X, y) > Q^opt then
15         (lb^opt, ub^opt) ← (lb, ub)
16         Q^opt ← Q(lb, ub, X, y)
17 if (Q^opt > min_{l∈{1,...,w}} Q^(l)) and
      (lb^opt, ub^opt) ∉ {(lb^(cand, l), ub^(cand, l)) | l ∈ {1, ..., w}} then
18     l ← arg min_{l∈{1,...,w}} Q^(l)              // Replace worst candidate
19     (lb^(cand, l), ub^(cand, l)) ← (lb^opt, ub^opt)
20     cand_has_changed^(l) ← true
21     Q^(l) ← Q^opt
```

Best Interval *Best Interval* [82] offers an update procedure for subgroups (cf. Algorithm 4) that is tailored towards WRAcc (cf. Equation 1) as the subgroup-quality function. This update procedure can be used within a generic beam-search strategy (cf. Algorithm 2). As before, the best new solution candidate from an update step becomes part of the beam if it improves upon the worst subgroup quality there and is not a duplicate (Lines 17–21).

However, solution candidates are generated differently than in the update procedure of *Beam Search* (cf. Algorithm 3). In particular, *Best Interval* updates lower and upper bounds for a given Feature j simultaneously rather than separately (Lines 1–16). Thus, this procedure optimizes over all potential combinations of lower and upper bounds. However, it still only requires one pass over the unique values of Feature j rather than quadratic cost, due to theoretical properties of the WRAcc function [82].

Algorithm 5: *MORS* for subgroup discovery.

Input: Dataset $X \in \mathbb{R}^{m \times n}$,
Prediction target $y \in \{0,1\}^m$
Output: Subgroup bounds $lb, ub \in \{\mathbb{R} \cup \{-\infty, +\infty\}\}^n$

for $j \leftarrow 1$ **to** n **do**
  $lb_j \leftarrow \min_{\substack{i \in \{1,\dots,m\} \\ y_i = 1}} X_{ij}$
  $ub_j \leftarrow \max_{\substack{i \in \{1,\dots,m\} \\ y_i = 1}} X_{ij}$
  if $lb_j = \min_{i \in \{1,\dots,m\}} X_{ij}$ **then** $lb_j \leftarrow -\infty$
  if $ub_j = \max_{i \in \{1,\dots,m\}} X_{ij}$ **then** $ub_j \leftarrow +\infty$
for $j \notin$ *get_permissible_feature_idxs(...)* **do**
  $(lb_j,\ ub_j) \leftarrow (-\infty, +\infty)$
return lb, ub

2.2.2 Exhaustive Search Methods

SD-Map *SD-Map* [8] is an exhaustive search method based on the pattern-mining algorithm *FP-growth* [51]. It assumes discretized features and produces subgroup descriptions with equality conditions of the form $X_{ij} = v$ instead of the numeric intervals we focus on (Definition 1). *SD-Map** [6] adds support for numeric targets and quality-based pruning of the search space.

Bitset-based Subgroup Discovery (BSD) *BSD* [73] is another exhaustive search method that produces equality conditions in the subgroup description. It combines a branch-and-bound strategy with a special binary data representation and pruning techniques to speed up the search. *NumBSD* [71] can handle numeric targets, though the features are still assumed to be discrete.

3 Baselines

In this section, we propose and analyze two baselines for subgroup discovery, *MORS* (cf. Section 3.1) and *Random Search* (cf. Section 3.2). They are conceptually simpler than the heuristic search methods (cf. Section 2.2.1) and serve as further reference points in our experiments. While they technically also are heuristics, we use the term *baselines* to refer to these two methods specifically.

3.1 MORS

This baseline builds on the following definition:

Definition 3 (Minimal Optimal-Recall Subgroup (MORS))**.** Given a dataset $X \in \mathbb{R}^{m \times n}$ with prediction target $y \in \{0,1\}^m$, the *Minimal Optimal-Recall*

Subgroup (MORS) is the subgroup (cf. Definition 1) whose lower and upper bounds of each feature correspond to the minimum and maximum value of that feature over all positive data objects (i.e., with $y_i = 1$) from the dataset X.

The definition ensures that all positive data objects are contained in the subgroup. Thus, the evaluation metric *recall*, i.e., the fraction of positive data objects becoming subgroup members, reaches its *optimal* value of 1. At the same time, raising the lower bounds or lowering the upper bounds would exclude positive data objects from the subgroup. In this sense, the set of subgroup members is *minimal*. The corresponding subgroup description is unique and implicitly solves the following variant of the subgroup-discovery problem:

Definition 4 (Minimal-optimal-recall-subgroup discovery)**.** Given a dataset $X \in \mathbb{R}^{m \times n}$ with prediction target $y \in \{0,1\}^m$, *minimal-optimal-recall-subgroup discovery* is the problem of finding a subgroup (cf. Definition 1) that contains as few negative data objects (i.e., with $y_i = 0$) as possible but all positive data objects (i.e., with $y_i = 1$) from the dataset X.

I.e., the problem targets at minimizing the number of false positives subject to producing no false negatives. With the constraint on the positive data objects, this problem is equivalent to maximizing the number of true negatives, i.e., negative data objects excluded from the subgroup.

Algorithm 5 outlines the procedure to determine the *MORS* bounds. Slightly deviating from Definition 3, but consistent to Algorithm 1, *MORS* replaces all non-excluding bounds with infinity (Lines 4–5). Further, if only certain features are permissible to be bounded, as we discuss later, we reset the bounds of the remaining features (Lines 6–7).

Since *MORS* only needs to iterate over all data objects and features once to determine the minima and maxima, the time complexity of this algorithm is $O(m{\cdot}n)$. This places minimal-optimal-recall-subgroup discovery in complexity class $\mathcal{P}$:

Proposition 1 (Complexity of minimal-optimal-recall-subgroup discovery)**.** *The problem of minimal-optimal-recall-subgroup discovery (cf. Definition 4) can be solved in* $O(m \cdot n)$.

For complexity proofs later in our article, we define another variant of the subgroup-discovery problem based on another particular type of subgroups [85]:

Definition 5 (Perfect subgroup)**.** Given a dataset $X \in \mathbb{R}^{m \times n}$ with prediction target $y \in \{0,1\}^m$, a *perfect subgroup* is a subgroup (cf. Definition 1) that contains all positive data objects (i.e., with $y_i = 1$) but no negative data objects (i.e., with $y_i = 0$) from the dataset X.

Perfect subgroups reach the theoretical maximum WRAcc for a dataset (cf. Equation 2). Next, we define a corresponding search problem:

Definition 6 (Perfect-subgroup discovery)**.** Given a dataset $X \in \mathbb{R}^{m \times n}$ with prediction target $y \in \{0,1\}^m$, *perfect-subgroup discovery* is the problem of finding a perfect subgroup (cf. Definition 5) if it exists or determining that it does not exist.

Algorithm 6: *Random Search* for subgroup discovery.

Input: Dataset $X \in \mathbb{R}^{m \times n}$,
Prediction target $y \in \{0,1\}^m$,
Subgroup-quality function $Q(lb, ub, X, y)$,
Number of iterations $n\_iters \in \mathbb{N}$
Output: Subgroup bounds $lb, ub \in \{\mathbb{R} \cup \{-\infty, +\infty\}\}^n$

1 $Q^{\text{opt}} \leftarrow -\infty$
2 **for** *iters* $\leftarrow 1$ **to** n_iters **do**
3 **for** $j \leftarrow 1$ **to** n **do**
4 $(lb_j,\ ub_j) \leftarrow (-\infty, +\infty)$
5 **for** $j \in$ *get_permissible_feature_idxs(...)* **do**
6 $(lb_j,\ ub_j) \leftarrow$ sample_uniformly(unique($X_{\cdot j}$))
7 **if** $Q(lb, ub, X, y) > Q^{opt}$ **then**
8 $Q^{\text{opt}} \leftarrow Q(lb, ub, X, y)$
9 $(lb^{\text{opt}},\ ub^{\text{opt}}) \leftarrow (lb,\ ub)$
10 **for** $j \leftarrow 1$ **to** n **do**
11 **if** $lb_j^{opt} = \min_{i \in \{1,\ldots,m\}} X_{ij}$ **then** $lb_j^{\text{opt}} \leftarrow -\infty$
12 **if** $ub_j^{opt} = \max_{i \in \{1,\ldots,m\}} X_{ij}$ **then** $ub_j^{\text{opt}} \leftarrow +\infty$
13 **return** $lb^{opt},\ ub^{opt}$

Since the algorithm *MORS* helps solving this problem in $O(m \cdot n)$, we obtain the following complexity result:

Proposition 2 (Complexity of perfect-subgroup discovery)**.** *The problem of perfect-subgroup discovery (cf. Definition 6) can be solved in* $O(m \cdot n)$.

In particular, after *MORS* (cf. Algorithm 5) has found a subgroup, one only needs to check whether the subgroup contains any negative data objects. If the found subgroup does not contain negative data objects, then it is perfect. If it does, then no perfect subgroup exists. In particular, the bounds found by *MORS* cannot be made tighter to exclude negative data objects from the subgroup without also excluding positive data objects, thereby violating perfection.

3.2 Random Search

Algorithm 6 outlines a randomized search procedure that constitutes the second baseline. *Random Search* generates and evaluates subgroups for a fixed number of iterations, which the user controls with the parameter $n\_iters \in \mathbb{N}$. Hereby, subgroup generation samples a lower bound and an upper bound uniformly random from the unique values for each permissible feature, leaving the remaining features unrestricted (Lines 3–6). The algorithm tracks the best generated subgroup so far over the iterations (Lines 7–9) and finally returns the

subgroup with the highest quality. Like Algorithm 1, *Random Search* replaces all non-excluding bounds with infinity (Lines 10–12).

4 Constrained Subgroup Discovery

In this section, we discuss subgroup discovery with constraints. First, we frame subgroup discovery as an SMT optimization problem (cf. Section 4.1). Second, we give a brief overview of potential constraint types (cf. Section 4.2). Third, we formalize and analyze feature-cardinality constraints (cf. Section 4.3). Fourth, we formalize and analyze alternative subgroup descriptions (cf. Section 4.4).

4.1 SMT Encoding of Subgroup Discovery

To find optimal subgroups exactly, one can encode subgroup discovery as a white-box optimization problem and employ a solver. Here, we propose a Satisfiability Modulo Theories (SMT) [18, 95] encoding, which is straightforward given the problem definition (cf. Definition 2). SMT allows expressions in first-order logic with particular interpretations, e.g., arrays, arithmetic, or bit vectors [18]. Our encoding of subgroup discovery uses linear real arithmetic (LRA). Complementing this SMT encoding, Appendix A.1 describes further encodings: SMT for categorical features (cf. Section A.1.1), mixed integer linear programming (cf. Section A.1.2), and maximum satisfiability (cf. Section A.1.3).

The optimization problem consists of an objective function and constraints.

Objective function We use WRAcc as the objective function, which should be maximized. In the formula for WRAcc (cf. Equation 1), m and m^+ are constants, while m_b and m_b^+ depend on the decision variables. The previously provided formula seems to be non-linear in the decision variables since m_b appears in the numerator and denominator. However, one can reformulate the expression by multiplying its two factors, obtaining the following expression:

$$\text{WRAcc} = \frac{m_b^+}{m} - \frac{m_b \cdot m^+}{m^2} = \frac{m_b^+ \cdot m - m_b \cdot m^+}{m^2} \tag{4}$$

In this new expression, the denominators are constant, and the factor m^+ in the numerator is constant as well. Thus, the whole expression is linear in m_b^+ and m_b. We define these two quantities as linear expressions from binary decision variables $b \in \{0,1\}^m$ that denote subgroup membership. I.e., b_i expresses whether the i-th data object is in the subgroup or not:

$$\begin{aligned} m_b &:= \sum_{i=1}^{m} b_i \\ m_b^+ &:= \sum_{\substack{i \in \{1,\dots,m\} \\ y_i = 1}} b_i \end{aligned} \tag{5}$$

Since the values of the target variable y are fixed, the expression for m_b^+ only sums over the positive data objects. Further, one may define m_b^+ and m_b as separate integer variables or directly insert their expressions into Equation 4. We chose the latter formulation in our implementation and therefore wrote $:=$ in Equation 5 instead of using a proper propositional operator like $\leftrightarrow$.

The formula for nWRAcc (cf. Equation 3) is linear as well, having the same enumerator as Equation 4 and a different constant in the denominator.

Constraints The subgroup membership b_i of a data object depends on the bounds of the subgroup (cf. Definition 1). Thus, we define real-valued decision variables $lb, ub \in \{\mathbb{R} \cup \{-\infty, +\infty\}\}^n$ for the latter. In particular, there is one lower bound and one upper bound for each of the n features. The upper bounds naturally need to be at least as high as the lower bounds:

$$\forall j \in \{1, \ldots, n\} : \; lb_j \leq ub_j \tag{6}$$

A data object is a member of the subgroup if all its feature values are contained within the bounds:

$$\forall i \in \{1, \ldots, m\} : \; b_i \leftrightarrow \bigwedge_{j \in \{1,\ldots,n\}} ((X_{ij} \geq lb_j) \wedge (X_{ij} \leq ub_j)) \tag{7}$$

Instead of defining separate decision variables b_i and binding them to the bounds with an equivalence constraint, one could also insert the Boolean expression into the right-hand-side of Equation 5 directly. In particular, lb_j and ub_j are the only decision variables strictly necessary for the optimization problem. However, for formulating some constraint types on subgroups (cf. Section 4.2), it is helpful to be able to refer to b_i.

Complete optimization problem Combining all prior definitions of decision variables, constraints, and the objective function, we obtain the following SMT optimization problem:

$$\begin{aligned}
\max \quad & Q_{\text{WRAcc}} = \frac{m_b^+}{m} - \frac{m_b \cdot m^+}{m^2} \\
\text{s.t.:} \quad & m_b := \sum_{i=1}^{m} b_i \\
& m_b^+ := \sum_{\substack{i \in \{1,\ldots,m\} \\ y_i = 1}} b_i \\
\forall i \in \{1, \ldots, m\} \quad & b_i \leftrightarrow \bigwedge_{j \in \{1,\ldots,n\}} ((X_{ij} \geq lb_j) \wedge (X_{ij} \leq ub_j)) \\
\forall j \in \{1, \ldots, n\} \quad & lb_j \leq ub_j \\
& b \in \{0,1\}^m \\
& lb, ub \in \{\mathbb{R} \cup \{-\infty, +\infty\}\}^n
\end{aligned} \tag{8}$$

We call this optimization problem *unconstrained subgroup discovery* in the following since it only contains constraints that are technically necessary to define subgroup discovery properly but no additional constraints (cf. Section 4.2).

Post-processing In our implementation, we add a small post-processing step. In particular, we do not use the solver-determined values of the variables lb_j and ub_j when evaluating subgroup quality. Instead, we set the lower and upper bounds to the minimum and maximum feature values of all data objects in the subgroup (i.e., with $b_i = 1$). Thus, we ensure that the bounds correspond to actual feature values. This guarantee is not formally necessary but consistent with the subgroup descriptions returned by heuristic search methods and baselines. Also, we avoid potential minor numerical issues caused by extracting the values of real variables from the solver. Finally, if the subgroup does not contain any data objects, we use invalid bounds (i.e., $ub_j = -\infty < \infty = lb_j$) to ensure that the subgroup remains empty even for arbitrary new data objects.

4.2 Overview of Constraint Types

A white-box formulation of subgroup discovery, like our SMT encoding, supports directly integrating a variety of constraint types in a declarative manner. In contrast, algorithmic search methods for subgroups need to explicitly check constraints or implicitly ensure that generated solution candidates satisfy constraints. Such implicit guarantees may only hold for particular constraint types or may require adapting the search method accordingly.

Domain knowledge Constraints can express firm knowledge or hypotheses from the domain that solutions of machine-learning techniques should adhere to [16]. Since the definition of such constraints depends on the use case, we do not give domain-specific examples here. [7, 9, 10] provide a taxonomy and examples for knowledge-based constraint types in subgroup discovery. In a white-box formulation of subgroup discovery, such constraints may restrict the values of decision variables, e.g., lower and upper bounds, subgroup membership (cf. Equation 8), or selected features (cf. Equation 9). For example, certain bound values or feature combinations used in the subgroup description may contradict domain knowledge and should therefore be prevented with constraints. In our formulation of subgroup discovery as an SMT problem with linear real arithmetic (cf. Section 4.1), one can employ propositional logic, basic arithmetic operators, and inequalities to express constraints over the decision variables [18].

Secondary objectives Various notions of subgroup quality can serve as an objective in subgroup discovery [5, 53]. If one wants to consider several quality metrics simultaneously, one option is multi-objective optimization. However, the latter typically requires using different search methods than single-objective optimization. Also, there may be not one but a set of Pareto-optimal solutions,

or users may need to define trade-offs between objectives manually. Alternatively, one can keep a single primary objective and add the other objectives as inequality constraints, e.g., enforcing that their values are below or above user-defined thresholds. According to [86], such lower bounds on subgroup quality are a common constraint type in subgroup discovery. Without constraints, one may prune the set of discovered subgroups as a post-processing step [5]. Finally, quality-based pruning can also reduce the search space during subgroup discovery, e.g., using optimistic estimates in exhaustive search [5, 6, 46]. However, such automatically determined bounds on subgroup quality relate to the primary optimization objective rather than being user-provided constraints.

Regularization Regularization aims to control the complexity of machine-learning models, preventing overfitting and increasing interpretability. In subgroup discovery, we see three directions for regularization: the data objects via subgroup membership, the feature values via the subgroup's bounds, and the features via the subgroup's feature selection.

Regarding *subgroup membership*, one can introduce lower or upper bounds on the number of data objects in the subgroup, using the decision variables b_i (cf. Equation 7). Such constraints are particularly useful for notions of subgroup quality that incentivize including all data objects in the subgroup, like recall, or including very few data objects, like precision. In contrast, WRAcc (cf. Equation 1) automatically considers the number of data objects in the subgroup. According to [86], lower bounds on subgroup membership are a common constraint type in subgroup discovery.

Regarding *bounds*, one can define minimum or maximum values for the range of a feature in the subgroup, i.e., the difference between lower and upper bound, using the decision variables lb_j and ub_j. Such constraints can prevent choosing ranges that are too small or too large for user needs. If the features are normalized, one can also constrain the volume of the subgroup, i.e., the product of all ranges, or the density, i.e., the number of data objects per volume. Generally, however, a feature's bounded value range need not indicate how many data objects are excluded from the subgroup. Alternatively, one can constrain the subgroup membership implied by individual features' bounds, e.g., enforcing that selected features exclude at least a certain fraction of data objects. The latter constraint type may prevent setting oversensitive bounds that only exclude few data objects and do not generalize to unseen data.

Regarding *feature selection*, one can limit the number of features used in the subgroup description, which is a common constraint type [86] and also a metric for subgroup complexity [52, 53, 113], already proposed in the article introducing PRIM [38]. Section 4.3 discusses such feature-cardinality constraints in detail. Instead of using this constraint type, one may also post-process subgroups to eliminate irrelevant features after search [38]. Further, instead of limiting the total number of used features, one may also introduce constraints to remove individual irrelevant features based on the number of data objects they represent correctly in absolute terms or relative to other features [66].

Alternatives As for regularization, constraints for alternatives may relate to data objects, features, or feature values. We see two major notions of alternatives, which we call *alternative subgroups* and *alternative subgroup descriptions*.

Alternative subgroups aim to contain different sets of data objects. Since subgroups only intend to cover specific regions of the data, it is natural to search for multiple subgroups to cover multiple regions; see [5] for an overview of subgroup-set selection. One can search for multiple subgroups sequentially or simultaneously. The 'covering' approach allows sequential search for any subgroup-discovery method by removing all data objects contained in previous subgroups and repeating subgroup discovery [38]. Alternatively, one may reweigh [42, 67] or resample [107] data objects based on their subgroup membership. In contrast, simultaneous search requires subgroup-discovery methods that specifically target at multiple solutions [69, 70, 73, 81, 101]. E.g., a heuristic search may retain multiple solution candidates at each step. The notion of subgroup quality typically becomes broader: Besides measures for predictive quality like WRAcc, the diversity of subgroups [19, 69, 70, 81], e.g., to which extent they contain different data objects, and their number [52, 53, 113] may also serve as metrics. One may also filter redundant subgroups as a post-processing step [24, 45, 54, 70]. In a white-box formulation of subgroup discovery, one can enforce diversity with appropriate constraints on subgroup membership (cf. Equation 7), e.g., limiting the number of data objects that can be members of two subgroups simultaneously.

In contrast, *alternative subgroup descriptions* explicitly aim to contain a similar set of data objects but use different subgroup descriptions, e.g., a different feature selection. Section 4.4 discusses this constraint type in detail. Related to this concept, [22] introduces the notion of equivalent subgroup descriptions of minimal length, which cover exactly the same set of data objects.

4.3 Feature-Cardinality Constraints

In this section, we discuss feature-cardinality constraints for subgroup discovery. First, we motivate and formalize them (cf. Section 4.3.1). Next, we describe how to integrate them into our SMT encoding of subgroup discovery (cf. Section 4.3.2), heuristic search methods (cf. Section 4.3.3), and baselines (cf. Section 4.3.4). Finally, we analyze the time complexity of subgroup discovery with this constraint type (cf. Section 4.3.5).

4.3.1 Concept

Feature-cardinality constraints are a constraint type that regularizes subgroup descriptions (cf. Section 4.2). In particular, this constraint type limits the number of features used in the subgroup description, rendering the latter less complex and easier to interpret [86]. To formalize this constraint type, we define feature selection [50, 76] in the context of subgroup discovery as follows:

Definition 7 (Feature selection in subgroups)**.** Given a dataset $X \in \mathbb{R}^{m \times n}$ and a subgroup (cf. Definition 1) with bounds $lb, ub \in \{\mathbb{R} \cup \{-\infty, +\infty\}\}^n$, Feature j

is *selected* if the bounds exclude at least one data object of X from the subgroup, i.e., $\exists i \in \{1, \ldots, m\} : \ (X_{ij} < lb_j) \vee (X_{ij} > ub_j)$.

The bounds of unselected features can be considered infinite, effectively removing these features from the subgroup description. The *feature cardinality* of the subgroup is the number of selected features. Related work also uses the terms *depth* [86] or *length* [5, 52], though partly referring to the number of conditions in the subgroup description rather than selected features. I.e., if there is a lower and an upper bound for a feature, some related work counts this feature twice instead of once.

To formulate a feature-cardinality constraint, users provide an upper bound on the number of selected features:

Definition 8 (Feature-cardinality constraint)**.** Given a cardinality threshold $k \in \mathbb{N}$, a *feature-cardinality constraint* for a subgroup (cf. Definition 1) requires the subgroup to have at most k features selected (cf. Definition 7).

In practice, less than k features may be selected if selecting more features does not improve the subgroup quality.

4.3.2 SMT Encoding

We first need to encode whether a feature is selected or not. Thus, we introduce binary decision variables $s, s^{\text{lb}}, s^{\text{ub}} \in \{0,1\}^n$. A feature is selected if its bounds exclude at least one data object from the subgroup (cf. Definition 7), i.e., the lower bound is higher than the minimum feature value or the upper bound is lower than the maximum feature value:

$$\begin{aligned} \forall j: \quad & s_j^{\text{lb}} \leftrightarrow \left(lb_j > \min_{i \in \{1,\ldots,m\}} X_{ij} \right) \\ \forall j: \quad & s_j^{\text{ub}} \leftrightarrow \left(ub_j < \max_{i \in \{1,\ldots,m\}} X_{ij} \right) \\ \forall j: \quad & s_j \leftrightarrow \left(s_j^{\text{lb}} \vee s_j^{\text{ub}} \right) \\ \text{with index:} \quad & j \in \{1, \ldots, n\} \end{aligned} \tag{9}$$

In this equation, minimum and maximum feature values are constants that can be determined before formulating the optimization problem.

Given the definition of s_j, setting an upper bound on the number of selected features (cf. Definition 8) is straightforward:

$$\sum_{j=1}^{n} s_j \leq k \tag{10}$$

Instead of explicitly defining the decision variables s_j, s_j^{lb}, and s_j^{ub}, one could also insert the corresponding expressions into Equation 10 directly. However, we will also use s_j for alternative subgroup descriptions (cf. Section 4.4.2), so we define corresponding variables in our implementation.

The overall SMT encoding of subgroup discovery with a feature-cardinality constraint is the SMT encoding of unconstrained subgroup discovery (cf. Equation 8) supplemented by the variables and constraints from Equations 9 and 10.

In our implementation, we also add a post-processing step that sets non-excluding lower bounds (i.e., with $s_j^{\text{lb}} = 0$) to $-\infty$ and non-excluding upper bounds (i.e., with $s_j^{\text{ub}} = 0$) to $+\infty$. This step is consistent with the heuristic search methods and baselines (e.g., cf. Lines 21–23 in Algorithm 1).

4.3.3 Integration into Heuristic Search Methods

The feature-cardinality constraint (cf. Equation 10) has the form $|F_s| \leq k$ for the feature set F_s induced by the selection decisions $s \in \{0,1\}^n$. Thus, the constraint is *antimonotonic* [94] regarding the set of selected features. In particular, the empty feature set satisfies the constraint for any $k \geq 0$. If a selected feature set satisfies the constraint, all its subsets also satisfy it. Vice versa, if a feature set violates the constraint, all its supersets violate it as well.

This property allows to easily integrate the constraint into the three heuristic search methods from Section 2.2.1, i.e., *PRIM* (cf. Algorithm 1), *Beam Search* (cf. Algorithms 2 and 3), and *Best Interval* (cf. Algorithms 2 and 4), which all iteratively enlarge the selected feature set. In particular, they start with unrestricted subgroup bounds, i.e., an empty feature set. Each iteration may either add bounds on one further feature, thereby increasing the feature-set size by one, or refine the bounds on an already selected feature, so the feature-set size remains constant. Features cannot be deselected, so the feature-set size is non-decreasing overall. Thus, one can prevent generation of invalid solution candidates by defining the function *get_permissible_feature_idxs(...)* (cf. Line 7 in Algorithm 1 and Line 12 in Algorithm 2) as follows: If already k features are selected, only these features are permissible, i.e., only their bounds may be refined. If less than k features are selected, all features are permissible to be bounded, as in the unconstrained search.

4.3.4 Integration into Baselines

MORS *MORS* calls the function *get_permissible_feature_idxs(...)* in Line 6 of Algorithm 5. To instantiate this function, we employ a univariate, quality-based heuristic for feature selection: For each feature, we evaluate what would happen if only this feature was restricted according to *MORS* (cf. Definition 3). In particular, we determine the number of false positives, i.e., negative data objects in the subgroup, defined by each feature's *MORS* bounds (Lines 2–3). We select the k features with the lowest number of false positives.

This heuristic is equivalent to selecting the features with the highest WRAcc for univariate *MORS* bounds: Due to *MORS*, not only m and m^+ are constant in Equation 1 but also the number of positive data objects in the subgroup m_b^+, which equals m^+. In particular, one can rephrase Equation 1 as follows:

$$\text{WRAcc}_{\text{MORS}} = \frac{m_b^+}{m} - \frac{m_b \cdot m^+}{m^2} = \frac{m^+}{m} - \frac{m_b \cdot m^+}{m^2} = \frac{m^+}{m} \cdot \left(1 - \frac{m_b}{m}\right) \quad (11)$$

Thus, maximizing WRAcc corresponds to minimizing m_b/m, i.e., the relative frequency of the data objects in the subgroup. Since the number of positive data objects in the subgroup is fixed, this objective amounts to including as few negative data objects as possible in the subgroup, i.e., minimizing the number of false positives, which is what our univariate heuristic does as well.

The proposed heuristic only entails a linear runtime in the number of features, like the unconstrained *MORS*, since it evaluates each feature independently. With quadratic runtime, one can also consider interactions between features and thereby potentially increase subgroup quality. In particular, one could select features sequentially instead of simultaneously. In each iteration, one would select the feature whose *MORS* bounds, combined with the *MORS* bounds of all features selected in previous iterations, yield the lowest number of false positives. This sequential procedure mimics an existing greedy heuristic for the Maximum Coverage problem [27] (cf. Sections A.2.1 and A.2.2).

Random Search *Random Search* calls *get_permissible_feature_idxs(...)* in Line 5 of Algorithm 6). To observe a feature-cardinality threshold k, we simply sample k out of n features uniformly random without replacement. The bounds for these features will be restricted in the next step of the algorithm, while all remaining features remain unrestricted.

4.3.5 Time Complexity

We analyze three aspects of time complexity: the size of the search space for exhaustive search, parameterized complexity, and $\mathcal{NP}$-hardness.

Exhaustive search Before addressing feature-cardinality constraints, we analyze the unconstrained case. In general, the search space of subgroup discovery depends on the number of relevant candidate values for lower and upper bounds. With m data objects, each real-valued feature may have up to m unique values. It suffices to treat these unique values as bound candidates since any bounds between feature values or outside the feature's range do not change the subgroup membership during optimization, though the prediction on a test set with further data objects may vary. Thus, there are $O(m^2)$ relevant lower-upper-bound combinations per feature. Since we need to combine bounds over all n features, the size of the search space is $O(m^{2n})$:

Proposition 3 (Complexity of exhaustive search for subgroup discovery)**.** *An exhaustive search for subgroup discovery (cf. Definition 2) needs to evaluate* $O(m^{2n})$ *subgroups.*

For each of these candidate subgroups, the cost of evaluating a quality function like WRAcc (cf. Equation 1) typically is $O(m \cdot n)$ since it requires a constant number of passes over the dataset. Additionally, the number of potential subgroups should be seen as an upper bound: More efficient exhaustive search methods employ quality-based pruning to not explicitly evaluate all solution candidates while still implicitly covering the full search space [5].

Next, we adapt the result from Proposition 3 to feature-cardinality constraints. Instead of combining bounds from all n features, there are $\binom{n}{k} \leq n^k$ feature sets of size k with $O(m^{2k})$ bound candidates each:

Proposition 4 (Complexity of exhaustive search for subgroup discovery with feature-cardinality constraint)**.** *An exhaustive search for subgroup discovery (cf. Definition 2) with a feature-cardinality constraint (cf. Definition 8) needs to evaluate $O(n^k \cdot m^{2k})$ subgroups.*

For the special case $k = 1$, the size of the search space becomes $O(m^2 \cdot n)$, which is leveraged by heuristic search methods that only consider updating the bounds of each feature separately instead of jointly (cf. Section 2.2.1). With the update procedure of *Best Interval* (cf. Algorithm 4), the cost for $k = 1$ even reduces to $O(m \cdot n)$ since it only requires one pass over the unique values of each feature to evaluate all lower-upper-bound combinations for WRAcc implicitly. The update procedure of *Beam Search* (cf. Algorithm 3) also requires $O(m \cdot n)$, by only checking updates of either lower or upper bound.

Parameterized complexity For unconstrained subgroup discovery, the complexity term from Proposition 3 is polynomial in m if we consider n to be a small constant. In particular, the term takes the form $O(f(n) \cdot m^{g(n)})$ with parameter n and polynomial functions $f(\cdot)$ and $g(\cdot)$ [33]. Thus, the problem of subgroup discovery belongs to the parameterized complexity class $\mathcal{XP}$:

Proposition 5 (Parameterized complexity of subgroup discovery)**.** *The problem of subgroup discovery (cf. Definition 2) resides in the parameterized complexity class $\mathcal{XP}$ for the parameter n.*

Due to the exponent $2n$ in Proposition 3, an exhaustive search may be infeasible in practice, even for a small, constant n. Further, the complexity remains exponential in n if m is fixed. I.e., the number of features has an exponential impact on the size of the search space, while the number of data objects has a polynomial impact.

With a feature-cardinality constraint, the problem retains its $\mathcal{XP}$ membership. Considering the size of the search space from Proposition 4, the parameter is k instead of n now:

Proposition 6 (Parameterized complexity of subgroup discovery with feature–cardinality constraint)**.** *The problem of subgroup discovery (cf. Definition 2) with a feature-cardinality constraint (cf. Definition 8) resides in the parameterized complexity class $\mathcal{XP}$ for the parameter k.*

NP-Hardness [22] showed that it is an $\mathcal{NP}$-hard problem to find a subgroup description with minimum feature cardinality that induces exactly the same subgroup membership as a given subgroup. We transfer this result to optimizing subgroup quality under a feature-cardinality constraint. First, we tackle the search problem for perfect subgroups (cf. Appendix A.2.1 for the proof):

Proposition 7 (Complexity of perfect-subgroup discovery with feature-cardinality constraint)**.** *The problem of perfect-subgroup discovery (cf. Definition 6) with a feature-cardinality constraint (cf. Definition 8) is $\mathcal{NP}$-complete.*

This hardness result under a feature-cardinality constraint contrasts with the polynomial runtime of unconstrained perfect-subgroup discovery (cf. Proposition 2), which corresponds to a cardinality constraint with $k = n$.

Generalizing Proposition 7, the optimization problem of subgroup discovery with a feature-cardinality constraint is $\mathcal{NP}$-complete under a reasonable assumption on the notion of subgroup quality (cf. Appendix A.2.2 for the proof):

Proposition 8 (Complexity of subgroup discovery with feature-cardinality constraint)**.** *Assuming a subgroup-quality function $Q(lb, ub, X, y)$ for which only perfect subgroups (cf. Definition 5) reach its maximal value, the problem of subgroup discovery (cf. Definition 2) with a feature-cardinality constraint (cf. Definition 8) is $\mathcal{NP}$-complete.*

WRAcc as the subgroup-quality function satisfies this assumption since only perfect subgroups yield the theoretical maximum WRAcc (cf. Equation 2).

4.4 Alternative Subgroup Descriptions

In this section, we propose the optimization problem of discovering alternative subgroup descriptions. First, we motivate and formalize the problem (cf. Section 4.4.1). Next, we describe how to phrase it within our SMT encoding of subgroup discovery (cf. Section 4.4.2), heuristic search methods (cf. Section 4.4.3), and baselines (cf. Section 4.4.4). Finally, we analyze the time complexity of this problem (cf. Section 4.4.5).

4.4.1 Concept

Overview For alternative subgroup descriptions, we assume to have an *original subgroup* given, with subgroup membership $b^{(0)} \in \{0,1\}^m$ of data objects and with feature selection $s^{(0)} \in \{0,1\}^n$. This subgroup may originate from any subgroup-discovery method. When searching alternatives, we do not optimize subgroup quality but the similarity to the original subgroup. We express this similarity in terms of subgroup membership. If this similarity is very high, then the subgroup quality should also be similar since evaluation metrics for subgroup quality typically base on subgroup membership.

Additionally, we constrain the new subgroup description to be alternative enough. We express this dissimilarity in terms of the subgroups' feature selection. The user chooses a dissimilarity threshold $\tau \in \mathbb{R}_{\geq 0}$ and can thereby control alternatives. Further, we recommend employing a feature-cardinality constraint (cf. Definition 8) when determining the original subgroup, so there are sufficiently many features left for the alternative description. The alternative may be feature-cardinality-constrained as well, to increase its interpretability.

In a nutshell, alternative subgroup descriptions should produce similar predictions as the original subgroup but use different features.

Sequential search One can search for multiple alternative subgroup descriptions sequentially. After determining the original subgroup, each iteration yields one further alternative. The user may prescribe a number of alternatives $a \in \mathbb{N}$ a priori or interrupt the procedure whenever the alternatives are not interesting anymore, e.g., too dissimilar to the original subgroup. Each alternative should have a similar subgroup membership as the original subgroup but a dissimilar feature selection compared to all *existing subgroups*, i.e., subgroups found in prior iterations. The following definition captures this optimization problem:

Definition 9 (Alternative-subgroup-description discovery)**.** Given

- a dataset $X \in \mathbb{R}^{m \times n}$,
- $a \in \mathbb{N}$ existing subgroups with subgroup membership $b^{(l)} \in \{0,1\}^m$ and feature selection $s^{(l)} \in \{0,1\}^n$ for $l \in \{0, \dots, a-1\}$,
- a similarity measure $\mathrm{sim}(\cdot)$ for subgroup-membership vectors,
- a dissimilarity measure $\mathrm{dis}(\cdot)$ for feature-selection vectors of subgroups,
- and a dissimilarity threshold $\tau \in \mathbb{R}_{\geq 0}$,

alternative-subgroup-description discovery is the problem of finding a subgroup (cf. Definition 1) with membership $b^{(a)} \in \{0,1\}^m$ and feature selection $s^{(a)} \in \{0,1\}^n$ that maximizes the subgroup-membership similarity $\mathrm{sim}(b^{(a)}, b^{(0)})$ to the original subgroup while being dissimilar to all existing subgroups regarding the feature selection, i.e., $\forall l \in \{0, \dots, a-1\} : \ \mathrm{dis}(s^{(a)}, s^{(l)}) \geq \tau$.

Compared to the conventional subgroup-discovery problem (cf. Definition 2), $b^{(0)}$ replaces the prediction target y, and $\mathrm{sim}(\cdot)$ replaces the subgroup-quality function $Q(lb, ub, X, y)$. In the following, we discuss $\mathrm{sim}(\cdot)$ and $\mathrm{dis}(\cdot)$.

Similarity in objective function There are various options to quantify the similarity of subgroup membership, i.e., between two binary vectors. For example, the Hamming distance counts how many vector entries differ [28]. We turn this distance into a similarity measure by counting identical vector entries. Further, we normalize the similarity with the vector length, i.e., number of data objects m, to obtain the following *normalized Hamming similarity* for two subgroup-membership vectors $b', b'' \in \{0,1\}^m$:

$$\mathrm{sim}_{\mathrm{nHamm}}(b', b'') = \frac{1}{m} \cdot \sum_{i=1}^{m} (b'_i = b''_i) \tag{12}$$

If either b' or b'' is constant, then this similarity measure is linear in its remaining argument, as discussed later (cf. Equation 15). Further, if one considers one vector to be a prediction and the other to be the ground truth, Equation 12 equals prediction accuracy for classification.

Another popular similarity measure for sets or binary vectors is the Jaccard index [28], which relates the overlap of positive vector entries to their union:

$$\mathrm{sim}_{\mathrm{Jacc}}(b', b'') = \frac{\sum_{i=1}^{m} (b'_i \wedge b''_i)}{\sum_{i=1}^{m} (b'_i \vee b''_i)} \tag{13}$$

However, this similarity measure is not linear in b' and b'', which prevents its use in certain white-box solvers. Thus, we use the normalized Hamming similarity as the objective function.

Dissimilarity in constraints There are various options to quantify the dissimilarity between feature-selection vectors. We employ the following *deselection dissimilarity* in combination with an adapted dissimilarity threshold:

$$\mathrm{dis}_{\mathrm{des}}(s^{\mathrm{new}}, s^{\mathrm{old}}) = \sum_{j=1}^{n} (\neg s_j^{\mathrm{new}} \wedge s_j^{\mathrm{old}}) \geq \min\left(\tau_{\mathrm{abs}},\ k^{\mathrm{old}}\right) \tag{14}$$

This dissimilarity counts how many of the previously selected features are *not* selected in the new subgroup description. These features may either be replaced by other features, or the total number of selected features may be reduced. The constraint ensures that at least $\tau_{\mathrm{abs}} \in \mathbb{N}$ features are deselected but never more than there were selected before (k^{old}), which would be infeasible. For maximum dissimilarity, none of the previously selected features may be selected again. Note that this dissimilarity measure is asymmetric, i.e., $\mathrm{dis}_{\mathrm{des}}(s^{\mathrm{new}}, s^{\mathrm{old}}) \neq \mathrm{dis}_{\mathrm{des}}(s^{\mathrm{old}}, s^{\mathrm{new}})$. While this property would be an issue in a simultaneous search for multiple alternatives, i.e., without an explicit ordering, it is acceptable for sequential search, where 'old' and 'new' are well-defined.

Conceptually, one could also employ a more common dissimilarity measure like the Jaccard distance or the Dice dissimilarity [28]. The latter two are even symmetric and normalized to $[0, 1]$. However, our deselection dissimilarity has two advantages: First, if s^{old} is constant, the dissimilarity is linear in s^{new}, as it amounts to a simple sum, even if the exact number of newly selected features is unknown yet. This property is useful for solver-based search (cf. Section 4.4.2). In contrast, Jaccard distance and Dice dissimilarity involve a ratio and are therefore non-linear. Second, the constraint from Equation 14 is antimonotonic in the new feature selection, which is useful for heuristic search (cf. Section 4.4.3). Using the Jaccard distance or Dice dissimilarity in the constraint violates this property. In particular, these dissimilarities can increase by selecting features that were not selected in the existing subgroup, i.e., an invalid feature set can become valid instead of remaining invalid by selecting further features.

4.4.2 SMT Encoding

We only need to reformulate Equation 12 slightly to obtain a linear objective function regarding the alternative subgroup-membership vector $b^{(a)}$:

$$\begin{aligned} \mathrm{sim}_{\mathrm{nHamm}}(b^{(a)}, b^{(0)}) &= \frac{1}{m} \cdot \sum_{i=1}^{m} \left(b_i^{(a)} \leftrightarrow b_i^{(0)} \right) \\ &= \frac{1}{m} \cdot \Big(\sum_{\substack{i \in \{1,\dots,m\} \\ b_i^{(0)}=1}} b_i^{(a)} + \sum_{\substack{i \in \{1,\dots,m\} \\ b_i^{(0)}=0}} \neg b_i^{(a)} \Big) \end{aligned} \tag{15}$$

In particular, since $b^{(0)}$ is known and therefore constant, we employ the expression from the second line, i.e., without the logical equivalence operator. Instead, we compute two sums, one for data objects that are members of the original subgroup and one for non-members. The negated expression $\neg b_i^{(a)}$ may be expressed as $1 - b_i^{(a)}$.

To formulate the dissimilarity constraints, we leverage that the feature-selection vector $s^{(l)}$ and the corresponding number of selected features $k^{(l)}$ are known for all existing subgroups as well. Thus, we instantiate and adapt Equation 14 as follows:

$$\forall l \in \{0, \ldots, a-1\} : \ \mathrm{dis}_{\mathrm{des}}(s^{(a)}, s^{(l)}) = \sum_{\substack{j \in \{1,\ldots,n\} \\ s_j^{(l)}=1}} \neg s_j^{(a)} \geq \min\left(\tau_{\mathrm{abs}},\ k^{(l)}\right) \quad (16)$$

In particular, we only sum over features that were selected in the existing subgroup and check whether they are deselected now. To tie the variables $s_j^{(a)}$ to the subgroup's bounds, we use Equation 9, which we already employed for feature cardinality constraints.

Finally, the overall SMT encoding of alternative-subgroup-description discovery (cf. Definition 9) combines the similarity objective (cf. Equation 15) and dissimilarity constraints (cf. Equation 16) for alternatives with the previously introduced variables and constraints for bounds (cf. Equation 6), subgroup membership (cf. Equation 7), and feature selection (cf. Equation 9). Optionally, one may add a feature-cardinality constraint (cf. Equation 10).

4.4.3 Integration into Heuristic Search Methods

The situation here is similar to integrating feature-cardinality constraints into heuristic search methods (cf. Section 4.3.3). In particular, the constraint for alternatives based on the deselection dissimilarity (cf. Equation 14) is antimonotonic in the subgroup's selected feature set. I.e., the dissimilarity constraint is satisfied for an empty set of selected features, and once it is violated for a feature set, it remains violated for any superset. Thus, the constraint type is suitable for heuristic search that iteratively enlarges the set of selected features, like *PRIM* (cf. Algorithm 1), *Beam Search* (cf. Algorithms 2 and 3), and *Best Interval* (cf. Algorithms 2 and 4). We only need to adapt the function *get_permissible_feature_idxs(...)* (cf. Line 7 in Algorithm 1 and Line 12 in Algorithm 2) to check the constraint. I.e., the function should return the indices of all features that may be selected into the subgroup without violating the dissimilarity constraint (cf. Equation 14). In particular, once $k^{(l)} - \tau_{\mathrm{abs}}$ features from an existing subgroup with $k^{(l)}$ features are selected again, no further features from this subgroup may be selected.

4.4.4 Integration into Baselines

Adapting our two baselines to alternative subgroup descriptions is less straightforward than to feature-cardinality constraints (cf. Section 4.3.4) since the op-

timization objective changes and the search space under the dissimilarity constraint (cf. Equation 14) is harder to describe. Thus, we did not implement and evaluate concrete adaptations but still discuss possible ideas in the following.

MORS A major issue for adapting *MORS* (cf. Algorithm 5) is that *MORS* is tailored to a particular objective, i.e., perfect subgroup quality in terms of recall. In contrast, alternative subgroup descriptions should optimize subgroup-membership similarity to an original subgroup. Also, the normalized Hamming similarity (cf. Equation 12) for alternatives measures accuracy rather than recall, i.e., it considers all data objects rather than only the positive ones.

For the dissimilarity constraint, we would like to enforce a valid feature set by implementing the function *get_permissible_feature_idxs(...)* in Line 6 of Algorithm 5 appropriately. The univariate, quality-based selection heuristic we proposed for feature-cardinality constraints (cf. Section 4.3.4) may produce an invalid solution. To alleviate this issue, we could adapt this heuristic as follows: Still order the features by their univariate quality and iteratively select them in this order, but check the dissimilarity constraint in each iteration and skip over features that violate it.

Random Search For *Random Search* (cf. Algorithm 6), changing the optimization objective from subgroup quality to subgroup-membership similarity is not an issue since the objective is treated as a black-box function for evaluating randomly generated subgroups (cf. Line 7 of Algorithm 6). For the dissimilarity constraint, we would like to implement *get_permissible_feature_idxs(...)* in Line 5 of Algorithm 6) by uniformly sampling from the constrained search space. In general, uniform sampling from a constrained space is a computationally hard problem [35], though it may be feasible for the particular constraint type. We could also sample from the unconstrained space and then check the dissimilarity constraint, repeating sampling till a valid feature set is found. However, this strategy may produce a high fraction of invalid solution candidates, depending on how strong the constraint is for the particular dataset and user parameters. Another option would be using non-uniform sampling, e.g., only sample features not selected in any existing subgroup. This guarantees constraint satisfaction but does not cover the entire constrained search space since it ignores the feature-set overlap allowed by the dissimilarity threshold τ.

4.4.5 Time Complexity

As for the feature-cardinality constraint (cf. Section 4.3.5), we analyze three aspects of time complexity: the size of the search space for exhaustive search, parameterized complexity, and $\mathcal{NP}$-hardness.

Exhaustive search The search for an alternative subgroup description can iterate over the same solution candidates as the search for an original subgroup description, i.e., all combinations of bound values over features. Thus, the results from Propositions 3 and 4 remain valid:

Proposition 9 (Complexity of exhaustive search for alternative-subgroup-description discovery)**.** *An exhaustive search for alternative-subgroup-description discovery (cf. Definition 9) needs to evaluate $O(m^{2n})$ subgroups for each alternative in general or $O(n^k \cdot m^{2k})$ subgroups for each alternative if a feature-cardinality constraint (cf. Definition 8) is employed.*

The evaluation of solution candidates differs from the original subgroup descriptions but has a similar time complexity, i.e., $O(m \cdot n + a \cdot n)$ instead of $O(m \cdot n)$. In particular, evaluating the subgroup-membership-similarity-based objective function (e.g., Equation 12) should typically have a cost of $O(m \cdot n)$, like subgroup-quality functions have. Unlike the unconstrained case, some solution candidates violate the dissimilarity constraint (e.g., Equation 14) and need not be evaluated. The corresponding constraint check requires determining the selected features and computing the dissimilarity. The former (cf. Definition 7) runs in $O(n)$ if the minimum and maximum feature values of the dataset are precomputed. The latter should typically entail a cost of $O(n)$ per existing subgroup description for reasonably simple dissimilarity functions.

Parameterized complexity Due to the similar search space as for original subgroup descriptions, the parameterized-complexity results from Propositions 5 and 6 apply to discovering alternative subgroup descriptions as well:

Proposition 10 (Parameterized complexity of alternative-subgroup-description discovery)**.** *The problem of alternative-subgroup-description discovery (cf. Definition 9) resides in the parameterized complexity class $\mathcal{XP}$ for the parameter n in general and for the parameter k if a feature-cardinality constraint (cf. Definition 8) is employed.*

NP-Hardness We prove $\mathcal{NP}$-completeness for a special case of alternative-subgroup-description discovery (cf. Definition 9) first. To this end, we introduce the following definition:

Definition 10 (Perfect alternative subgroup description)**.** Given

- a dataset $X \in \mathbb{R}^{m \times n}$,
- an original subgroup with subgroup membership $b^{(0)} \in \{0,1\}^m$ and feature selection $s^{(0)} \in \{0,1\}^n$,
- a dissimilarity measure $\mathrm{dis}(\cdot)$ for feature-selection vectors of subgroups,
- and a dissimilarity threshold $\tau \in \mathbb{R}_{\geq 0}$,

a *perfect alternative subgroup description* defines a subgroup (cf. Definition 1) with membership $b^{(a)} \in \{0,1\}^m$ and feature selection $s^{(a)} \in \{0,1\}^n$ that exactly replicates the subgroup membership of the original subgroup, i.e., $b^{(a)} = b^{(0)}$, while being dissimilar regarding the feature selection, i.e., $\mathrm{dis}(s^{(a)}, s^{(0)}) \geq \tau$.

In particular, the value of the subgroup-membership similarity is fixed here rather than an optimization objective. Similar to perfect subgroups (cf. Definition 5), perfect alternative subgroup descriptions only exist in some datasets. Next, we define a corresponding search problem:

Definition 11 (Perfect-alternative-subgroup-description discovery)**.** Given

- a dataset $X \in \mathbb{R}^{m \times n}$,
- an original subgroup with subgroup membership $b^{(0)} \in \{0,1\}^m$ and feature selection $s^{(0)} \in \{0,1\}^n$,
- a dissimilarity measure $\mathrm{dis}(\cdot)$ for feature-selection vectors of subgroups,
- and a dissimilarity threshold $\tau \in \mathbb{R}_{\geq 0}$,

perfect-alternative-subgroup-description discovery is the problem of finding a perfect alternative subgroup description (cf. Definition 10) if it exists or determining that it does not exist.

Next, we prove the following hardness result for this search problem with a perfect original subgroup and under a reasonable assumption on the notion of feature-selection dissimilarity (cf. Appendix A.2.3 for the proof):

Proposition 11 (Complexity of perfect-alternative-subgroup-description discovery with feature-cardinality constraint and perfect original subgroup)**.** *Assuming*

- *a combination of a dissimilarity measure dis*$(\cdot)$ *and a dissimilarity threshold* $\tau \in \mathbb{R}_{\geq 0}$ *that prevents selecting any selected feature from the original subgroup description again,*
- *and the original subgroup being perfect (cf. Definition 5),*

the problem of perfect-alternative-subgroup-description discovery (cf. Definition 11) with a feature-cardinality constraint (cf. Definition 8) is $\mathcal{NP}$*-complete.*

Our deselection dissimilarity (cf. Equation 14) as $\mathrm{dis}(\cdot)$ satisfies the dissimilarity assumption if we choose a dissimilarity threshold $\tau_{\mathrm{abs}} \geq k^{\mathrm{old}}$. Other dissimilarity measures should typically also have such a threshold value that enforces zero overlap between the sets of selected features.

The problem naturally remains $\mathcal{NP}$-complete when dropping the two assumptions in Proposition 11. Nevertheless, we explicitly extend this result to imperfect original subgroups (cf. Appendix A.2.4 for the proof):

Proposition 12 (Complexity of perfect-alternative-subgroup-description discovery with feature-cardinality constraint and imperfect original subgroup)**.** *Assuming*

- *a combination of a dissimilarity measure dis*$(\cdot)$ *and a dissimilarity threshold* $\tau \in \mathbb{R}_{\geq 0}$ *that prevents selecting any selected feature from the original subgroup description again,*
- *and the original subgroup* not *being perfect (cf. Definition 5),*

the problem of perfect-alternative-subgroup-description discovery (cf. Definition 11) with a feature-cardinality constraint (cf. Definition 8) is $\mathcal{NP}$*-complete.*

Finally, we switch from the search problem for perfect alternatives to the optimization problem of alternative-subgroup-description discovery. We establish $\mathcal{NP}$-completeness under a reasonable assumption on the notion of subgroup-membership similarity (cf. Appendix A.2.5 for the proof):

Proposition 13 (Complexity of alternative-subgroup-description discovery with feature-cardinality constraint)**.** *Assuming*

- *a combination of a dissimilarity measure $dis(\cdot)$ and a dissimilarity threshold $\tau \in \mathbb{R}_{\geq 0}$ that prevents selecting any selected feature from the original subgroup description again,*
- *and a similarity measure $sim(\cdot)$ for which only perfect alternative subgroup descriptions (cf. Definition 10) reach its maximal value regarding the original subgroup,*

the problem of alternative-subgroup-description discovery (cf. Definition 9) with a feature-cardinality constraint (cf. Definition 8) is $\mathcal{NP}$-complete.

Normalized Hamming similarity (cf. Equation 12) as $\text{sim}(\cdot)$ satisfies the similarity assumption since only perfect alternative subgroup descriptions yield the theoretical maximum similarity of 1 to the original subgroup description.

5 Experimental Design

In this section, we introduce our experimental design. After a brief overview of its components (cf. Section 5.1), we describe subgroup-discovery methods (cf. Section 5.2), experimental scenarios (cf. Section 5.3), evaluation metrics (cf. Section 5.4), and datasets (cf. Section 5.5). Finally, we briefly outline our implementation (cf. Section 5.6).

5.1 Overview

In our experiments, we evaluate eight subgroup-discovery methods on 27 binary-classification datasets. We measure the subgroup quality in terms of nWRAcc and also record the methods' runtime. We analyze four *experimental scenarios*: First, we compare all subgroup-discovery methods without constraints. Second, we vary the timeout in solver-based search. Third, we compare all subgroup-discovery methods with a feature-cardinality constraint, varying the cardinality threshold k. Fourth, we search for alternative subgroup descriptions with one solver-based and one heuristic search method. We vary the number of alternatives a and the dissimilarity threshold τ_{abs}.

5.2 Subgroup-Discovery Methods

We employ eight subgroup-discovery methods: a solver-based one using our SMT encoding (cf. Section 4.1), two exhaustive search methods from related work (cf. Section 2.2.2), three heuristic search methods from related work (cf. Section 2.2.1), and our two baselines (cf. Section 3).

Solver-based search For solver-based search, denoted as *SMT*, we employ the solver *Z3* [21, 31] with our SMT encoding of subgroup discovery (cf. Equation 8). This search method is exhaustive, i.e., it finds the global optimum, if granted sufficient time. In practice, however, we set solver timeouts to control the runtime (cf. Section 5.3), so the method may not be truly exhaustive.

Exhaustive algorithmic search We evaluate two exhaustive algorithmic search methods from related work (cf. Section 2.2.2), i.e., *BSD* [73] and *SD-Map* [8], based on their implementation in the package *SD4Py* [54]. We chose these two methods after comparing the runtimes of several algorithms from four Python packages for subgroup discovery (cf. Section A.3). Both methods require discretizing numeric features, which makes them non-exhaustive regarding the full numeric search space in our experiments. We use their built-in equal-width discretization and tune it by choosing the best subgroup quality out of ten different bin counts. i.e., $\{2, 3, 4, 5, 10, 15, 20, 30, 40, 50\}$. Each subgroup description may comprise at most one bin for each feature.

Heuristic search We evaluate three heuristic search methods from related work (cf. Section 2.2.1): *PRIM* [38] (cf. Algorithm 1), *Beam Search* (cf. Algorithms 2 and 3), subsequently called *Beam*, and *Best Interval* [82] (cf. Algorithms 2 and 4), subsequently called *BI*. We set the peeling fraction of *PRIM* to $\alpha = 0.05$, consistent with other implementations [2, 63] and within the recommended value range proposed by its authors [38]. Further, we set the support threshold to $\beta_0 = 0$, so there is no constraint on the subgroup's size. For *Beam* and *BI*, we choose a beam width of $w = 10$, falling between default values used in other implementations [2, 72].

Baselines We also include baselines that are simpler than the heuristic search methods. In particular, we employ our own methods *MORS* (cf. Algorithm 5) and *Random Search* (cf. Algorithm 6, subsequently called *Random*). *MORS* is parameter-free. For *Random*, we set the number of iterations $n\_iters = 1000$.

5.3 Experimental Scenarios

We evaluate the subgroup-discovery methods in four experimental scenarios. Two of the scenarios do not involve all subgroup-discovery methods.

Unconstrained subgroup discovery Our first experimental scenario (cf. Section 6.1 for results) compares all eight subgroup-discovery methods without constraints. This comparison assesses the effectiveness of the solver-based search method *SMT* for conventional subgroup discovery and serves as a reference point for subsequent experiments with constraints. All methods except *MORS* use WRAcc (cf. Equation 1) as the subgroup-quality function $Q(lb, ub, X, y)$ for search. *MORS* optimizes its built-in objective (cf. Definition 4).

Solver timeouts Our second experimental scenario (cf. Section 6.2 for results) takes a deeper dive into *SMT* as the subgroup-discovery method. In particular, we analyze whether setting solver timeouts enables finding solutions with reasonable quality in a shorter time frame. If the solver does not finish optimization within a given timeout, we record the currently best solution at this time, which may be suboptimal. Note that the timeout only applies to the optimization procedure, while our runtime measurements also include the time for formulating the optimization problem upfront.

We evaluate twelve exponentially scaled timeout values, i.e., {1 s, 2 s, 4 s, ..., 2048 s}. In the three other experimental scenarios, we employ the maximum timeout of 2048 s for *SMT*. Since the heuristic search methods and baselines are significantly faster, we do not conduct a timeout analysis for them. The two exhaustive algorithmic search methods *BSD* and *SD-Map* are usually fast as well. However, they did not finish within two days in the unconstrained scenario for five datasets, and they do not have a timeout parameter. We replace these missing values with the results for a feature-cardinality threshold of $k = 5$.

Feature-cardinality constraints Our third experimental scenario (cf. Section 6.3 for results) analyzes feature-cardinality constraints (cf. Section 4.3) for all eight subgroup-discovery methods. In particular, we study $k \in \{1, 2, 3, 4, 5\}$ selected features. These values of k are upper bounds (cf. Equation 10), i.e., the subgroup-discovery methods may select fewer features if selecting more does not improve subgroup quality.

Alternative subgroup descriptions Our fourth experimental scenario (cf. Section 6.4 for results) studies alternative subgroup descriptions (cf. Section 4.4) for *SMT* and *Beam*, i.e., one solver-based and one heuristic search method. This scenario still optimizes WRAcc (cf. Equation 1) for the original subgroup but normalized Hamming similarity (cf. Equation 12) for the alternatives. The deselection dissimilarity (cf. Equation 14) ensures that the alternatives are dissimilar enough. We limit the feature cardinality to $k = 3$, which yields reasonably high subgroup quality (cf. Section 6.3). We search for $a = 5$ alternative subgroup descriptions with a dissimilarity threshold $\tau_{\text{abs}} \in \{1, 2, 3\}$. Since each dataset has $n \geq 20$ features (cf. Section 5.5), our choices of a, k, and τ ensure that there always is a valid alternative.

5.4 Evaluation Metrics

Subgroup quality We use *nWRAcc* (cf. Equation 3) to report subgroup quality. To analyze how well the subgroup descriptions generalize, we conduct a stratified five-fold cross-validation. In particular, each run of a subgroup-discovery method uses only 80% of a dataset's data objects as training data, while the remaining data objects serve as test data. Based on the bounds of each found subgroup, we determine subgroup membership for all data objects and compute *training-set nWRAcc* and *test-set nWRAcc* on the corresponding part of the data separately, using the true class labels y.

Subgroup similarity For evaluating alternative subgroup descriptions, we consider not only their quality but also their induced subgroup's similarity to the original subgroup. To this end, we use *normalized Hamming similarity* (cf. Equation 12) and *Jaccard similarity* (cf. Equation 13) to compare subgroup membership of data objects between the original and the alternative.

Runtime As *runtime*, we report the training time of the subgroup-discovery methods. In particular, we measure how long the search for each subgroup takes. For solver-based search, we also record whether the solver timed out.

5.5 Datasets

We use binary-classification datasets from the Penn Machine Learning Benchmarks (PMLB) [96, 104]. If the class labels occur with different frequencies, we encode the minority class as positive, i.e., assign 1 as its class label. To avoid prediction scenarios that may be too easy or do not have enough features for alternative subgroup descriptions, we only select datasets with at least 100 data objects and 20 features. Next, we exclude one dataset with 1000 features, which has a significantly higher dimensionality than all remaining datasets and would dominate runtime. Finally, we manually exclude datasets that seem to be duplicated or modified versions of other datasets in our experiments.

Based on these criteria, we obtain 27 datasets with 106 to 9822 data objects and 20 to 168 features (cf. Table 1). The datasets do not contain any missing values. Further, PMLB encodes categorical features ordinally by default.

5.6 Implementation and Execution

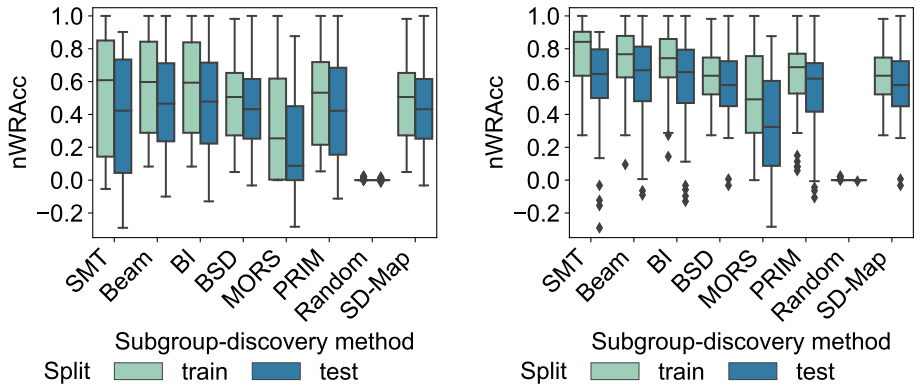
We implemented all subgroup-discovery methods, experiments, and evaluations in Python 3.8. The code is available on *GitHub*[1] and additionally backed up in the *Software Heritage archive*[2]. A requirements file in our repository specifies the versions of all dependencies. Further, we released the subgroup-discovery methods and evaluation metrics as the Python package *csd*[3] on *PyPI* to ease reuse. Finally, we published all experimental data[4].

Our experimental pipeline parallelizes over datasets, cross-validation folds, and subgroup-discovery methods, while each of these experimental tasks runs single-threaded. We ran the pipeline on a server with 160 GB RAM and an *AMD EPYC 7551* CPU, having 32 physical cores and a base clock of 2.0 GHz. With this hardware, the parallelized pipeline run took approximately 34 hours.

[1] `https://github.com/Jakob-Bach/Constrained-Subgroup-Discovery`
[2] swh:1:dir:8119c420804e9027b12cf3aaf82ce12157c049e1
[3] `https://pypi.org/project/csd/`
[4] `https://doi.org/10.35097/8ppb5x50nyvw1wa7`

Table 1: Datasets from PMLB used in our experiments. m denotes the number of data objects and n the number of features. In dataset names, we replaced *GAMETES_Epistasis* with *GE_* and *GAMETES_Heterogeneity* with *GH_* to reduce the table's width. *Timeouts* indicates whether at least one timeout occurred with *SMT* as the subgroup-discovery method and the highest timeout setting (2048 s), optimizing the original subgroup without cardinality constraints (*Max* k) or in any cardinality setting (*Any* k).

Dataset	m	n	Timeouts	
			Max k	Any k
backache	180	32	No	No
chess	3196	36	No	No
churn	5000	20	Yes	Yes
clean1	476	168	No	No
clean2	6598	168	No	No
coil2000	9822	85	Yes	Yes
credit_g	1000	20	Yes	Yes
dis	3772	29	No	No
GE_2_Way_20atts_0.1H_EDM_1_1	1600	20	Yes	Yes
GE_2_Way_20atts_0.4H_EDM_1_1	1600	20	No	No
GE_3_Way_20atts_0.2H_EDM_1_1	1600	20	Yes	Yes
GH_20atts_1600_Het_0.4_0.2_50_EDM_2_001	1600	20	Yes	Yes
GH_20atts_1600_Het_0.4_0.2_75_EDM_2_001	1600	20	Yes	Yes
Hill_Valley_with_noise	1212	100	Yes	Yes
horse_colic	368	22	No	No
hypothyroid	3163	25	No	No
ionosphere	351	34	No	No
molecular_biology_promoters	106	57	No	No
mushroom	8124	22	No	No
ring	7400	20	Yes	Yes
sonar	208	60	No	Yes
spambase	4601	57	No	Yes
spect	267	22	No	No
spectf	349	44	No	Yes
tokyo1	959	44	No	Yes
twonorm	7400	20	Yes	Yes
wdbc	569	30	No	No



(a) All 27 datasets.

(b) 17 datasets without *SMT* timeouts.

Figure 2: Distribution of subgroup quality over datasets and cross-validation folds, by subgroup-discovery method. Results from the unconstrained experimental scenario.

6 Evaluation

In this section, we evaluate our experiments. In particular, we cover our four experimental scenarios, i.e., unconstrained subgroup discovery (cf. Section 6.1), solver timeouts (cf. Section 6.2), feature-cardinality constraints (cf. Section 6.3), and alternative subgroup descriptions (cf. Section 6.4). Finally, we summarize key experimental results (cf. Section 6.5).

6.1 Unconstrained Subgroup Discovery

In this section, we compare all eight subgroup-discovery methods in the experimental scenario without constraints. *SMT* uses its default maximum solver timeout of 2048 s.

Subgroup quality Figure 2a visualizes the subgroup quality achieved by the eight subgroup-discovery methods. On the training set, the two heuristic search methods *Beam* and *BI* have roughly the same median nWRAcc as the solver-based search method *SMT*. In particular, the heuristics are even better than *SMT* on some datasets but worse on others. The former can only happen when *SMT* yields suboptimal solutions due to timeouts, as we analyze later (cf. Section 6.2). However, even if we limit our analysis to the 17 datasets without *SMT* timeouts, *Beam* and *BI* are still remarkably close to the optimum quality (cf. Figure 2b). This result is not specific to *SMT* but also sets up *Beam* and *BI* as serious competitors for any other exhaustive search method.

On the test set, *Beam* and *BI* are even better than *SMT* on median, also excluding timeout datasets, since their training-test nWRAcc difference is smaller. This result indicates that *Beam* and *BI* are less susceptible to overfitting,

so their solutions generalize better. In detail, the average difference between training-set nWRAcc and test-set nWRAcc is 0.122 for *SMT*, 0.101 for *BI*, 0.095 for *Beam*, 0.094 for *MORS*, 0.068 for *PRIM*, 0.042 for *SD-Map*, 0.041 for *BSD*, and 0.001 for *Random*.

BSD and *SD-Map* yield worse average subgroup quality than *Beam* and *BI* as well. While the former two are exhaustive, they require discretizing numeric features. Thus, they effectively only have a fixed set of intervals to use as bounds instead of being able to choose the bounds independently at each possible feature value. As a positive side-effect, this limitation may have reduced their overfitting and runtime. *PRIM*'s worse subgroup quality may equally arise from a reduced search space. Although it follows an iterative subgroup-refinement procedure like *Beam* and *BI*, its refinement options are more limited. In particular, *PRIM* always has to remove a fixed fraction α of data objects from the subgroup (cf. Algorithm 1), while *Beam* and *BI* can remove more or fewer data objects. Finally, the two baselines *MORS* and *Random* yield the lowest quality of all eight subgroup-discovery methods. While *Random* mostly yields the same quality as not restricting the subgroup at all, i.e., an nWRAcc of 0, *MORS* is considerably above 0 and thus a suitable baseline for future studies comparing subgroup-discovery methods.

Runtime Table 2 summarizes the runtimes of the subgroup-discovery methods. On average, *SMT* is an order of magnitude slower than *Beam* and *BI*. The exhaustive search methods with discretization, i.e., *BSD* and *SD-Map*, fall in between. The heuristic *PRIM* and the baseline *Random* are yet another order of magnitude faster than *Beam* and *BI*. Finally, the baseline *MORS* runs in negligible time and, therefore, is a good tool for instantaneously obtaining a lower bound on subgroup quality. Taking subgroup quality into consideration, the heuristic search methods offer a good quality in a short time. Among the three heuristics, *PRIM* is the fastest but yields the lowest subgroup quality, so users should decide which runtime is acceptable.

For *SMT*, the overall runtime not only comprises optimization but also formulating the optimization problem. Since the latter depends on the dataset size, e.g., involves $O(m)$ constraints with length $O(n)$ each to define the subgroup-membership variables b_i (cf. Equation 7), the preparation time can become considerable for large datasets. In our experiments, formulating the *SMT* problem took 45 s on average, with a maximum of 379 s. This average preparation time is already greater than the average total runtime of the heuristics.

To determine which factors influence runtime, we analyze the Spearman correlation between runtime and four simple metrics for dataset size. In particular, Table 3 considers the number of data objects m, the number of features n, the product of these two quantities $m \cdot n$, and the number of unique values per feature summed over the features Σn^u. For the three heuristic search methods, the latter metric shows a high correlation to runtime, while the three exhaustive search methods exhibit the highest runtime correlation to $m \cdot n$.

Table 2: Aggregated runtime (in seconds) over datasets and cross-validation folds, by subgroup-discovery method. Results from the unconstrained experimental scenario.

(a) All 27 datasets.

Method	Mean	Standard dev.	Median
BI	34.95	103.61	2.60
BSD	55.70	151.46	1.06
Beam	30.47	85.69	2.95
MORS	0.01	0.00	0.01
PRIM	1.26	1.51	0.66
Random	0.91	0.95	0.51
SD-Map	367.43	1143.05	4.08
SMT	849.02	929.60	254.21

(b) 17 datasets without *SMT* timeouts.

Method	Mean	Standard dev.	Median
BI	12.40	21.17	2.60
BSD	44.02	128.33	1.57
Beam	11.77	20.47	2.95
MORS	0.01	0.00	0.01
PRIM	1.29	1.62	0.80
Random	0.82	0.89	0.56
SD-Map	109.92	290.69	12.56
SMT	168.13	243.11	57.23

6.2 Solver Timeouts

In this section, we evaluate the impact of solver timeouts for *SMT* search.

Finished tasks Figure 3a displays how many of the *SMT* optimization tasks for original subgroups finished within the evaluated solver timeouts. Besides the unconstrained tasks, we also consider tasks with different feature-cardinality thresholds, though the overall trend is the same. In particular, the number of finished tasks only increases slowly over time, and some tasks take orders of magnitude longer than others. E.g., in the unconstrained experimental scenario, 21.5% of the *SMT* tasks finished within 4 s, 24.4% within 16 s, 37.0% within 64 s, 54.8% within 256 s, and 62.2% within 1024 s. For the maximum timeout setting of 2048 s, 67.4% of the *SMT* tasks finished, and 17 out of 27 datasets did not encounter timeouts (cf. Table 1).

Table 3: Spearman correlation between runtime and metrics for dataset size, over datasets and cross-validation folds, by subgroup-discovery method. Results from the unconstrained experimental scenario, using the 17 datasets without *SMT* timeouts.

Method	Σn^u	$m \cdot n$	m	n
BI	0.95	0.51	0.32	0.67
BSD	0.46	0.60	0.44	0.42
Beam	0.96	0.49	0.30	0.66
MORS	0.27	0.57	0.51	0.26
PRIM	0.84	0.56	0.29	0.76
Random	0.58	0.69	0.42	0.77
SD-Map	0.43	0.65	0.47	0.45
SMT	0.39	0.73	0.70	0.23

Subgroup quality Figure 3b visualizes the subgroup quality over solver timeouts for unconstrained *SMT* search. This plot uses the quality of the optimal solution for finished tasks and of the currently best solution for unfinished tasks. As for the number of finished tasks (cf. Figure 3a), the largest gains occur in the first minute. E.g., the mean test-set nWRAcc over datasets and cross-validation folds is 0.10 for 1 s, 0.19 for 4 s, 0.24 for 16 s, 0.32 for 64 s, and 0.39 for the maximum solver timeout of 2024 s. The main cause for this trend is that many tasks finish relatively early (cf. Figure 3a), and finished tasks cannot improve their quality for higher solver timeouts. In contrast, if we only consider the tasks where the solver did not finish even within the maximum solver timeout, the quality increase of the currently best solution over time is marginal.

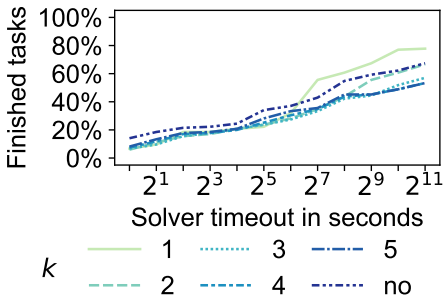
Further, even *SMT* with a timeout does not compare favorably to fast heuristic search methods. E.g., with a solver timeout of 64 s, corresponding to an average overall runtime of 88 s, *SMT* achieves a mean training-set nWRAcc of 0.43, compared to 0.56 for *Beam* with an average runtime of 30 s (cf. Table 2a).

Finally, setting a lower solver timeout decreases overfitting, i.e., the difference between training-set nWRAcc and test-set nWRAcc increases over time (cf. Figure 3b). However, since the test-set nWRAcc still increases with the timeout, choosing lower timeouts does not help quality-wise.

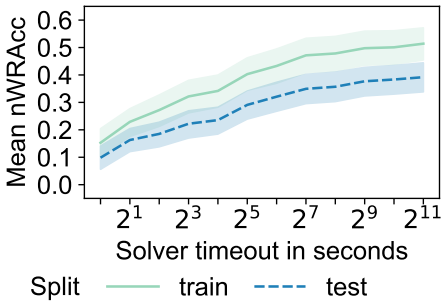
6.3 Feature-Cardinality Constraints

In this section, we compare all eight subgroup-discovery methods in the experimental scenario with feature-cardinality constraints. *SMT* uses its default maximum solver timeout of 2048 s.

Subgroup quality Figure 4 displays the mean subgroup quality, averaging over datasets and cross-validation folds, for different values of the feature-cardinality threshold k. For most subgroup-discovery methods, mean training-



(a) Frequency of finished *SMT* tasks over datasets and cross-validation folds, by feature cardinality k.



(b) Mean subgroup quality, with 95% confidence intervals based on datasets and cross-validation folds. Results from the unconstrained experimental scenario.

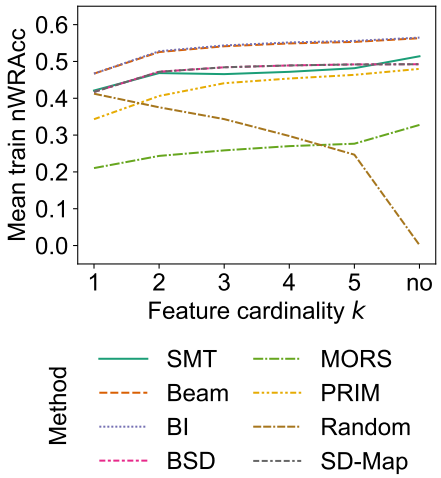
Figure 3: Impact of solver timeouts for *SMT* as the subgroup-discovery method. Results from the search for original subgroups.

set nWRAcc (cf. Figure 4a) increases with k, though the marginal utility decreases. In particular, even with $k = 1$, the mean nWRAcc is already clearly above 50% of the nWRAcc achieved with all features. Further, the quality increase between $k = 1$ and $k = 2$ is usually the largest. On the test set (cf. Figure 4b), the benefit of a larger k is even smaller: The mean test-set nWRAcc of all methods except *PRIM* barely improves beyond $k = 2$. These results indicate that sparse subgroup descriptions already yield a high subgroup quality.

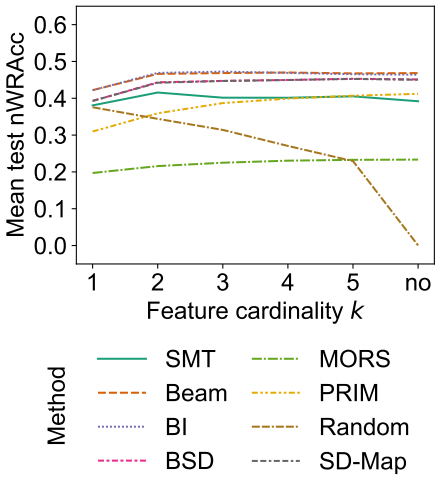
The baseline *Random* even improves subgroup quality with lower k due to its particular design (cf. Algorithm 6): It randomly samples bounds for each feature independently. Thus, each feature excludes a certain fraction of data objects from the subgroup. The more features are used in the subgroup description, the smaller the expected number of data objects in the subgroup becomes. Since the number of subgroup members is one factor in WRAcc (cf. Equation 1), quality naturally decreases for smaller subgroups.

Figures 4a and 4b also reveal that the heuristic search methods *Beam* and *BI* still yield higher average subgroup quality than the solver-based search *SMT* due to timeouts, for any feature-cardinality setting. Even excluding the datasets with *SMT* timeouts (cf. Figures 4c and 4d), these two heuristics yield nearly the same average subgroup quality as *SMT* for constrained k and have an advantage on the test set with unconstrained k. *BSD* and *SD-Map* do not beat *Beam* and *BI* either, suffering from their discretized search space. The heuristic *PRIM* exhibits a larger increase of subgroup quality over k than *Beam* and *BI*, thereby narrowing the quality gap to the latter. The baseline *MORS* displays the least effect of k on mean test-set nWRAcc, showing very stable subgroup quality.

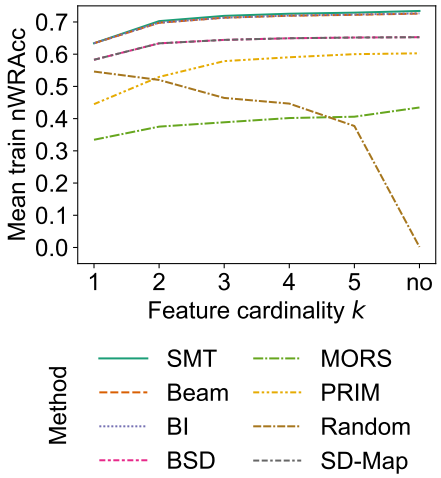
Finally, the results indicate that limiting k reduces overfitting. For example, for *Beam*, the mean difference between training-set and test-set nWRAcc is 0.095 without a feature-cardinality constraint, 0.073 for $k = 3$, and 0.045 for



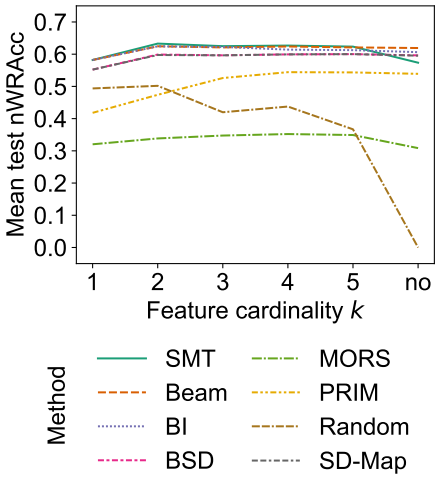
(a) All 27 datasets, training set.



(b) All 27 datasets, test set.



(c) 13 datasets without *SMT* timeouts, training set.



(d) 13 datasets without *SMT* timeouts, test set.

Figure 4: Mean subgroup quality over datasets and cross-validation folds, by subgroup-discovery method and feature cardinality k. Results from the search for original subgroups.

Table 4: Mean runtime (in seconds) over datasets and cross-validation folds, by subgroup-discovery method and feature cardinality k. Results from the search for original subgroups.

k	1	2	3	4	5	no
BI	7.8	11.7	14.2	16.7	18.7	35.0
BSD	0.9	0.9	0.9	2.7	29.5	55.7
Beam	6.8	10.1	12.8	14.6	16.1	30.5
MORS	0.0	0.0	0.0	0.0	0.0	0.0
PRIM	0.1	0.2	0.3	0.3	0.5	1.3
Random	0.6	0.6	0.6	0.7	0.7	0.9
SD-Map	2.3	3.3	9.6	54.0	345.2	367.4
SMT	648.2	911.3	1091.7	1113.4	1117.4	849.0

$k = 1$. The increasing tendency to overfit with larger k explains why mean training-set nWRAcc increases more than mean test-set nWRAcc over k in Figure 4. From the eight subgroup-discovery methods, *BSD* and *SD-Map* show the smallest increase of overfitting over k, *MORS* and *SMT* the largest.

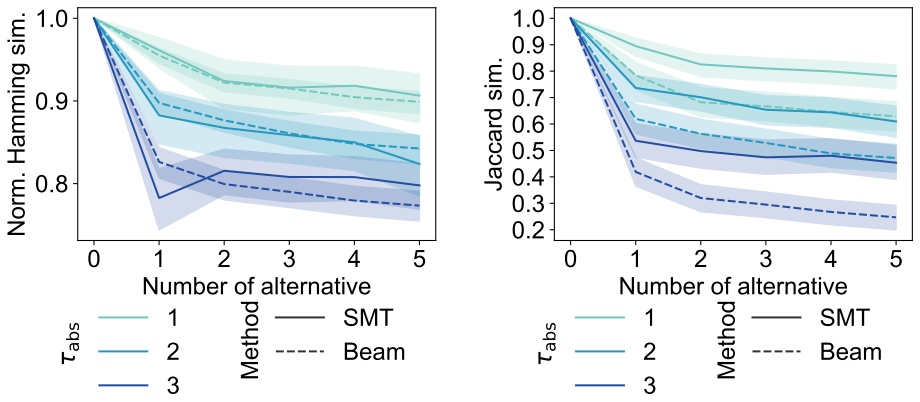
Runtime As Table 4 displays, the three heuristic search methods as well as *BSD* and *SD-Map* become faster the smaller k is. The baseline *Random* shows a similar trend, though less prominent, while *MORS* yields results instantaneously in any case. In contrast, the picture for the solver-based search method *SMT* is less clear. While its average runtime clearly increases from $k = 1$ till $k = 3$, it roughly remains constant for $k \in \{4, 5\}$ and even decreases without a feature-cardinality constraint, only remaining higher than for $k = 1$.

6.4 Alternative Subgroup Descriptions

In this section, we analyze alternative subgroup descriptions for the subgroup-discovery methods *Beam* and *SMT*. Both employ a feature-cardinality threshold of $k = 3$. *SMT* uses its default maximum solver timeout of 2048 s.

Subgroup similarity Figure 5 visualizes the average similarity between the original subgroups and those induced by alternative subgroup descriptions. As one would expect, the subgroup-membership similarity decreases the more alternatives one desires and the more the selected features in subgroup descriptions should differ. Further, the decrease is strongest from the original subgroup, i.e., the zeroth alternative, to the first alternative but smaller beyond. This observation indicates that one may find several alternative subgroup descriptions of comparable similarity to the original.

These trends hold for both similarity measures, i.e., the normalized Hamming similarity we optimize (cf. Equation 12 and Figure 5a) as well as the



(a) Normalized Hamming similarity.

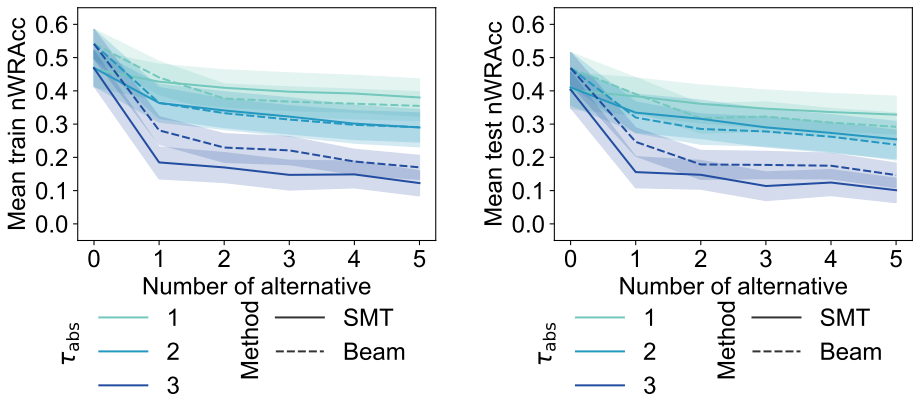
(b) Jaccard similarity.

Figure 5: Mean subgroup similarity of alternative subgroup descriptions to the original subgroup, with 95% confidence intervals based on datasets and cross-validation folds, by subgroup-discovery method, number of alternative, and dissimilarity threshold τ_{abs}.

Jaccard similarity (cf. Equation 13 and Figure 5b). The latter yields lower similarity values than the former since it ignores data objects that are not contained in either compared subgroup. Further, the observed trends exist for the solver-based search method *SMT* as well as the heuristic search method *Beam*. *SMT* yields clearly more similar subgroups than *Beam* for the Jaccard similarity, while the normalized Hamming similarity does not show a clear winner.

Subgroup quality The average subgroup quality of alternative subgroup descriptions (cf. Figure 6) shows similar trends as subgroup similarity (cf. Figure 5). In particular, quality decreases over the dissimilarity threshold τ_{abs} and over the number of alternatives a, with the largest decrease to the first alternative. For the highest dissimilarity threshold $\tau_{\text{abs}} = 3$, *Beam* consistently yields higher average quality than *SMT* for the original subgroup and each alternative. In contrast, the other two values of the dissimilarity threshold do not clearly favor either subgroup-discovery method. The observed trends on the test set (cf. Figure 6b) are very similar to those on the training set (cf. Figure 6a). For both subgroup-discovery methods, overfitting, as measured by the training-test difference in nWRAcc, is lower for the alternative subgroup descriptions than for the original subgroups. This phenomenon may result from the alternative subgroup descriptions not directly optimizing subgroup quality.

Runtime Table 5 displays the average runtime for searching original and alternative subgroup descriptions. The search for alternatives is faster for both analyzed search methods, i.e., *Beam* and *SMT*. As for the original subgroups, *Beam* search for alternatives is one to two orders of magnitude faster than the



(a) Training set.

(b) Test set.

Figure 6: Mean subgroup quality of alternative subgroup descriptions, with 95% confidence intervals based on datasets and cross-validation folds, by subgroup-discovery method, number of alternative, and dissimilarity threshold τ_{abs}.

solver-based *SMT* search. For *Beam*, runtime tends to decrease over the number of alternatives, while *SMT* shows a less clear behavior. In particular, its runtime increases over alternatives for $\tau_{\text{abs}} \in \{1, 2\}$, i.e., settings that allow reusing features from previous subgroup descriptions. In contrast, runtime decreases over alternatives for $\tau_{\text{abs}} = k = 3$, which forbids reusing any feature selected before. Finally, the number of *SMT* tasks finished within the solver timeout behaves similarly to the runtime. In particular, there are more finished tasks when searching for alternative subgroup descriptions than for original subgroups.

6.5 Summary

Unconstrained subgroup discovery (cf. Section 6.1) We recommend using the heuristic search methods *Beam* and *BI*, which were overall best. These two methods from the literature yielded close-to-optimal quality while being significantly faster than our exhaustive, solver-based search method *SMT*. Additionally, heuristic solutions were less prone to overfitting than optimal solutions from exhaustive search. These results highlight the heuristics as serious competitors for any exhaustive search method from the literature, not only our method *SMT*. Further, *Beam* and *BI* beat the quality of the exhaustive competitors *BSD* and *SD-Map*, which required discretizing numeric features beforehand. The heuristic search method *PRIM* was faster than *Beam* and *BI* but yielded lower subgroup quality. Our novel baseline *MORS* provided instantaneous, non-trivial lower bounds for subgroup quality.

Solver timeouts (cf. Section 6.2) Setting larger solver timeouts showed a decreasing marginal utility regarding the number of finished *SMT* tasks and

Table 5: Mean runtime (in seconds) over datasets and cross-validation folds, by subgroup-discovery method, dissimilarity threshold τ_{abs}, and number of alternative. Results from the search for alternative subgroup descriptions.

Method	τ_{abs}	Number of alternative					
		0	1	2	3	4	5
Beam	1	12.8	8.0	7.6	7.3	7.3	7.3
	2	12.8	7.7	7.4	7.2	7.0	6.8
	3	12.8	5.8	5.1	4.7	4.1	3.5
SMT	1	1091.7	166.0	221.5	239.6	258.1	277.9
	2	1105.2	377.5	463.5	537.5	599.4	658.3
	3	1107.4	869.1	670.8	597.6	588.1	557.6

subgroup quality, i.e., most gains occurred within the first few seconds or dozens of seconds. About half the *SMT* tasks that finished at all finished in under a minute. However, the average subgroup quality with this solver timeout was lower than for heuristic search methods with even lower runtime.

Feature-cardinality constraints (cf. Section 6.3) Using more features in subgroup descriptions showed a decreasing marginal utility regarding subgroup quality. For most subgroup-discovery methods, subgroups with as few as $k = 2$ features in their description often yielded already a similar quality as unconstrained subgroups, i.e., using all features. Further, feature-cardinality constraints reduced overfitting and sped up the heuristic search methods as well as *BSD* and *SD-Map*. These results speak for using small feature sets in subgroup descriptions, which may also benefit interpretability for users.

Alternative subgroup descriptions (cf. Section 6.4) The heuristic *Beam* was one to two orders of magnitude faster than solver-based *SMT* when searching for alternative subgroup descriptions, while both search methods found alternatives faster than original subgroups. The quality and similarity of alternative subgroup descriptions strongly depended on two user parameters, i.e., the number of alternatives a and the dissimilarity threshold on feature selection τ_{abs}. The difference in quality and similarity between the original and the first alternative was higher than among the first few alternatives. In particular, there may be several promising alternatives from which users may choose one.

7 Related Work

In this section, we review related work. Next to the literature on subgroup discovery (cf. Section 7.1), we also discuss relevant work from the adjacent field of feature selection (cf. Section 7.2) and other related areas (cf. Section 7.3).

7.1 Subgroup Discovery

In this section, we present related work from the field of subgroup discovery. First, we discuss algorithmic search methods (cf. Section 7.1.1) as well as white-box formulations (cf. Section 7.1.2) for this problem. Second, we cover constrained subgroup discovery in general (cf. Section 7.1.3) and for the two constraint types we focus on, i.e., feature-cardinality constraints (cf. Section 7.1.4) and alternative subgroup descriptions (cf. Section 7.1.5).

7.1.1 Algorithmic Search Methods

The terms 'subgroup' and 'subgroup discovery' have different meanings in different communities, and similar concepts have been reinvented over time. Our problem definition (Section 2.1) builds on related work in data mining; see [5, 52, 53, 113] for broad surveys of subgroup discovery in this field. There are also recent works independent from this research [4, 68, 98, 106]. These works are only loosely related to ours since their problem definition differs in the optimization objective, type of subgroup description, and employed constraints.

Nearly all existing subgroup-discovery methods are algorithmic. In particular, there are heuristic search methods, like PRIM [38] and Best Interval [82], as well as exhaustive search methods, like SD-Map [6, 8], MergeSD [46], and BSD [71, 73]. To the best of our knowledge, optimizing subgroup discovery with a general-purpose SMT solver is a novel concept. There are a few other white-box formulations of particular variants of subgroup discovery, which differ from our work in several aspects, as we discuss next.

7.1.2 White-Box Formulations

Maximum box problem [34] formulates an integer program for the Maximum Box problem, which is about finding a hyperrectangle containing as many positive data objects as possible but no negative data objects, i.e., no false positives. This problem is an intermediate between subgroup discovery (cf. Definition 2), which allows false positives and false negatives, and perfect-subgroup discovery (cf. Definition 6), which allows neither. Also, the problem defines a kind of inverse scenario to minimal-optimal-recall-subgroup discovery (cf. Definition 4), which is about finding a subgroup with as *few negative* data objects as possible but *all positive* data objects. While the latter problem is in $\mathcal{P}$ (cf. Proposition 1), [34] proves $\mathcal{NP}$-hardness of the Maximum Box problem by reduction from the Maximum Independent Set [112] problem. In their evaluation, the authors only use a customized branch-and-bound algorithm but neither solver-based nor heuristic search methods. Further, they consider neither feature-cardinality constraints nor alternative descriptions.

Maximum α-pattern problem [23] investigates the Maximum α-Pattern problem. This problem is similar to the Maximum Box problem but involves a binary dataset and requires a user-selected data object α to be a subgroup

member. Again, cardinality constraints or alternative descriptions are not considered. The authors formulate two integer programs as well as two heuristics. They evaluate their approaches, but no existing subgroup-discovery methods, on generated and benchmark datasets, comparing subgroup quality as well as runtime. Similar to us, they find that heuristics may reach a subgroup quality similar to a solver-based search with orders of magnitude less runtime.

Box search problem [80] proposes two integer-programming formulations for the Box Search problem, which is about finding a hyperrectangle that optimizes the sum of the target variable of all contained data objects. In particular, the target variable is continuous rather than binary, and the chosen objective differs from ours. Further, there are no constraints on feature cardinality or for alternative descriptions. In their evaluation, the authors compare solver-based search to multiple versions of a new branch-and-bound approach but not heuristic search methods. They use synthetically generated datasets rather than typical machine-learning benchmark datasets.

Discriminative itemset mining problem Besides three related problems, [62] formulates Closed Discriminative Itemset Mining and Relevant Subgroup Discovery with the constraint-specification language *Essence* [39]. Both these formulations are close to frequent itemset mining, where items correspond to binary features and itemsets to subgroup descriptions. There is no optimization objective but constraints on the frequency of itemsets and their relevance, excluding dominated solutions. The authors propose a transformation of their problem specification to enable using standard propositional-satisfiability (SAT) or constraint-programming (CP) solvers. Their evaluation exclusively analyzes solver runtime rather than subgroup quality and does not compare against heuristic search methods. Finally, they do not search for alternative descriptions. Instead of feature-cardinality constraints, they randomly generate costs and values for items and corresponding bounds for itemsets.

[48] also provides a constraint-programming formulation of Discriminative Itemset Mining. The authors compare two configurations of a constraint solver against existing itemset-mining algorithms. They evaluate runtime but not quality, as they enumerate all non-redundant itemsets, and do not include constraints for feature cardinality or alternative descriptions.

7.1.3 Constrained Subgroup Discovery

Section 4.2 has already discussed various constraint types in subgroup discovery. Typically, constraints are not formulated declaratively for solver-based optimization but integrated into algorithmic search methods. [10] expresses domain knowledge with the logic programming language Prolog, creating a knowledge base composed of facts and rules. However, the authors do not use a solver to optimize subgroup discovery. In the following, we discuss particular related work for the two constraint types we analyze in detail.

7.1.4 Feature-Cardinality Constraints

Formulation Feature cardinality is a common constraint type [86] and a well-known metric for subgroup complexity [52, 53, 113]. However, our SMT formulation of this constraint type is novel. [77] formulates a quadratic program to select non-redundant features for subgroups, but this is only a subroutine within an algorithmic search for subgroups. Also, the authors define a continuous optimization problem with real-valued feature weights as decision variables, while feature selection within our SMT formulation is discrete (cf. Equation 9).

Empirical studies While several articles on subgroup discovery use a feature-cardinality constraint in their experiments [3, 82, 66, 69, 70], there is a lack of studies that analyze the impact of different feature-cardinality thresholds on different subgroup-discovery methods broadly and systematically. [38] analyzes the subgroup quality over eliminating a different number of redundant features as a post-processing step, but only for PRIM and one dataset. [73] evaluates different search depths for their algorithm BSD but only regarding runtime and number of subgroups after quality-based pruning, not subgroup quality. [101] compares multiple search depths for their algorithm SSD++, which returns a list of multiple subgroups, regarding an information-theoretic quality measure. [52] compares five subgroup-discovery methods with categorical datasets and also evaluates feature cardinality but does not systematically constrain the latter to different values. [86] evaluates three subgroup-discovery methods, including a beam search and an exhaustive search. The authors use feature-cardinality constraints with $k \in \{1, 2, 3, 4\}$ but mainly focus their evaluation on comparing strategies for handling numeric data. Also, they only use six classification datasets, five of them with at most ten numeric features, while we employ more and higher-dimensional datasets. Additionally, we compare subgroup discovery with feature-cardinality constraints to an unconstrained setting.

7.1.5 Alternative Subgroup Descriptions

To the best of our knowledge, alternative subgroup descriptions in the sense of this article are a novel concept. In particular, we aim to maximize the set similarity of contained data objects relative to an original subgroup while using a different subgroup description (cf. Definition 9). In contrast, there are various existing approaches striving for alternatives in the sense of diverse or non-redundant sets of subgroups, which aim to minimize rather than maximize the overlap of contained data objects [5] (cf. Section 4.2). In the following, we discuss approaches that focus on subgroup descriptions.

Description-based diverse subgroup set selection [69] introduces six strategies to foster diversity while searching for multiple subgroups simultaneously. Besides strategies assessing the subgroup members and the information-theoretic compression achieved by subgroups, two strategies refer to subgroup

descriptions. The first excludes subgroup descriptions that have the same quality and differ in only one condition from an existing subgroup description. The second uses a global upper bound on how often a feature may be selected in a set of subgroup descriptions rather than controlling pairwise dissimilarity. Both these strategies give users less control over the overlap of subgroup descriptions than our dissimilarity parameter τ does. Further, [69] targets at simultaneous beam search, optimizing subgroup quality and using the diversity strategies only to prune certain solution candidates. In contrast, we search for alternative descriptions sequentially, optimize similarity to the original subgroup, and consider a solved-based search method in addition to heuristic search.

Diverse top-k characteristics lists [78] introduces the notion of *diverse top-k characteristic lists*, which is a set of lists, each containing multiple patterns, e.g., subgroups. Within each list, the subgroups should be alternative to each other in terms of data objects contained. Between lists, subgroup descriptions should be diverse. However, the latter goal is implemented with a very simple notion of diversity, i.e., exactly the same subgroup description must not appear in two lists, but any other overlap is allowed.

Equivalent subgroup descriptions of minimal length [22] introduces the notion of *equivalent subgroup descriptions of minimal length*, which is stricter than our notion of alternative subgroup descriptions. In particular, the former descriptions need to cover exactly the same set of data objects, like our notion of perfect alternative subgroup descriptions (cf. Definition 10), instead of maximizing similarity. Further, the original feature set should be minimized, i.e., a subset be found, instead of using a different feature set subject to a dissimilarity constraint. The authors prove $\mathcal{NP}$-hardness and propose two algorithms for their problem but do not pursue a solver-based search. We adapt their hardness proof to the perfect-subgroup-discovery problem with a feature-cardinality constraint (cf. Proposition 7), based on which we derive further proofs for problems with feature-cardinality constraints and alternative subgroup descriptions.

Redescription mining Redescription mining aims to find pairs or sets of descriptions that cover exactly or approximately the same data objects [40, 102]. Our notion of alternative subgroup descriptions pursues a similar goal. However, we search for alternative descriptions sequentially instead of simultaneously. Also, our original subgroup description usually optimizes subgroup quality, while redescription mining has no target variable, i.e., is unsupervised [102]. Further, redescription mining works with different dissimilarity criteria than we do, e.g., having features pre-partitioned into non-overlapping sets [40, 41, 87] or requiring only one arbitrary part of the description to differ [97]. In contrast, we allow users to control the overlap between feature sets with the parameter τ. Also, the language for redescriptions may be more complex than for subgroups, e.g., also involve logical negation ($\neg$) and disjunction ($\vee$) [40, 41], while subgroup descriptions only use the logical AND ($\wedge$) over features. Finally, most

existing approaches for redescription mining are algorithmic rather than using white-box optimization [40, 87], though [49] provides a constraint-programming formulation of this problem and other pattern-set mining problems. Several formulations and parametrizations of the redescription-mining problem are $\mathcal{NP}$-hard; see [87] for a detailed analysis.

7.2 Feature Selection

Both constraint types we analyze, i.e., feature-cardinality constraints and alternative subgroup descriptions, relate to the features used in the subgroup description. In the field of feature selection [50, 76], constraints are a topic as well [12, 15, 16]. While limiting feature-set cardinality is very common in feature-selection methods, [12, 15] are unique since they propose a white-box formulation of alternative feature selection. Similar to Equation 16, they use a threshold-based dissimilarity constraint on feature selection, though with a different dissimilarity measure. Besides a sequential search for alternatives, which we use as well, they also analyze simultaneous search.

Despite the similarities, traditional feature selection generally tackles a different optimization problem than subgroup discovery. In particular, the former problem 'only' concerns selecting the features instead of also determining bounds on them. The selected features do not form a prediction model per se but are used in another machine-learning model afterward. For feature selection itself, a notion of feature-set quality serves as the objective function. The latter depends on the feature-selection method but typically assesses features globally, while subgroups describe a particular region in the data.

7.3 Other Fields

Alternatives and constraints in data mining Working with constraints is an area of research for various sub-fields of data mining [44], e.g., for clustering [29, 30], pattern mining [94, 110], explainable AI [32, 43, 91, 109], and automated machine learning [93]. There is work in the other direction as well, i.e., using machine-learning techniques in constraint solving [100]. Finding alternative or diverse solutions is also a concern in various fields, e.g., clustering [17], subspace search [37], and explainable-AI paradigms like counterfactuals [47], criticisms [60], and semifactuals [1].

White-box classification There are white-box formulations for various types of classification models [55]. E.g., there are formulations in propositional logic (SAT) for optimal decision trees, decision sets, and decision lists [92, 108, 115]. Similar to subgroup descriptions, these three model types also use conjunctions of conditions to form decision rules. Creating sparse models to reduce model complexity, as we do with feature-cardinality constraints, is an issue for such models as well [115]. However, these model types use multiple rules to classify data globally, while subgroup discovery employs one rule to describe an inter-

esting region. Further, some of these particular white-box formulations target at perfect predictions rather than optimizing prediction quality.

Counterfactual explanations Searching for counterfactual explanations is an explainable-AI paradigm that targets at data objects with feature values as similar as possible to a given data object but with a different prediction of a given classifier [47]. Thus, counterfactuals provide alternative explanations on the local level, i.e., for individual data objects. In contrast, alternative subgroup descriptions aim to reproduce subgroup membership globally, striving for a similar prediction but a different feature selection. Approaches yielding multiple counterfactuals often foster diversity, e.g., by extending the optimization objective [91] or introducing constraints [56, 88, 105]. However, only some approaches have a user-friendly parameter to actively control the diversity of solutions. In particular, [88] offers a dissimilarity threshold comparable to our parameter τ for alternative subgroup descriptions.

8 Conclusions and Future Work

In this section, we recap our article (cf. Section 8.1) and propose directions for future work (cf. Section 8.2).

8.1 Conclusions

Subgroup-discovery methods constitute an important category of interpretable machine-learning models. In this article, we analyzed constrained subgroup discovery as another step to improve interpretability. First, we formalized subgroup discovery as an SMT optimization problem. This formulation supports a variety of user constraints and enables a solver-based search for subgroups. In particular, we studied two constraint types, i.e., limiting the number of features used in subgroups and searching for alternative subgroup descriptions. For the latter constraint type, we let users control the number of alternatives and a dissimilarity threshold. We showed how to integrate these constraint types into our SMT formulation as well as existing heuristic search methods for subgroup discovery. Further, we proved $\mathcal{NP}$-hardness of the optimization problem with constraints. Finally, we evaluated heuristic and solver-based search with 27 binary-classification datasets. In particular, we analyzed four experimental scenarios: unconstrained subgroup discovery, our two constraint types, and timeouts for solver-based search. Section 6.5 summarized key results.

8.2 Future Work

Datasets Our evaluation used over two dozen generic benchmark datasets (cf. Section 5.5). While such an evaluation shows general trends, the impact of constraints naturally depends on the dataset. Thus, our results may not transfer to each particular scenario. This caveat calls for domain-specific case studies.

In such studies, one could also interpret alternative subgroup descriptions qualitatively, i.e., from the domain perspective.

Constraint types We formalized, analyzed, and evaluated two constraint types, i.e., feature-cardinality constraints (cf. Sections 4.3 and 6.3) and alternative subgroup descriptions (cf. Sections 4.4 and 6.4). While our solver-based approach was not convincing for these two, it retains the conceptual advantage that constraints can be added and combined declaratively instead of needing to integrate individual constraint types into the search procedure. The two particular constraint types studied in this article were antimonotonic, which eased their integration into heuristic search methods. As mentioned in Section 4.2, there are further constraint types one could investigate, e.g., domain-specific constraints, secondary objectives, or alternatives in the sense of covering different data objects rather than covering the same data objects differently.

For alternative subgroup descriptions, one could analyze other dissimilarities, e.g., symmetric ones rather than the asymmetric deselection dissimilarity we used (cf. Equation 14). While the SMT encoding of subgroup discovery is relatively flexible regarding dissimilarities, integrating them into heuristic search methods may be challenging, e.g., if the dissimilarity is not antimonotonic.

Formalization In the solver-based search for subgroups, we used an SMT encoding (cf. Section 4.1) and one particular solver. Different white-box encodings or solvers may speed up the search and lead to fewer timeouts, potentially improving the subgroup quality. We already proposed MILP and MaxSAT encodings (cf. Appendices A.1.2 and A.1.3), though without evaluation.

In our article, two assumptions for subgroup discovery were numerical features and a binary target (cf. Section 2.1). One could adapt the SMT encoding to multi-valued categorical features (cf. Appendix A.1.1) and continuous targets.

Time complexity We established $\mathcal{NP}$-hardness for subgroup discovery with a feature-cardinality constraint. In particular, we tackled the search problem for perfect subgroups (cf. Proposition 7) and the general optimization problem (cf. Proposition 8). While our proof for the latter reduced from the former, one could try to explicitly prove hardness for optimizing imperfect subgroups. Further, while we presented a polynomial-time algorithm for finding perfect subgroups without constraints (cf. Proposition 2), we did not analyze the unconstrained optimization problem of subgroup discovery (cf. Definition 2).

We also showed $\mathcal{NP}$-hardness for finding alternative subgroup descriptions. Again, we tackled the search problem for perfect alternatives first (cf. Propositions 11 and 12) before reducing to the general optimization problem later (cf. Proposition 13). One could try to explicitly prove hardness for optimizing imperfect alternatives. Also, our proofs focused on scenarios where all originally selected features must not be selected in the alternative subgroup description, i.e., a specific value of the dissimilarity threshold τ. One could analyze scenarios with overlapping feature sets explicitly. Additionally, our proofs assumed a

feature-cardinality constraint when searching alternative subgroup descriptions. One could examine scenarios without this constraint type.

For parameterized complexity, we established membership in the relatively broad complexity class $\mathcal{XP}$ for the unconstrained scenario, feature-cardinality constraints, and alternative subgroup descriptions (cf. Propositions 5, 6, and 10). One may attempt to tighten these results.

Finally, while we described how one can integrate feature-cardinality constraints and alternative subgroup descriptions into heuristic search methods (cf. Sections 4.3.3 and 4.4.3), we did not provide quality guarantees relative to the exact optimum. In that regard, one could seek an approximation complexity result, e.g., membership in the complexity class $\mathcal{APX}$, as established for the problem of finding equivalent subgroup descriptions of minimal length [22].

A Appendix

In this section, we provide supplementary materials. Appendix A.1 describes further problem encodings of subgroup discovery, complementing Section 4.1. Appendix A.2 contains proofs for propositions from Section 4. Appendix A.3 summarizes the runtime experiments with subgroup-discovery packages that led to integrating *BSD* and *SD-Map* into the main experiments (cf. Section 6).

A.1 Further Problem Encodings of Subgroup Discovery

In this section, we provide additional white-box encodings of subgroup discovery beyond the SMT encoding from Section 4.1. First, we describe how to encode categorical features within the SMT formulation (cf. Section A.1.1). Next, we discuss encodings as a mixed integer linear program (cf. Section A.1.2) and a maximum-satisfiability problem (cf. Section A.1.3).

A.1.1 Handling Categorical Features in the SMT Encoding

In general, there are many different options to encode categorical data in machine learning numerically [84]. Similarly, there are also multiple options for considering categorical features in an SMT formulation of subgroup discovery. We present three of them in the following.

Two variables per categorical feature As a straightforward option, one may map all categories, i.e., unique values, of each categorical feature to distinct integers before instantiating the optimization problem. One can directly apply our existing SMT formulation (cf. Equation 8) to such an ordinally encoded dataset, at least technically. In particular, there would be two integer-valued bound variables for each encoded categorical feature. However, the ordering of categories should be semantically meaningful since it influences which categories may jointly be included in the subgroup. In particular, only sets of categories that form contiguous integer ranges in the ordinal encoding may define subgroup

membership. I.e., the subgroup may comprise the encoded categories $\{3, 4, 5\}$, but not only $\{3, 5\}$ since it needs to include all values between a lower and an upper bound. Thus, if there is no meaningful ordering of categories, one should choose a different encoding.

Two variables per categorical feature value One can achieve more flexibility by introducing separate bound variables for each category of a feature rather than only for each feature. This approach corresponds to a one-hot encoding of the dataset, which creates one new binary feature for each category. Thus, the bound variables are effectively binary as well. By default, our SMT encoding uses a logical AND ($\wedge$) over the binary features, i.e., categories. The interpretation of bound values for one binary Feature j is as follows:

(Case 1) $lb_j = ub_j = 1$ means that data objects that assume the corresponding category for Feature j are members of the subgroup. In practice, this case may apply to at most one category of each feature. Otherwise, the AND ($\wedge$) operator would require each data object to assume multiple categories for one feature, which is unsatisfiable. Thus, this encoding cannot directly express that a set of categories is included in the subgroup.

(Case 2) $lb_j = ub_j = 0$ means that data objects that do *not* assume the corresponding category for Feature j are members of the subgroup. I.e., data objects assuming the corresponding category are not subgroup members. Other than Case 1, this case can apply to multiple categories of each feature, i.e., the subgroup may explicitly exclude multiple categories. Further, if one category is actively included in the subgroup (Case 1), then Case-2 bounds on other categories are redundant since they are implied by the former.

(Case 3) $lb_j = 0,\ ub_j = 1$ explicitly deselects a binary feature, i.e., both binary values do not restrict subgroup membership.

(Case 4) $lb_j = 1,\ ub_j = 0$ cannot occur since it violates the bound constraints (cf. Equation 6).

Finally, note that binary features allow us to slightly simplify the subgroup-membership expression (cf. Equation 7). In general, we need to check the lower and upper bound for a feature. However, if a binary feature assumes the value 0 for a data object, checking the upper bound is unnecessary since it is always satisfied. Similarly, if a binary feature assumes the value 1 for a data object, checking the lower bound is unnecessary since it is always satisfied. Both these simplifications assume that the bounds are explicitly defined as binary or at least in $[0, 1]$, which can be enforced with straightforward constraints. Otherwise, the bounds may theoretically be placed outside the feature's range and exclude all data objects, producing an empty subgroup.

One variable per categorical feature value In some scenarios, it does not make sense to include the absence of a category in the subgroup, i.e., to permit $lb_j = ub_j = 0$. In particular, some existing subgroup-discovery methods for categorical data assume that only the presence of categories is interesting [5]. In this case, introducing one instead of two bound variable(s) for each category suffices.

Assume the categorical Feature j has $|c_j| \in \mathbb{N}$ different categories $\{c_j^1, \ldots, c_j^{|c_j|}\}$. Let $cb_j \in \{0,1\}^{|c_j|}$ denote the corresponding bound variables, which denote whether a category is included in the subgroup. The ordering of categories in this vector is arbitrary but fixed.

As a difference to previously described encodings, the subgroup-membership expression (cf. Equation 7) should still use a logical AND ($\wedge$) over features but not over categories belonging to the same feature. Otherwise, the expression would be unsatisfiable since each data object only assumes one category for each feature. Instead, we replace the numeric bound check $(X_{ij} \geq lb_j) \wedge (X_{ij} \leq ub_j)$ for Feature j with the following OR ($\vee$) expression:

$$\bigvee_{l \in \{1,\ldots,|c_j|\}} \left(cb_j^l \wedge \left(X_{ij} = c_j^l \right) \right) \tag{17}$$

Since the equality holds for exactly one category, all conjunctions except one are false, and the expression simplifies to one variable $cb_j^{l'}$, where l' is the index of the category X_{ij}. I.e., for each categorical feature, a data object can only be a subgroup member if the variable belonging to its category is 1.

In general, multiple cb_j^l for Feature j may be 1, representing multiple categories included in the subgroup, which is an advantage over the previous encoding. If all categories are in the subgroup, the feature becomes deselected. Thus, for a categorical Feature j, Equation 9 for feature selection becomes:

$$s_j \leftrightarrow \neg \bigwedge_{l \in \{1,\ldots,|c_j|\}} cb_j^l \tag{18}$$

One can also constrain the number of categories in the subgroup, e.g., to either include one category of Feature j in the subgroup or deselect the feature altogether by including all categories:

$$\left(\left(\sum_{l=1}^{|c_j|} cb_j^l \right) = 1 \right) \vee \left(\left(\sum_{l=1}^{|c_j|} cb_j^l \right) = |c_j| \right) \tag{19}$$

A.1.2 Mixed Integer Linear Programming (MILP)

We start from the SMT formulation and introduce additional variables and constraints to linearize certain logical expressions.

Unconstrained subgroup discovery We can keep all decision variables from the corresponding SMT formulation (cf. Equation 8): the binary variables b_i for subgroup membership and the real-valued variables lb_j and ub_j for the bounds. The bound constraints (cf. Equation 6) remain unchanged as well. Further, we retain the optimization objective, which already is linear in b_i (cf. Equations 4 and 5). However, we need to linearize the logical AND operators ($\wedge$) in the definition of subgroup membership b_i (cf. Equation 7) by introducing auxiliary variables and further constraints. In particular, we supplement

the variables $b \in \{0,1\}^m$ by $b^{\text{lb}} \in \{0,1\}^{m \times n}$ and $b^{\text{ub}} \in \{0,1\}^{m \times n}$. These new binary variables indicate whether a particular data object satisfies the lower or upper bound for a particular feature. Using linearization techniques for constraint satisfaction and AND operators from [90], we obtain the following set of constraints to replace Equation 7:

$$\begin{aligned}
\forall i\ \forall j: \quad & X_{ij} + m_j \cdot b_{ij}^{\text{lb}} \leq lb_j - \varepsilon_j \\
\forall i\ \forall j: \quad & lb_j \leq X_{ij} + M_j \cdot \left(1 - b_{ij}^{\text{lb}}\right) \\
\forall i\ \forall j: \quad & ub_j + m_j \cdot b_{ij}^{\text{ub}} \leq X_{ij} - \varepsilon_j \\
\forall i\ \forall j: \quad & X_{ij} \leq ub_j + M_j \cdot \left(1 - b_{ij}^{\text{ub}}\right) \\
\forall i\ \forall j: \quad & b_i \leq b_{ij}^{\text{lb}} \\
\forall i\ \forall j: \quad & b_i \leq b_{ij}^{\text{ub}} \\
\forall i: \quad & \sum_{j=1}^{n} \left(b_{ij}^{\text{lb}} + b_{ij}^{\text{ub}}\right) \leq b_i + 2n - 1 \\
\text{with indices:} \quad & i \in \{1, \ldots, m\} \\
& j \in \{1, \ldots, n\}
\end{aligned} \tag{20}$$

The first two inequalities ensure that $b_{ij}^{\text{lb}} = 1$ if and only if $lb_j \leq X_{ij}$. The following two inequalities perform a corresponding check for b_{ij}^{ub}. The values $\varepsilon_j \in \mathbb{R}_{>0}$ are small constants that turn strict inequalities into non-strict inequalities since a MILP solver may only be able to handle the latter. One possible choice, which we used in a demo implementation, is sorting all unique feature values and taking the minimum difference between two consecutive values in that order.

The values $M_j \in \mathbb{R}_{>0}$ and $m_j \in \mathbb{R}_{<0}$ are large positive and negative constants, respectively. They allow us to express logical implications between real-valued and binary-valued expressions, compensating the latter's smaller range. One choice for M_j is a value larger than the difference between the feature's minimum and maximum, which can be pre-computed before optimization:

$$\begin{aligned}
\forall j \in \{1, \ldots, n\} \quad & M_j := 2 \cdot \left(\max_{i \in \{1,\ldots,m\}} X_{ij} - \min_{i \in \{1,\ldots,m\}} X_{ij} \right) \\
\forall j \in \{1, \ldots, n\} \quad & m_j := 2 \cdot \left(\min_{i \in \{1,\ldots,m\}} X_{ij} - \max_{i \in \{1,\ldots,m\}} X_{ij} \right)
\end{aligned} \tag{21}$$

In particular, the difference between the subgroup's bounds and arbitrary feature values must be smaller than M_j and larger than m_j, unless the bounds are placed outside the feature's value range. Since the latter does not improve the subgroup's quality in any case, we prevent it with additional constraints on the bound variables lb_j and ub_j:

$$\begin{aligned}
\forall j \in \{1, \ldots, n\} \quad & \min_{i \in \{1,\ldots,m\}} X_{ij} \leq lb_j \quad \leq \max_{i \in \{1,\ldots,m\}} X_{ij} \\
\forall j \in \{1, \ldots, n\} \quad & \min_{i \in \{1,\ldots,m\}} X_{ij} \leq ub_j \quad \leq \max_{i \in \{1,\ldots,m\}} X_{ij}
\end{aligned} \tag{22}$$

Finally, the last three inequalities in Equation 20 tie b_{ij}^{lb} and b_{ij}^{ub} to b_i and linearize the logical AND operators ($\wedge$) from Equation 7. In particular, these constraints ensure that a data object is a member of the subgroup, i.e., $b_i = 1$, if and only if all feature values of the data object observe the lower and upper bounds, i.e., all corresponding $b_{ij}^{\text{lb}} = 1$ and $b_{ij}^{\text{ub}} = 1$.

Feature-cardinality constraints The feature-cardinality constraint of the SMT formulation (cf. Equation 10) already is a linear expression in the feature-selection variables s_j, so we can keep it as-is. However, the constraints defining s_j (cf. Equation 9) contain a logical OR ($\vee$) operator and comparison ($<$) expressions. We linearize these constraints as follows:

$$\begin{aligned}
\forall i\ \forall j: \quad & 1 - b_{ij}^{\text{lb}} \leq s_j^{\text{lb}} \\
\forall i\ \forall j: \quad & 1 - b_{ij}^{\text{ub}} \leq s_j^{\text{ub}} \\
\forall j: \quad & s_j^{\text{lb}} \leq s_j \\
\forall j: \quad & s_j^{\text{ub}} \leq s_j \\
\forall j: \quad & s_j \leq 2m - \sum_{i=1}^{m} \left(b_{ij}^{\text{lb}} + b_{ij}^{\text{ub}} \right) \\
\text{with indices:} \quad & i \in \{1, \ldots, m\} \\
& j \in \{1, \ldots, n\}
\end{aligned} \tag{23}$$

The first four inequalities ensure that a feature is selected, i.e., $s_j = 1$, if any data object's feature value lies outside the subgroup's bounds, i.e., any $b_{ij}^{\text{lb}} = 0$ or $b_{ij}^{\text{ub}} = 0$. The last inequality covers the other direction of the logical equivalence: If a feature is selected, then at least one data object's feature value lies outside the subgroup's bounds.

Alternative subgroup descriptions The objective function for alternative subgroup descriptions in the SMT formulation (cf. Equation 15) is already linear. We only need to replace the logical negation operators ($\neg$):

$$\text{sim}_{\text{nHamm}}(b^{(a)}, b^{(0)}) = \frac{1}{m} \cdot \Big(\sum_{\substack{i \in \{1,\ldots,m\} \\ b_i^{(0)}=1}} b_i^{(a)} + \sum_{\substack{i \in \{1,\ldots,m\} \\ b_i^{(0)}=0}} \left(1 - b_i^{(a)}\right) \Big) \tag{24}$$

The same replacement also applies to the dissimilarity constraints (cf. Equation 16), which now look as follows:

$$\forall l \in \{0, \ldots, a-1\}: \ \text{dis}_{\text{des}}(s^{(a)}, s^{(l)}) = \sum_{\substack{j \in \{1,\ldots,n\} \\ s_j^{(l)}=1}} \left(1 - s_j^{(a)}\right) \geq \min\left(\tau_{\text{abs}},\ k^{(l)}\right) \tag{25}$$

Otherwise, this expression is linear as well, so no further auxiliary variables or constraints are necessary.

Implementation Our published code (cf. Section 5.6) contains a MILP implementation for unconstrained and feature-cardinality-constrained subgroup discovery. We use the package *OR-Tools* [99] with *SCIP* [20] as the optimizer. However, in preliminary experiments, this implementation was (on average) slower than the SMT implementation or yielded worse subgroup quality in the same runtime. Further, it sometimes finished considerably after the prescribed timeout or ran out of memory after consuming dozens of gigabytes. Thus, we stuck to the SMT implementation for our main experiments (cf. Section 5.2).

A.1.3 Maximum Satisfiability (MaxSAT)

Our SMT formulation of subgroup discovery with and without constraints uses a combination of propositional logic and linear arithmetic. However, if all feature values are binary or binarized, i.e., $X \in \{0,1\}^{m \times n}$, we can also define a partial weighted MaxSAT problem [11, 74]. This formulation involves hard constraints in propositional logic and an objective function containing weighted clauses, i.e., OR terms. In our case, it even is a Max One [58] problem since the 'clauses' in the objective are plain binary variables.

Unconstrained subgroup discovery For binary feature values, the bound variables lb_j and ub_j become binary rather than real-valued as well. The subgroup membership variables b_i were binary already (cf. Equation 8). In the hard constraints, all less-or-equal inequalities ($\leq$) become logical implications ($\rightarrow$). Thus, the bound constraints (cf. Equation 6) become:

$$\forall j \in \{1, \ldots, n\} : \; lb_j \rightarrow ub_j \tag{26}$$

I.e., if the lower bound is 1, then the upper bound also needs to be 1; otherwise, the upper bound may be 0 or 1.

The subgroup-membership expressions (cf. Equation 7) turn into:

$$\forall i \in \{1, \ldots, m\} : \; b_i \leftrightarrow \bigwedge_{j \in \{1,\ldots,n\}} ((lb_j \rightarrow X_{ij}) \wedge (X_{ij} \rightarrow ub_j)) \tag{27}$$

Since all values X_{ij} are known, we can remove and simplify terms in the definition of b_i. In particular, if $X_{ij} = 1$, then $lb_j \rightarrow X_{ij}$ is a tautology, which we can remove, and $X_{ij} \rightarrow ub_j$ becomes ub_j. Vice, versa, if $X_{ij} = 0$, then $X_{ij} \rightarrow ub_j$ is a tautology and $lb_j \rightarrow X_{ij}$ becomes $\neg lb_j$.

Further, having determined the bound values, the final subgroup description can be expressed as a plain conjunction of propositional literals, e.g., $b_i \leftrightarrow (X_{i2} \wedge \neg X_{i5} \wedge X_{i6})$. In particular, there are four cases: (1) If $lb_j = 0$ and $ub_j = 1$, then the feature's value does not restrict subgroup membership and therefore does not need to be checked in the final subgroup description. (2) If $lb_j = ub_j = 0$, then only $X_{ij} = 0$ is in the subgroup, i.e., a negative literal becomes part of the final subgroup description. (3) If $lb_j = ub_j = 1$, then only $X_{ij} = 1$ is in the subgroup, i.e., a positive literal becomes part of the final

subgroup description. (4) The combination $lb_j = 1$ and $ub_j = 0$ violates the bound constraints and will therefore not appear in a valid solution.

Finally, the objective function is already a weighted sum of the subgroup-membership variables b_i, which form the soft constraints for the problem. In particular, we can re-formulate Equation 4 as follows:

$$\text{WRAcc} = \frac{1}{m} \cdot \sum_{\substack{i \in \{1,\ldots,m\} \\ y_i = 1}} b_i - \frac{m^+}{m^2} \cdot \sum_{i=1}^{m} b_i \tag{28}$$

Thus, for negative data objects, i.e., with $y_i = 0$, the weight is $-m^+/m^2$. For positive data objects, i.e., with $y_i = 1$, the weight is $(m - m^+)/m^2$. Since m is a constant, we can also multiply with m^2 to obtain integer-valued weights.

Feature-cardinality constraints For binary features, the definition of the feature selection variables s_j (cf. Equation 9), which are binary by default, amounts to:

$$\begin{aligned}
\forall j: \quad & s_j^{\text{lb}} \leftrightarrow \left(lb_j \wedge \neg \left(\bigwedge_{i \in \{1,\ldots,m\}} X_{ij} \right) \right) \\
\forall j: \quad & s_j^{\text{ub}} \leftrightarrow \left(\neg ub_j \wedge \left(\bigvee_{i \in \{1,\ldots,m\}} X_{ij} \right) \right) \\
\forall j: \quad & s_j \leftrightarrow \left(s_j^{\text{lb}} \vee s_j^{\text{ub}} \right) \\
\text{with index:} \quad & j \in \{1, \ldots, n\}
\end{aligned} \tag{29}$$

I.e., a feature is selected regarding its lower bound if the lower bound is set to 1 and at least one feature value is 0, i.e., at least one feature value is excluded from the subgroup. Vice versa, a feature is selected regarding its upper bound if the upper bound is set to 0 and at least one feature value is 1, i.e., at least one feature value is excluded from the subgroup. Since all values X_{ij} are known, we can evaluate the corresponding AND and OR terms before optimization. If a feature is 0 and 1 for at least one data object each, which should usually be the case, Equation 29 becomes a much simpler expression:

$$s_j \leftrightarrow (lb_j \vee \neg ub_j) \tag{30}$$

To transform the actual feature-cardinality constraint (cf. Equation 10), which sums up the variables s_j and compares them to a user-defined k, into propositional logic, we can use a cardinality encoding from the literature [111].

Alternative subgroup descriptions The objective function for alternative subgroup descriptions (cf. Equation 15) already is a weighted sum of the subgroup-membership variables $b_i^{(a)}$. In particular, for negative data objects,

i.e., with $y_i = 0$, the weight of the literal $\neg b_i^{(a)}$ is $1/m$. For positive data objects, i.e., with $y_i = 1$, the weight of the literal $b_i^{(a)}$ is $1/m$. Since m is a constant, we can also use 1 as the weight.

We can encode the dissimilarity constraint on the feature selection (cf. Equation 16) with a cardinality encoding from the literature [111].

Non-binary features While we discussed binary features up to now, we can also encode multi-valued features in a way suitable for a MaxSAT formulation. In Section A.1.1, we already addressed how categorical features may be represented binarily. For numeric features, we can introduce two binary variables for each numeric value: Let the numeric Feature j have $|v_j| \in \mathbb{N}$ distinct values $\{v_j^1, \ldots, v_j^{|v_j|}\}$, with higher superscripts denoting higher values. Next, let $lb_j \in \{0,1\}^{|v_j|}$ and $ub_j \in \{0,1\}^{|v_j|}$ denote the corresponding binary bound variables. I.e., instead of two bound variables per feature, there are two bound variables for each unique feature value now. lb_j^l indicates whether the l-th unique value of Feature j is the lower bound. Vice versa, ub_j^l indicates whether the l-th unique value of Feature j is the upper bound. If this encoding generates too many variables, one may discretize the feature first, e.g., by binning its values and representing each bin by one value, e.g., the bin's mean.

The bound constraints (cf. Equations 6 and 26) take the following form:

$$\begin{aligned} \forall j: \quad & \sum_{l=1}^{|v_j|} lb_j^l = 1 \\ \forall j: \quad & \sum_{l=1}^{|v_j|} ub_j^l = 1 \\ \forall j\ \forall l_1 \in \{1,\ldots,|v_j|\}: \quad & ub_j^{l_1} \rightarrow \bigvee_{l_2 \in \{1,\ldots,l_1\}} lb_j^{l_2} \\ \text{with index:} \quad & j \in \{1,\ldots,n\} \end{aligned} \tag{31}$$

The first two constraints ensure that exactly one value of Feature j is chosen as the lower bound and upper bound, respectively. These constraints can be encoded into propositional logic with a cardinality encoding from the literature [111]. The third constraint enforces that the value chosen as the lower bound is less than or equal to the value chosen as the upper bound. Alternatively, one could also formulate that the value chosen as the upper bound is greater than or equal to the value chosen as the lower bound.

We formulate the subgroup-membership expressions (cf. Equations 7 and 27) as follows:

$$\forall i \in \{1,\ldots,m\}: \ b_i \leftrightarrow \bigwedge_{j \in \{1,\ldots,n\}} \Big(\Big(\bigvee_{\substack{l \in \{1,\ldots,\bar{l}\} \\ X_{ij} = v_{\bar{l}}}} lb_j^l \Big) \wedge \Big(\bigvee_{\substack{l \in \{\bar{l},\ldots,|v_j|\} \\ X_{ij} = v_{\bar{l}}}} ub_j^l \Big) \Big) \tag{32}$$

In particular, for a data object to be a subgroup member, each feature's lower bound needs to be lower or equal to the actual value X_{ij}, while the upper bound

needs to be higher or equal. For the binary lower-bound variables lb_j^l, this means that any of the bound variables representing values lower or equal to X_{ij} needs to be 1; vice versa for the upper bounds.

Finally, for feature-cardinality constraints, we define the feature-selection variables s_j (cf. Equations 9 and 29) as follows:

$$\forall j \in \{1, \ldots, n\} : \; s_j \leftrightarrow \left(\neg lb_j^1 \vee \neg ub_j^{|v_j|} \right) \tag{33}$$

In particular, we check whether the lower bound is not the minimum or the upper bound is not the maximum value of that feature, which indicates whether the bounds exclude at least one data object from the subgroup or not. The actual feature-cardinality constraint (cf. Equation 10) does not need to be specifically adapted for non-binary features in MaxSAT. The same goes for the definition of alternative subgroup descriptions (cf. Equations 15 and Equation 16), which only uses the original binary decision variables by default.

A.2 Proofs

In this section, we provide proofs for propositions from Section 4, particularly for the complexity results for subgroup discovery with a feature-cardinality constraint and for searching alternative subgroup descriptions.

A.2.1 Proof of Proposition 7

Proof. Let an arbitrary problem instance I of the decision problem Set Covering [57] be given. I consists of a set of elements $E = \{e_1, \ldots, e_m\}$, a set of sets $\mathbb{S} = \{S_1, \ldots, S_n\}$ with $E = \bigcup_{S \in \mathbb{S}} S$, and a cardinality $k \in \mathbb{N}$. The decision problem Set Covering asks whether a subset $\mathbb{C} \subseteq \mathbb{S}$ exists with $|\mathbb{C}| \leq k$ and $E = \bigcup_{S \in \mathbb{C}} S$, i.e., a subset of $\mathbb{S}$ which contains (= covers) each element from E in at least one set and consist of at most k sets.

We transform I into a problem instance I' of the perfect-subgroup-discovery problem (cf. Definition 6) with a feature-cardinality constraint (cf. Definition 8). To this end, we define a binary dataset $X \in \{0,1\}^{(m+1)\times n}$, prediction target $y \in \{0,1\}^{m+1}$, and retain the number of sets $k \in \mathbb{N}$ as feature cardinality k. In particular, data objects represent elements from E, and features represent sets from $\mathbb{S}$. I.e., X_{ij} denotes $e_i \in S_j$, i.e., membership of Element i in Set j. The additional index $i = m + 1$ represents a *dummy element* that is not part of any set, so all its feature values X_{ij} are set to 0. Further, we define the prediction target $y \in \{0,1\}^{m+1}$ as $y_{m+1} = 1$ and $y_i = 0$ for all other indices $i \in \{1, \ldots, m\}$. This prediction target represents whether an element should *not* be covered by the set of sets $\mathbb{C} \subseteq \mathbb{S}$. In particular, all actual elements from E should be covered but not the new dummy element. This 'inverted' definition of the prediction target stems from the different nature of set covers and subgroup descriptions: Set covers include elements from selected sets, with the empty cover $\mathbb{C} = \emptyset$ containing no elements. There is a logical OR ($\vee$) respectively set union over the selected sets. In contrast, subgroup descriptions exclude data objects based

on bounds for their selected features, with the unrestricted subgroup containing all data objects. There is a logical AND ($\wedge$) over the features' bounds.

A perfect subgroup (cf. Definition 5) exactly replicates the prediction target as subgroup membership. Here, it only contains the data object representing the dummy element but no data objects representing actual elements. Further, as all feature values of this dummy data object are 0, the subgroup description only consists of the bounds $lb_j = ub_j = 0$ for selected features and $lb_j = 0 < 1 = ub_j$ for unselected features. Therefore, the data objects described by the selected features represent elements not contained in any of the selected sets, which only applies to the dummy element. Vice versa, all remaining data objects represent elements that are part of at least one selected set, which applies to all actual elements from E. Further, the feature-cardinality constraint (cf. Definition 8) ensures that at most k features are selected, which means that at most k sets are selected. Thus, if the feature-cardinality constraint is satisfied in the perfect subgroup, the selected features represent sets forming a valid set cover $\mathbb{C}$.

In contrast, if no feature set of the desired size k can describe a perfect subgroup, then at least one data object with prediction target $y_i = 0$ has to be part of the subgroup. Thus, at least one element is not contained in any set forming the set cover, so no valid set cover of size k exists.

Overall, a solution to the instance I' of the perfect-subgroup discovery problem (cf. Definition 6) with a feature-cardinality constraint (cf. Definition 8) also solves the instance I of the decision problem Set Covering [57] with negligible computational overhead. In particular, an efficient solution algorithm for the former would also efficiently solve the latter. However, since the latter problem is $\mathcal{NP}$-hard [57], the former is as well. To be more precise, the perfect-subgroup-discovery problem with a feature-cardinality constraint resides in the complexity class $\mathcal{NP}$ and therefore is $\mathcal{NP}$-complete. In particular, checking a solution induces a polynomial cost of $O(m \cdot n)$, requiring one pass over the dataset to determine subgroup membership and feature selection. □

This proof is an adaptation of the proof of [22] for minimizing the feature cardinality of a given subgroup description. The latter proof reduces from the optimization problem Minimum Set Cover, while we use the decision problem Set Covering since perfect-subgroup discovery (cf. Definition 6) is not an optimization problem. Further, we replace the notion of a given subgroup description [22] with the notion of a perfect subgroup. Also, we employ inequalities with lower and upper bounds in the subgroup description, while [22] uses 'feature=value' conditions. However, this difference is irrelevant for binary datasets, where selected features have $lb_j = ub_j$ bounds and thereby implicitly select a feature value instead of a range. The hardness result naturally extends to real-valued datasets, which generalize binary datasets.

Note that the hardness reduction does not work for the special case $k = n$. For Set Covering, this case allows all sets to be selected, which leads to a trivial solution since the complete set of sets $\mathbb{S}$ contains all elements from E by definition. Vice versa, being able to use all features in the subgroup description

leads to the unconstrained problem of perfect-subgroup discovery (cf. Definition 6), which admits a polynomial-time solution (cf. Proposition 2).

A.2.2 Proof of Proposition 8

Proof. Let an arbitrary problem instance I of the perfect-subgroup-discovery problem (cf. Definition 6) with a feature-cardinality constraint (cf. Definition 8) be given. We transform I into a problem instance I' of the subgroup-discovery problem (cf. Definition 2) with the same constraint. In particular, we define the objective as optimizing a subgroup-quality function $Q(lb, ub, X, y)$ rather than searching for a perfect subgroup (cf. Definition 5) that may or may not exist. The other inputs of the problem instance (X, y, and k) remain the same.

Based on the assumption we made on $Q(lb, ub, X, y)$ in Proposition 8, the optimal solution for I' is a perfect subgroup if the latter exists. Thus, if the optimal subgroup for I' is not perfect, then a perfect subgroup does not exist at all. Checking whether a subgroup is perfect entails a cost of $O(m \cdot n)$, i.e., computing subgroup membership and checking for false positives and false negatives. Overall, an algorithm for subgroup discovery (cf. Definition 2) with a feature-cardinality constraint (cf. Definition 8) solves perfect-subgroup discovery (cf. Definition 6) with the same constraint with negligible overhead. Since the latter problem is $\mathcal{NP}$-complete (cf. Proposition 7) and the former resides in the complexity class $\mathcal{NP}$, the former is $\mathcal{NP}$-complete as well. □

As an alternative proof, one could reduce from the optimization problem Maximum Coverage [27] instead of the search problem of perfect-subgroup discovery (cf. Definition 6) with a feature-cardinality constraint (cf. Definition 8). This proof idea is strongly related to the proof for Proposition 7 (cf. Section A.2.1), which reduces from the decision problem Set Covering [57] to perfect-subgroup discovery (cf. Definition 6) with a feature-cardinality constraint (cf. Definition 8). In contrast to Set Covering, the $k \in \mathbb{N}$ selected subsets in Maximum Coverage need not cover all elements but should cover as many elements as possible. In the terminology of subgroup discovery, the latter objective corresponds to a particular notion of subgroup quality: maximizing the number of true negatives or minimizing the number of false positives, i.e., excluding as many negative data objects from the subgroup as possible. We introduced this problem as minimal-optimal-recall-subgroup discovery (cf. Definition 4), which resides in $\mathcal{P}$ without a feature-cardinality constraint (cf. Proposition 1) due to the baseline *MORS* (cf. Algorithm 5). When equipping *MORS* with feature-cardinality constraints (cf. Section 4.3.4), existing heuristics for the Maximum Coverage problem may provide approximation guarantees.

However, minimizing the number of false positives is a simpler objective than WRAcc (cf. Equation 1), which we focus on in this article. Our proof approach chosen above is more general regarding the notion of subgroup quality but more narrow in the sense that it reduces from a search problem, assuming a particular value of the objective function, instead of an optimization problem.

A.2.3 Proof of Proposition 11

Proof. Let an arbitrary problem instance I of the perfect-subgroup-discovery problem (cf. Definition 6) with a feature-cardinality constraint (cf. Definition 8) be given. We transform I into a problem instance I' of the perfect-alternative-subgroup-description-discovery problem (cf. Definition 11) with the same constraint. In particular, we retain the feature-cardinality threshold $k \in \mathbb{N}$. We will slightly modify the dataset $X \in \mathbb{R}^{m \times n}$, as explained later.

Based on the assumptions we made in Proposition 11, we define the original subgroup for I' to be perfect (cf. Definition 5), i.e., having subgroup membership $b^{(0)} = y$. Also, we choose the dissimilarity threshold $\tau \in \mathbb{R}_{\geq 0}$ high enough that the alternative subgroup description may not select any features that were selected in the original subgroup description. This choice of τ depends on the choice of the dissimilarity measure $\mathrm{dis}(\cdot)$ for feature-selection vectors. E.g., we can choose the deselection dissimilarity used in our article (cf. Equation 14) and $\tau_{\mathrm{abs}} = k$. Note that we do not even need to explicitly define the actual feature selection for the original subgroup description since we must not select these features in the alternative subgroup description anyway. For the sake of completeness, we can define dataset $X' \in \mathbb{R}^{m \times (n+k)}$ of problem instance I' as dataset $X \in \mathbb{R}^{m \times n}$ of problem instance I with k extra *perfect features* added. In particular, we define the Features $n+1, \ldots, n+k$ to be identical to the binary prediction target y. Choosing the bounds $lb_j = ub_j = 1$ on any of these extra features produces the desired original subgroup membership $b^{(0)} = y$. We further assume that all extra features were selected in the original subgroup description but none of the actual features from X was, i.e., $\forall j \in \{1, \ldots, n\} : \; s_j^{(0)} = 0$ and $\forall j \in \{n+1, \ldots, n+k\} : \; s_j^{(0)} = 1$.

A solution for problem instance I' is also a solution for problem instance I. In particular, the perfect alternative subgroup description (cf. Definition 10) defines a perfect subgroup since it perfectly replicates the original subgroup membership, which constitutes a perfect subgroup. I.e., $b^{(a)} = b^{(0)} = y$. Due to the dissimilarity constraint, the alternative subgroup description only selects features from dataset X, not those newly added to create X'. Finally, both I and I' use a feature-cardinality constraint with threshold k. Thus, if a perfect alternative subgroup description for I' exists, it also solves I. If it does not exist, then there is also no other perfect subgroup for I.

Thus, an efficient solution algorithm for the perfect-alternative-subgroup-description-discovery problem (cf. Definition 11) with a feature-cardinality constraint (cf. Definition 8) would also efficiently solve perfect-subgroup discovery (cf. Definition 6) with the same constraint. However, we already established that the latter problem is $\mathcal{NP}$-complete (cf. Proposition 7). Further, evaluating a solution for the former problem entails a polynomial cost of $O(m \cdot n)$ for checking subgroup membership, the bound constraints, the feature-cardinality constraint, and the dissimilarity constraint, placing the problem in complexity class $\mathcal{NP}$. Thus, perfect-alternative-subgroup-description discovery (cf. Definition 11) with a feature-cardinality constraint (cf. Definition 8) is $\mathcal{NP}$-complete. □

A.2.4 Proof of Proposition 12

Proof. Let an arbitrary problem instance I of the perfect-alternative-subgroup-description-discovery problem (cf. Definition 11) with a feature-cardinality constraint (cf. Definition 8) and a perfect original subgroup (cf. Definition 5) be given. We transform I into a problem instance I' of the same problem but with an imperfect original subgroup. In particular, we retain all inputs of the problem as-is except defining dataset $X' \in \mathbb{R}^{(m+1)\times n}$ of problem instance I' as dataset $X \in \mathbb{R}^{m\times n}$ of problem instance I plus an additional *imperfect data object*. This special data object has the label $y_{m+1} = 0$ but exactly the same feature values as an arbitrary existing data object $X_{i\cdot}$ with $y_i = 1$. In particular, such a data object makes it impossible to find a perfect subgroup. However, we assume this data object to be a member of the original subgroup, i.e., $b^{(0)}_{m+1} = 1$, while subgroup membership of all other data objects corresponds to their prediction target, i.e., $\forall i \in \{1, \ldots, m\} : \; b^{(0)}_i = y_i$.

If there is a solution for problem instance I', we can easily transform it to a solution for I. In particular, since the solution is a perfect alternative subgroup description (cf. Definition 10), it replicates $b^{(0)}$, i.e., assigns all positive data objects of I to the alternative subgroup and places all negative data objects of I outside the subgroup. The additional imperfect data object is also a member of the alternative subgroup in I' but does not exist in I. Thus, the solution is a perfect subgroup for I. On the other hand, if no solution for problem instance I' exists, then there is also no solution for I.

Overall, an efficient solution algorithm for the problem of perfect-alternative-subgroup-description discovery (cf. Definition 11) with a feature-cardinality constraint (cf. Definition 8) and an imperfect original subgroup (cf. Definition 5) could also be used to efficiently solve this problem for a perfect original subgroup. However, we proved that the latter problem is $\mathcal{NP}$-complete (cf. Proposition 11), making the former, which resides in $\mathcal{NP}$ as well, also $\mathcal{NP}$-complete. □

A.2.5 Proof of Proposition 13

The following proof is similar to the proof of Proposition 8 (cf. Section A.2.2), which reduced the search problem of perfect-subgroup discovery with a feature-cardinality constraint (cf. Definitions 6 and 8) to the optimization problem of subgroup discovery (cf. Definition 2) with the same constraint.

Proof. Let an arbitrary problem instance I of the perfect-alternative-subgroup-description-discovery problem (cf. Definition 11) with a feature-cardinality constraint (cf. Definition 8) be given. We transform I into a problem instance I' of the alternative-subgroup-description-discovery problem (cf. Definition 9) with the same constraint. In particular, we define the objective as optimizing the subgroup-membership similarity $\mathrm{sim}(\cdot)$ rather than asking for a perfect alternative subgroup description (cf. Definition 10) that may or may not exist. The other inputs of the problem instance remain the same.

Based on the assumption we made on $\mathrm{sim}(\cdot)$ in Proposition 13, the optimal solution for I' is a perfect alternative subgroup description if the latter exists.

Table 6: Datasets used in our competitor-runtime experiments. m denotes the number of data objects and n the number of features.

Dataset	m	n
backache	180	32
horse_colic	368	22
ionosphere	351	34
iris	150	4
spect	267	22
spectf	349	44

Thus, if the optimal alternative subgroup description for I' is not a perfect alternative, then a perfect alternative subgroup description does not exist. Overall, an algorithm for alternative-subgroup-description discovery (cf. Definition 9) with a feature-cardinality constraint (cf. Definition 8) solves perfect-alternative-subgroup-description discovery (cf. Definition 11) with the same constraint with negligible overhead. Since the latter problem is $\mathcal{NP}$-complete (cf. Propositions 11 and 12) and the former resides in $\mathcal{NP}$, the former is $\mathcal{NP}$-complete. □

A.3 Competitor-Runtime Experiments

In this section, we describe the runtime experiments we conducted to find exhaustive subgroup-discovery methods as competitors for our solver-based approach *SMT* in the main experiments (cf. Section 5.2). Section A.3.1 explains the experimental design, and Section A.3.2 presents the results.

A.3.1 Experimental Design

Goal The competitor-runtime experiments aim to find subgroup-discovery methods that are suitable competitors for solver-based search in our main experimental pipeline. An ideal competitor should satisfy several criteria to enable a fair comparison:

(1) Support optimizing WRAcc [65] as the subgroup-quality metric.
(2) Support setting a feature-cardinality threshold.
(3) Not have additional constraints, e.g., on the number of subgroup members.
(4) Support a continuous search space where subgroup descriptions define intervals on features (cf. Definition 1).
(5) Be reasonably fast.

Note that these criteria do not require the competitors to yield a certain subgroup quality, which we would study in the main experiments anyway. Instead, (1)-(4) ensure that the competitors address the same optimization problem as we do, and (5) concerns the experiments' feasibility.

Subgroup-discovery methods We consider competitors from four Python packages for subgroup discovery, i.e., *pysubdisc*, *pysubgroup* [72], *sd4py* [54], and *subgroups* [79]. All four packages offer exhaustive subgroup-discovery methods that support WRAcc as the optimization objective. Such methods are particularly interesting competitors for our solver-based search method *SMT*. Additionally, *pysubdisc*, *pysubgroup*, and *sd4py* provide beam search as a heuristic search method. For comparison, we also include *Beam* and *SMT* from our main experiments (cf. Section 5.2). Overall, we evaluate the following 17 subgroup-discovery methods in our competitor-runtime experiments:

- *csd* (our package): *Beam*, *SMT*
- *pysubdisc*: *Beam*, *BestFirst*, *BreadthFirst*, *DepthFirst*
- *pysubgroup*: *Apriori*, *Beam*, *DepthFirst*, *GPGrowth*
- *sd4py*: *Beam*, *BSD*, *SDMap*
- *subgroups*: *BSD*, *SDMap*, *SDMapStar*, *VLSD*

Note that we harmonized the method names between the packages, i.e., the actual function names in each package may differ slightly.

The methods from *sd4py* and *subgroups* require discretizing numeric features. To this end, we create up to 50 bins per feature, with fewer bins if a feature has fewer unique values. *sd4py* has a built-in equal-width discretization, while we manually employ equal-frequency discretization for *subgroups*.

The exhaustive search methods from *subgroups* do not support a feature-cardinality threshold, so we run them without it.

Finally, we generally leave the hyperparameters of all subgroup-discovery methods at their defaults except to adapt them to our main experiments. For example, we always set WRAcc as the objective and use a beam width of $w = 10$ for all beam-search methods.

Experimental task We focus on finding competitors with a reasonable runtime. To this end, each experimental task in the competitor-runtime experiments combines one subgroup-discovery method with one cross-validation fold of a dataset (in five-fold cross-validation). Within each experimental task, we sequentially measure the runtime for a feature-cardinality threshold of $k \in \{1, 2, 3, 4, 5\}$. In particular, we exclude the unconstrained setting, which may take significantly longer. We run these experiments separately for each dataset and grant each task 9 hours of runtime to process these five cardinality settings. This runtime limit is considerably higher than the maximum solver timeout of 2048 s in our main experiments, even though the latter timeout applies to each cardinality setting individually instead of jointly.

Datasets To accompany potentially slow subgroup-discovery methods, we employ smaller datasets than in our main experiments. In particular, we use five of the smallest datasets from our main experiments, all with $m < 500$ data objects and $n < 50$ features, plus the even smaller *iris* dataset with $m = 150$ and $n = 4$. Table 6 lists these datasets. For comparison: The maxima in our main experiments are considerably higher, i.e., $m = 9822$ and $n = 168$ (cf. Table 1).

Table 7: Which subgroup-discovery methods finished their experimental task (feature-cardinality thresholds $k \in \{1, 2, 3, 4, 5\}$ evaluated sequentially) within 9 hours on which dataset?

Method	back.	horse.	iono.	iris	spect	spectf
csd.Beam	✓	✓	✓	✓	✓	✓
csd.SMT	✓	✓	✓	✓	✓	
pysubdisc.Beam	✓	✓	✓	✓	✓	✓
pysubdisc.BestFirst	✓	✓		✓	✓	
pysubdisc.BreadthFirst	✓	✓		✓	✓	
pysubdisc.DepthFirst	✓	✓		✓	✓	
pysubgroup.Apriori				✓	✓	
pysubgroup.Beam	✓	✓	✓	✓	✓	✓
pysubgroup.DepthFirst				✓	✓	
pysubgroup.GPGrowth				✓		
sd4py.BSD	✓	✓	✓	✓	✓	✓
sd4py.Beam	✓	✓	✓	✓	✓	✓
sd4py.SDMap	✓	✓	✓	✓	✓	✓
subgroups.BSD	✓	✓		✓	✓	
subgroups.SDMap				✓		
subgroups.SDMapStar				✓		
subgroups.VLSD				✓		

Implementation We implemented a dedicated experimental pipeline for the competitor-runtime experiments. This pipeline parallelizes over the experimental tasks. The code and data are available together with the main experiments (cf. Section 5.6). We used the same hardware but Python 3.9 instead of Python 3.8 to support recent versions of all subgroup-discovery packages.

A.3.2 Evaluation

Timeouts Table 7 displays which experimental tasks finished within the 9-hour timeout. Generally, each subgroup-discovery method finished either on all or no cross-validation folds of a dataset. All methods finished on the *iris* dataset, but fewer methods on the five other datasets. From the exhaustive search methods, i.e., without beam search, only *sd4py.BSD* and *sd4py.SDMap* finished on each of the six small datasets.

Runtime For a more detailed view, Table 8 displays the exact runtimes for the dataset *spect*, which had the second-most methods finishing within the 9-hour timeout. This table clearly shows an exponential runtime increase over the feature-cardinality threshold k for the exhaustive search methods from the packages *pysubdisc*, *pysubgroup*, and *subgroups*, while *sd4py* exhibits considerably smaller and less varying runtimes.

Table 8: Runtime (in seconds) on the dataset *spect*, averaged over five cross-validation folds. Missing subgroup-discovery methods compared to Table 7 did not finish their experimental task within 9 hours.

Method	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
csd.Beam	0.03	0.05	0.06	0.08	0.09
csd.SMT	3.58	3.92	3.63	3.49	3.50
pysubdisc.Beam	2.40	0.20	0.17	0.32	0.26
pysubdisc.BestFirst	1.61	0.27	2.12	26.39	829.56
pysubdisc.BreadthFirst	2.35	0.33	1.79	6.23	47.98
pysubdisc.DepthFirst	1.96	0.29	1.95	27.60	948.02
pysubgroup.Apriori	0.02	0.07	0.81	8.93	70.20
pysubgroup.Beam	0.02	0.12	0.23	0.35	0.50
pysubgroup.DepthFirst	0.02	0.23	3.65	45.06	412.13
sd4py.BSD	0.36	0.66	0.07	0.10	0.18
sd4py.Beam	0.45	0.85	0.26	0.27	0.19
sd4py.SDMap	0.45	0.89	0.29	0.55	0.49
subgroups.BSD	0.15	0.86	6.40	36.90	166.24

Conclusions Overall, these competitor-runtime experiments show that only the exhaustive search methods from *sd4py* can be expected to have a reasonable runtime in our main experiments. In particular, the latter involve larger datasets and also an unrestricted feature-cardinality threshold, which corresponds to $k = n \geq 20$ (number of features in the dataset, which is at least 20). Thus, we integrated *sd4py*'s *BSD* [73] and *SDMap* [8] into our main experiments (cf. Section 5.2). Both methods retain the potential weakness that they require discretizing numeric features, which reduces their search space. Thus, they may yield a lower quality than an exhaustive search on the full space but also be faster. In our main experiments, we rely on *sd4py*'s built-in equal-width discretization but optimize subgroup quality over ten different numbers of bins.

References

[1] André Artelt and Barbara Hammer. ""Even if ..." – Diverse Semifactual Explanations of Reject". In: *Proc. SSCI*. Singapore, Singapore, 2022, pp. 854–859. DOI: 10.1109/SSCI51031.2022.10022139.

[2] Vadim Arzamasov and Klemens Böhm. "REDS: Rule Extraction for Discovering Scenarios". In: *Proc. SIGMOD*. Virtual Event, China, 2021, pp. 115–128. DOI: 10.1145/3448016.3457301.

[3] Vadim Arzamasov, Benjamin Jochum, and Klemens Böhm. *Pedagogical Rule Extraction to Learn Interpretable Models — an Empirical Study*. arXiv:2112.13285v2 [cs.LG]. 2022. DOI: 10.48550/arXiv.2112.13285.

[4] Abolfazl Asudeh, Zhongjun Jin, and H. V. Jagadish. “Assessing and Remedying Coverage for a Given Dataset”. In: *Proc. ICDE*. Macao, China, 2019, pp. 554–565. DOI: 10.1109/ICDE.2019.00056.

[5] Martin Atzmueller. “Subgroup discovery”. In: *WIREs Data Min. Knowl. Disc.* 5.1 (2015), pp. 35–49. DOI: 10.1002/widm.1144.

[6] Martin Atzmueller and Florian Lemmerich. “Fast Subgroup Discovery for Continuous Target Concepts”. In: *Proc. ISMIS*. Prague, Czech Republic, 2009, pp. 35–44. DOI: 10.1007/978-3-642-04125-9_7.

[7] Martin Atzmueller and Frank Puppe. “A Methodological View on Knowledge-Intensive Subgroup Discovery”. In: *Proc. EKAW*. Poděbrady, Czech Republic, 2006, pp. 318–325. DOI: 10.1007/11891451_28.

[8] Martin Atzmueller and Frank Puppe. “SD-Map – A Fast Algorithm for Exhaustive Subgroup Discovery”. In: *Proc. PKDD*. Berlin, Germany, 2006, pp. 6–17. DOI: 10.1007/11871637_6.

[9] Martin Atzmueller, Frank Puppe, and Hans-Peter Buscher. “Exploiting Background Knowledge for Knowledge-Intensive Subgroup Discovery”. In: *Proc. IJCAI*. Edinburgh, United Kingdom, 2005, pp. 647–652. URL: https://www.ijcai.org/Proceedings/05/Papers/1217.pdf.

[10] Martin Atzmueller and Dietmar Seipel. “Using Declarative Specifications of Domain Knowledge for Descriptive Data Mining”. In: *Proc. INAP*. Würzburg, Germany, 2007, pp. 149–164. DOI: 10.1007/978-3-642-00675-3_10.

[11] Fahiem Bacchus, Matti Järvisalo, and Ruben Martins. “Maximum Satisfiability”. In: *Handbook of Satisfiability*. 2nd ed. IOS Press, 2021. Chap. 24, pp. 929–991. DOI: 10.3233/FAIA201008.

[12] Jakob Bach. *Finding Optimal Diverse Feature Sets with Alternative Feature Selection*. arXiv:2307.11607v3 [cs.LG]. 2025. DOI: 10.48550/arXiv.2307.11607.

[13] Jakob Bach. “Leveraging Constraints for User-Centric Feature Selection”. PhD thesis. Karlsruhe Institute of Technology (KIT), 2025. DOI: 10.5445/IR/1000178649.

[14] Jakob Bach. “Subgroup Discovery with Small and Alternative Feature Sets”. In: *Proc. SIGMOD*. Berlin, Germany, 2025.

[15] Jakob Bach and Klemens Böhm. “Alternative feature selection with user control”. In: *Int. J. Data Sci. Anal.* (2024). DOI: 10.1007/s41060-024-00527-8.

[16] Jakob Bach et al. “An Empirical Evaluation of Constrained Feature Selection”. In: *SN Comput. Sci.* 3.6 (2022). DOI: 10.1007/s42979-022-01338-z.

[17] James Bailey. “Alternative Clustering Analysis: A Review”. In: *Data Clustering: Algorithms and Applications*. 1st ed. CRC Press, 2014. Chap. 21, pp. 535–550. DOI: 10.1201/9781315373515.

[18] Clark Barrett and Cesare Tinelli. “Satisfiability Modulo Theories”. In: *Handbook of Model Checking*. 1st ed. Springer, 2018. Chap. 11, pp. 305–343. DOI: 10.1007/978-3-319-10575-8_11.

[19] Adnene Belfodil et al. “FSSD - A Fast and Efficient Algorithm for Subgroup Set Discovery”. In: *Proc. DSAA*. Washington, DC, USA, 2019, pp. 91–99. DOI: 10.1109/DSAA.2019.00023.

[20] Ksenia Bestuzheva et al. *The SCIP Optimization Suite 8.0*. Tech. rep. Zuse Institute Berlin, 2021. URL: http://nbn-resolving.de/urn:nbn:de:0297-zib-85309.

[21] Nikolaj Bjørner, Anh-Dung Phan, and Lars Fleckenstein. "νZ - An Optimizing SMT Solver". In: *Proc. TACAS*. London, United Kingdom, 2015, pp. 194–199. DOI: 10.1007/978-3-662-46681-0_14.

[22] Mario Boley and Henrik Grosskreutz. "Non-redundant Subgroup Discovery Using a Closure System". In: *Proc. ECML PKDD*. Bled, Slovenia, 2009, pp. 179–194. DOI: 10.1007/978-3-642-04180-8_29.

[23] Tibérius O. Bonates, Peter L. Hammer, and Alexander Kogan. "Maximum patterns in datasets". In: *Discrete Appl. Math.* 156.6 (2008), pp. 846–861. DOI: 10.1016/j.dam.2007.06.004.

[24] Guillaume Bosc et al. "Anytime discovery of a diverse set of patterns with Monte Carlo tree search". In: *Data Min. Knowl. Disc.* 32.3 (2018), pp. 604–650. DOI: 10.1007/s10618-017-0547-5.

[25] Cristóbal J. Carmona, María J. del Jesus, and Francisco Herrera. "A unifying analysis for the supervised descriptive rule discovery via the weighted relative accuracy". In: *Knowledge-Based Syst.* 139 (2018), pp. 89–100. DOI: 10.1016/j.knosys.2017.10.015.

[26] Diogo V. Carvalho, Eduardo M. Pereira, and Jaime S. Cardoso. "Machine Learning Interpretability: A Survey on Methods and Metrics". In: *Electronics* 8.8 (2019). DOI: 10.3390/electronics8080832.

[27] Chandra Chekuri and Amit Kumar. "Maximum Coverage Problem with Group Budget Constraints and Applications". In: *Proc. APPROX*. Cambridge, MA, USA, 2004, pp. 72–83. DOI: 10.1007/978-3-540-27821-4_7.

[28] Seung-Seok Choi, Sung-Hyuk Cha, and Charles C. Tappert. "A Survey of Binary Similarity and Distance Measures". In: *J. Syst. Cybern. Inf.* 8.1 (2010), pp. 43–48. URL: http://www.iiisci.org/Journal/pdv/sci/pdfs/GS315JG.pdf.

[29] Thi-Bich-Hanh Dao, Khanh-Chuong Duong, and Christel Vrain. "A Declarative Framework for Constrained Clustering". In: *Proc. ECML PKDD*. Prague, Czech Republic, 2013, pp. 419–434. DOI: 10.1007/978-3-642-40994-3_27.

[30] Thi-Bich-Hanh Dao and Christel Vrain. "A review on declarative approaches for constrained clustering". In: *Int. J. Approximate Reasoning* 171 (2024). DOI: 10.1016/j.ijar.2024.109135.

[31] Leonardo De Moura and Nikolaj Bjørner. "Z3: An Efficient SMT Solver". In: *Proc. TACAS*. Budapest, Hungary, 2008, pp. 337–340. DOI: 10.1007/978-3-540-78800-3_24.

[32] Daniel Deutch and Nave Frost. "Constraints-based Explanations of Classifications". In: *Proc. ICDE*. Macao, China, 2019, pp. 530–541. DOI: 10.1109/ICDE.2019.00054.

[33] Rodney G. Downey, Michael R. Fellows, and Ulrike Stege. "Parameterized Complexity: A Framework for Systematically Confronting Computational Intractability". In: *Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future*. Štiřín Castle, Czech Republic, 1997, pp. 49–99. DOI: 10.1090/dimacs/049/04.

[34] Jonathan Eckstein et al. “The Maximum Box Problem and its Application to Data Analysis”. In: *Comput. Optim. Appl.* 23.3 (2002), pp. 285–298. DOI: 10.1023/A:1020546910706.

[35] Stefano Ermon, Carla Gomes, and Bart Selman. “Uniform Solution Sampling Using a Constraint Solver As an Oracle”. In: *Proc. UAI*. Catalina Island, CA, USA, 2012, pp. 255–264. URL: https://www.auai.org/uai2012/papers/160.pdf.

[36] Cyril Esnault et al. “Q-Finder: An Algorithm for Credible Subgroup Discovery in Clinical Data Analysis – An Application to the International Diabetes Management Practice Study”. In: *Front. Artif. Intell.* 3 (2020). DOI: 10.3389/frai.2020.559927.

[37] Edouard Fouché, Florian Kalinke, and Klemens Böhm. “Efficient subspace search in data streams”. In: *Inf. Syst.* 97 (2021). DOI: 10.1016/j.is.2020.101705.

[38] Jerome H. Friedman and Nicholas I. Fisher. “Bump hunting in high-dimensional data”. In: *Stat. Comput.* 9.2 (1999), pp. 123–143. DOI: 10.1023/A:1008894516817.

[39] Alan M. Frisch et al. “Essence: A constraint language for specifying combinatorial problems”. In: *Constraints* 13.3 (2008), pp. 268–306. DOI: 10.1007/s10601-008-9047-y.

[40] Esther Galbrun and Pauli Miettinen. *Redescription Mining*. 1st ed. Springer, 2017. DOI: 10.1007/978-3-319-72889-6.

[41] Arianna Gallo, Pauli Miettinen, and Heikki Mannila. “Finding Subgroups having Several Descriptions: Algorithms for Redescription Mining”. In: *Proc. SDM*. Atlanta, GA, USA, 2008, pp. 334–345. DOI: 10.1137/1.9781611972788.30.

[42] Dragan Gamberger and Nada Lavrač. “Expert-Guided Subgroup Discovery: Methodology and Application”. In: *J. Artif. Intell. Res.* 17 (2002), pp. 501–527. DOI: 10.1613/jair.1089.

[43] Niku Gorji and Sasha Rubin. “Sufficient Reasons for Classifier Decisions in the Presence of Domain Constraints”. In: *Proc. AAAI*. Virtual Conference, 2022, pp. 5660–5667. DOI: 10.1609/aaai.v36i5.20507.

[44] Valerio Grossi, Andrea Romei, and Franco Turini. “Survey on using constraints in data mining”. In: *Data Min. Knowl. Disc.* 31.2 (2017), pp. 424–464. DOI: 10.1007/s10618-016-0480-z.

[45] Henrik Großkreutz, Daniel Paurat, and Stefan Rüping. “An Enhanced Relevance Criterion For More Concise Supervised Pattern Discovery”. In: *Proc. KDD*. Beijing, China, 2012, pp. 1442–1450. DOI: 10.1145/2339530.2339756.

[46] Henrik Grosskreutz and Stefan Rüping. “On subgroup discovery in numerical domains”. In: *Data Min. Knowl. Disc.* 19.2 (2009), pp. 210–226. DOI: 10.1007/s10618-009-0136-3.

[47] Riccardo Guidotti. “Counterfactual explanations and how to find them: literature review and benchmarking”. In: *Data Min. Knowl. Disc.* (2022). DOI: 10.1007/s10618-022-00831-6.

[48] Tias Guns, Siegfried Nijssen, and Luc De Raedt. “Itemset mining: A constraint programming perspective”. In: *Artif. Intell.* 175.12-13 (2011), pp. 1951–1983. DOI: 10.1016/j.artint.2011.05.002.

[49] Tias Guns, Siegfried Nijssen, and Luc De Raedt. "k-Pattern Set Mining under Constraints". In: *IEEE Trans. Knowl. Data Eng.* 25.2 (2011), pp. 402–418. DOI: 10.1109/TKDE.2011.204.

[50] Isabelle Guyon and André Elisseeff. "An Introduction to Variable and Feature Selection". In: *J. Mach. Learn. Res.* 3.Mar (2003), pp. 1157–1182. URL: https://www.jmlr.org/papers/v3/guyon03a.html.

[51] Jiawei Han, Jian Pei, and Yiwen Yin. "Mining Frequent Patterns without Candidate Generation". In: *ACM SIGMOD Rec.* 29.2 (2000), pp. 1–12. DOI: 10.1145/335191.335372.

[52] Sumyea Helal. "Subgroup Discovery Algorithms: A Survey and Empirical Evaluation". In: *J. Comput. Sci. Technol.* 31.3 (2016), pp. 561–576. DOI: 10.1007/s11390-016-1647-1.

[53] Franciso Herrera et al. "An overview on subgroup discovery: foundations and applications". In: *Knowl. Inf. Syst.* 29.3 (2011), pp. 495–525. DOI: 10.1007/s10115-010-0356-2.

[54] Dan Hudson and Martin Atzmueller. "Subgroup Discovery with SD4Py". In: *Proc. ECAI Workshops*. Kraków, Poland, 2023, pp. 338–348. DOI: 10.1007/978-3-031-50396-2_19.

[55] Alexey Ignatiev et al. "Reasoning-Based Learning of Interpretable ML Models". In: *Proc. IJCAI*. Montreal, Canada, 2021, pp. 4458–4465. DOI: 10.24963/ijcai.2021/608.

[56] Amir-Hossein Karimi et al. "Model-Agnostic Counterfactual Explanations for Consequential Decisions". In: *Proc. AISTATS*. Online, 2020, pp. 895–905. URL: https://proceedings.mlr.press/v108/karimi20a.html.

[57] Richard M. Karp. "Reducibility among Combinatorial Problems". In: *Complexity of Computer Computations*. 1st ed. Plenum Press, 1972. Chap. 9, pp. 85–103. DOI: 10.1007/978-1-4684-2001-2_9.

[58] Sanjeev Khanna, Madhu Sudan, and David P. Williamson. "A Complete Classification of the Approximability of Maximization Problems Derived from Boolean Constraint Satisfaction". In: *Proc. STOC*. El Paso, TX, USA, 1997, pp. 11–20. DOI: 10.1145/258533.258538.

[59] Christoph Kiefer et al. "Subgroup discovery in structural equation models". In: *Psychol. Methods* (2022). DOI: 10.1037/met0000524.

[60] Been Kim, Rajiv Khanna, and Oluwasanmi Koyejo. "Examples are not Enough, Learn to Criticize! Criticism for Interpretability". In: *Proc. NIPS*. Barcelona, Spain, 2016. URL: https://proceedings.neurips.cc/paper_files/paper/2016/hash/5680522b8e2bb01943234bce7bf84534-Abstract.html.

[61] Mi-Young Kim et al. "A Multi-Component Framework for the Analysis and Design of Explainable Artificial Intelligence". In: *Mach. Learn. Knowl. Extr.* 3.4 (2021), pp. 900–921. DOI: 10.3390/make3040045.

[62] Gökberk Koçak et al. "Exploiting Incomparability in Solution Dominance: Improving General Purpose Constraint-Based Mining". In: *Proc. ECAI*. Santiago de Compostela, Spain, 2020, pp. 331–338. DOI: 10.3233/FAIA200110.

[63] Jan Kwakkel. "The Exploratory Modeling Workbench: An open source toolkit for exploratory modeling, scenario discovery, and (multi-objective) robust decision making". In: *Environ. Modell. Software* 96 (2017), pp. 239–250. DOI: 10.1016/j.envsoft.2017.06.054.

[64] Jan H. Kwakkel and Scott C. Cunningham. "Improving scenario discovery by bagging random boxes". In: *Technol. Forecasting Social Change* 111 (2016), pp. 124–134. DOI: 10.1016/j.techfore.2016.06.014.

[65] Nada Lavrač, Peter Flach, and Blaz Zupan. "Rule Evaluation Measures: A Unifying View". In: *Proc. ILP*. Bled, Slovenia, 1999, pp. 174–185. DOI: 10.1007/3-540-48751-4_17.

[66] Nada Lavrač and Dragan Gamberger. "Relevancy in Constraint-Based Subgroup Discovery". In: *Proc. European Workshop on Inductive Databases and Constraint Based Mining*. Hinterzarten, Germany, 2006, pp. 243–266. DOI: 10.1007/11615576_12.

[67] Nada Lavrač et al. "Subgroup discovery with CN2-SD". In: *J. Mach. Learn. Res.* 5.Feb (2004), pp. 153–188. URL: https://www.jmlr.org/papers/v5/lavrac04a.html.

[68] Connor Lawless et al. "Interpretable Clustering via Multi-Polytope Machines". In: *Proc. AAAI*. Virtual Event, 2022, pp. 7309–7316. DOI: 10.1609/aaai.v36i7.20693.

[69] Matthijs van Leeuwen and Arno Knobbe. "Diverse subgroup set discovery". In: *Data Min. Knowl. Disc.* 25.2 (2012), pp. 208–242. DOI: 10.1007/s10618-012-0273-y.

[70] Matthijs van Leeuwen and Antti Ukkonen. "Discovering Skylines of Subgroup Sets". In: *Proc. ECML PKDD*. Prague, Czech Republic, 2013, pp. 272–287. DOI: 10.1007/978-3-642-40994-3_18.

[71] Florian Lemmerich, Martin Atzmueller, and Frank Puppe. "Fast exhaustive subgroup discovery with numerical target concepts". In: *Data Min. Knowl. Disc.* 30.3 (2016), pp. 711–762. DOI: 10.1007/s10618-015-0436-8.

[72] Florian Lemmerich and Martin Becker. "pysubgroup: Easy-to-Use Subgroup Discovery in Python". In: *Proc. ECML PKDD*. Dublin, Ireland, 2019, pp. 658–662. DOI: 10.1007/978-3-030-10997-4_46.

[73] Florian Lemmerich, Mathias Rohlfs, and Martin Atzmueller. "Fast Discovery of Relevant Subgroup Patterns". In: *Proc. FLAIRS*. Daytona Beach, FL, USA, 2010, pp. 428–433. URL: https://aaai.org/papers/flairs-2010-1262/.

[74] Chu Min Li and Felip Manyà. "MaxSAT, Hard and Soft Constraints". In: *Handbook of Satisfiability*. 2nd ed. IOS Press, 2021. Chap. 23, pp. 903–927. DOI: 10.3233/FAIA201007.

[75] Haobo Li et al. "Subgroup Discovery Points to the Prominent Role of Charge Transfer in Breaking Nitrogen Scaling Relations at Single-Atom Catalysts on VS2". In: *ACS Catal.* 11.13 (2021), pp. 7906–7914. DOI: 10.1021/acscatal.1c01324.

[76] Jundong Li et al. "Feature Selection: A Data Perspective". In: *ACM Comput. Surv.* 50.6 (2017). DOI: 10.1145/3136625.

[77] Rui Li et al. “Efficient redundancy reduced subgroup discovery via quadratic programming”. In: *J. Intell. Inf. Syst.* 44.2 (2015), pp. 271–288. DOI: 10.1007/s10844-013-0284-1.

[78] Antonio Lopez-Martinez-Carrasco et al. “Discovering Diverse Top-K Characteristic Lists”. In: *Proc. IDA*. Louvain-la-Neuve, Belgium, 2023, pp. 262–273. DOI: 10.1007/978-3-031-30047-9_21.

[79] Antonio Lopez-Martinez-Carrasco et al. “Subgroups: A Python library for Subgroup Discovery”. In: *SoftwareX* 28 (2024). DOI: 10.1016/j.softx.2024.101895.

[80] Quentin Louveaux and Sébastien Mathieu. “A combinatorial branch-and-bound algorithm for box search”. In: *Discrete Optim.* 13 (2014), pp. 36–48. DOI: 10.1016/j.disopt.2014.05.001.

[81] Tarcísio Lucas, Renato Vimieiro, and Teresa Ludermir. “SSDP+: A Diverse and More Informative Subgroup Discovery Approach for High Dimensional Data”. In: *Proc. CEC*. Rio de Janeiro, Brazil, 2018. DOI: 10.1109/CEC.2018.8477855.

[82] Michael Mampaey et al. “Efficient Algorithms for Finding Richer Subgroup Descriptions in Numeric and Nominal Data”. In: *Proc. ICDM*. Brussels, Belgium, 2012, pp. 499–508. DOI: 10.1109/ICDM.2012.117.

[83] Romain Mathonat et al. “Anytime Subgroup Discovery in High Dimensional Numerical Data”. In: *Proc. DSAA*. Porto, Portugal, 2021. DOI: 10.1109/DSAA53316.2021.9564223.

[84] Federico Matteucci, Vadim Arzamasov, and Klemens Böhm. “A benchmark of categorical encoders for binary classification”. In: *Proc. NeurIPS*. New Orleans, LA, USA, 2023, pp. 54855–54875. URL: https://proceedings.neurips.cc/paper_files/paper/2023/hash/ac01e21bb14609416760f790dd8966ae-Abstract-Datasets_and_Benchmarks.html.

[85] Marvin Meeng, Wouter Duivesteijn, and Arno Knobbe. “ROCsearch – An ROC-guided Search Strategy for Subgroup Discovery”. In: *Proc. SDM*. Philadelphia, PA, USA, 2014, pp. 704–712. DOI: 10.1137/1.9781611973440.81.

[86] Marvin Meeng and Arno Knobbe. “For real: a thorough look at numeric attributes in subgroup discovery”. In: *Data Min. Knowl. Disc.* 35.1 (2021), pp. 158–212. DOI: 10.1007/s10618-020-00703-x.

[87] Matej Mihelčić and Adrian Satja Kurdija. “On the complexity of redescription mining”. In: *Theor. Comput. Sci.* 944 (2023). DOI: 10.1016/j.tcs.2022.12.023.

[88] Kiarash Mohammadi et al. “Scaling Guarantees for Nearest Counterfactual Explanations”. In: *Proc. AIES*. Virtual Event, USA, 2021, pp. 177–187. DOI: 10.1145/3461702.3462514.

[89] Christoph Molnar, Giuseppe Casalicchio, and Bernd Bischl. “Interpretable Machine Learning – A Brief History, State-of-the-Art and Challenges”. In: *Proc. ECML PKDD Workshops*. Ghent, Belgium, 2020, pp. 417–431. DOI: 10.1007/978-3-030-65965-3_28.

[90] MOSEK ApS. *MOSEK Modeling Cookbook : Mixed integer optimzation*. Accessed: 2022-10-18. 2022. URL: https://docs.mosek.com/modeling-cookbook/mio.html.

[91] Ramaravind K. Mothilal, Amit Sharma, and Chenhao Tan. "Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations". In: *Proc. FAT**. Barcelona, Spain, 2020, pp. 607–617. DOI: 10.1145/3351095.3372850.

[92] Nina Narodytska et al. "Learning Optimal Decision Trees with SAT". In: *Proc. IJCAI*. Stockholm, Sweden, 2018, pp. 1362–1368. DOI: 10.24963/ijcai.2018/189.

[93] Felix Neutatz, Marius Lindauer, and Ziawasch Abedjan. "AutoML in heavily constrained applications". In: *VLDB J.* (2023). DOI: 10.1007/s00778-023-00820-1.

[94] Raymond T. Ng et al. "Exploratory Mining and Pruning Optimizations of Constrained Associations Rules". In: *ACM SIGMOD Rec.* 27.2 (1998), pp. 13–24. DOI: 10.1145/276305.276307.

[95] Robert Nieuwenhuis and Albert Oliveras. "On SAT Modulo Theories and Optimization Problems". In: *Proc. SAT*. Seattle, WA, USA, 2006, pp. 156–169. DOI: 10.1007/11814948_18.

[96] Randal S. Olson et al. "PMLB: a large benchmark suite for machine learning evaluation and comparison". In: *BioData Min.* 10 (2017). DOI: 10.1186/s13040-017-0154-4.

[97] Laxmi Parida and Naren Ramakrishnan. "Redescription Mining: Structure Theory and Algorithms". In: *Proc. AAAI*. Pittsburgh, PA, USA, 2005, pp. 837–844. URL: https://aaai.org/papers/00837-aaai05-132-redescription-mining-structure-theory-and-algorithms/.

[98] Eliana Pastor, Luca de Alfaro, and Elena Baralis. "Looking for Trouble: Analyzing Classifier Behavior via Pattern Divergence". In: *Proc. SIGMOD*. Virtual Event, China, 2021, pp. 1400–1412. DOI: 10.1145/3448016.3457284.

[99] Laurent Perron and Vincent Furnon. *OR-Tools*. Accessed: 2022-10-18. Google, 2022. URL: https://developers.google.com/optimization/.

[100] Andrei Popescu et al. "An overview of machine learning techniques in constraint solving". In: *J. Intell. Inf. Syst.* 58.1 (2022), pp. 91–118. DOI: 10.1007/s10844-021-00666-5.

[101] Hugo M. Proença et al. "Robust subgroup discovery: Discovering subgroup lists using *MDL*". In: *Data Min. Knowl. Disc.* 36.5 (2022), pp. 1885–1970. DOI: 10.1007/s10618-022-00856-x.

[102] Naren Ramakrishnan et al. "Turning CARTwheels: An Alternating Algorithm for Mining Redescriptions". In: *Proc. KDD*. Seattle, WA, USA, 2004, pp. 266–275. DOI: 10.1145/1014052.1014083.

[103] Youcef Remil et al. ""What makes my queries slow?": Subgroup Discovery for SQL Workload Analysis". In: *Proc. ASE*. Melbourne, Australia, 2021, pp. 642–652. DOI: 10.1109/ASE51524.2021.9678915.

[104] Joseph D. Romano et al. *PMLB v1.0: An open source dataset collection for benchmarking machine learning methods*. arXiv:2012.00058v3 [cs.LG]. 2021. DOI: 10.48550/arXiv.2012.00058.

[105] Chris Russell. "Efficient Search for Diverse Coherent Explanations". In: *Proc. FAT**. Atlanta, GA, USA, 2019, pp. 20–28. DOI: 10.1145/3287560.3287569.

[106] Svetlana Sagadeeva and Matthias Boehm. "SliceLine: Fast, Linear-Algebra-based Slice Finding for ML Model Debugging". In: *Proc. SIGMOD*. Virtual Event, China, 2021, pp. 2290–2299. DOI: 10.1145/3448016.3457323.

[107] Martin Scholz. "Sampling-Based Sequential Subgroup Mining". In: *Proc. KDD*. Chicago, IL, USA, 2005, pp. 265–274. DOI: 10.1145/1081870.1081902.

[108] Pouya Shati, Eldan Cohen, and Sheila McIlraith. "SAT-Based Approach for Learning Optimal Decision Trees with Non-Binary Features". In: *Proc. CP*. Montpellier, France (Virtual Conference), 2021. DOI: 10.4230/LIPIcs.CP.2021.50.

[109] Aditya A. Shrotri et al. "Constraint-Driven Explanations for Black-Box ML Models". In: *Proc. AAAI*. Virtual Conference, 2022, pp. 8304–8314. DOI: 10.1609/aaai.v36i8.20805.

[110] Andreia Silva and Cláudia Antunes. "Constrained pattern mining in the new era". In: *Knowl. Inf. Syst.* 47.3 (2016), pp. 489–516. DOI: 10.1007/s10115-015-0860-5.

[111] Carsten Sinz. "Towards an Optimal CNF Encoding of Boolean Cardinality Constraints". In: *Proc. CP*. Sitges, Spain, 2005, pp. 827–831. DOI: 10.1007/11564751_73.

[112] Robert Endre Tarjan and Anthony E. Trojanowski. "Finding a Maximum Independent Set". In: *SIAM J. Comput.* 6.3 (1977), pp. 537–546. DOI: 10.1137/0206038.

[113] Sebastián Ventura and José María Luna. "Subgroup Discovery". In: *Supervised Descriptive Pattern Mining*. 1st ed. Springer, 2018. Chap. 4, pp. 71–98. DOI: 10.1007/978-3-319-98140-6_4.

[114] Danding Wang et al. "Designing Theory-Driven User-Centric Explainable AI". In: *Proc. CHI*. Glasgow, United Kingdom, 2019. DOI: 10.1145/3290605.3300831.

[115] Jinqiang Yu et al. "Learning Optimal Decision Sets and Lists with SAT". In: *J. Artif. Intell. Res.* 72 (2021), pp. 1251–1279. DOI: 10.1613/jair.1.12719.

[116] Cristina Zuheros et al. "Explainable Crowd Decision Making methodology guided by expert natural language opinions based on Sentiment Analysis with Attention-based Deep Learning and Subgroup Discovery". In: *Inf. Fusion* 97 (2023). DOI: 10.1016/j.inffus.2023.101821.