Anytime Pattern Mining in Numerical Datasets with Guarantees

Aimene Belfodil [*,1,2], Adnene Belfodil[*,1], and Mehdi Kaytoue[1,3]

[1] Univ Lyon, INSA Lyon, CNRS, LIRIS UMR 5205, F-69621, LYON, France
[2] Mobile Devices Ingénierie, 100 Avenue Stalingrad, 94800, Villejuif, France
[3] Infologic, 99 avenue de Lyon, 26500 Bourg-Lès-Valence, France
`firstname.surname@insa-lyon.fr`

Abstract. Subgroup discovery is the task of discovering patterns that accurately discriminate a class label from the others. Existing approaches can uncover such patterns either through an exhaustive or an approximate exploration of the pattern search space. However, an exhaustive exploration is generally unfeasible whereas approximate approaches do not provide guarantees bounding the *error* of the best pattern quality nor the exploration progression ("*How far are we of an exhaustive search*"). We design here an algorithm for mining numerical data with three key properties w.r.t. the state of the art: (i) It yields progressively interval patterns whose quality improves over time; (ii) It can be interrupted anytime and always gives a guarantee bounding the *error* on the top pattern quality and (iii) It always bounds a distance to the exhaustive exploration. After reporting experimentations showing the effectiveness of our method, we discuss its generalization to other kinds of patterns.

Keywords: Subgroup discovery, Anytime algorithms, Discretization

1 Introduction

We address the problem of discovering patterns that accurately discriminate one class label from the others in a numerical dataset. Subgroup discovery (SD) [28] is a well established pattern mining framework which strives to find out data regions uncovering such interesting patterns. When it comes to numerical attributes, a pattern is generally a conjunction of restrictions over the attributes, e.g., $50 \leq age < 70 \land smoke\_per\_day \geq 3$. To look for such patterns (namely interval patterns), various approaches are usually implemented. Common techniques perform *a discretization* transforming the numerical attributes to categorical ones in a pre-processing phase before using the wide spectrum of existing mining techniques [2,21,23,4]. This leads, however, to a loss of information even if an exhaustive enumeration is performed on the transformed data [2]. Other approaches explore the whole search space of all restrictions either exhaustively [18,16,7] or heuristically [24,6]. While an exhaustive enumeration is generally unfeasible in large data, the various state-of-the-art algorithms that heuristically

---

explore the search space provide no provable guarantee on how they *approximate* the top quality patterns and on how far they are from an exhaustive search. Recent techniques set up a third and elegant paradigm, that is direct sampling approaches [4,5,15]. Algorithms falling under this category are non-enumerative methods which directly sample solutions from the pattern space. They simulates a distribution which rewards high quality pattern with respect to some interestingness measure. While [4,5] propose a direct two-step sampling procedure dedicated for categorical/boolean datasets, authors in [15] devise an interesting framework which add a third step to handle the specificity of numerical data. The proposed algorithm addresses the discovery of dense neighborhood patterns by defining a new density metric. Nevertheless, it does not consider the discovery of discriminant numerical patterns in labeled numerical datasets. Direct sampling approaches abandons the completeness property and generates only approximate results. In contrast, anytime pattern mining algorithms [6,17] are enumerative methods which exhibits the anytime feature [30], a solution is always available whose quality improves gradually over time and which converges to an exhaustive search if given enough time, hence ensuring completeness. However, to the best of our knowledge, no existing anytime algorithm in SD framework, makes it possible to ensure guarantees on the patterns discriminative power and the remaining distance to an exhaustive search while taking into account the nature of numerical data.

To achieve this goal, we propose a novel anytime algorithm, `RefineAndMine`, tailored for discriminant interval patterns discovery in numerical data. It starts by mining interval patterns in a coarse discretization, followed by successive refinements yielding increasingly finer discretizations highlighting potentially new interesting patterns. Eventually, it performs a full traversal of the underlying search space, if given enough time. Additionally, our method gives two provable guarantees at each iteration (discretization refinement). The first one evaluates how close is the best found pattern so far to the optimal one which could be found in the whole search space. The second one measures how already found patterns are diverse and cover well all the interesting regions in the dataset. This is done by bounding the maximum distance between any pattern of the complete search space to its most similar pattern in the explored search space.

The outline of this paper is as follows. We recall basic definitions in Sec. 2. Next, we state formally the guarantee we want to have in Sec. **??** defining the problem. Afterwards, we introduce in Sec. 3 our mining algorithm. Then, in Sec. 4, we formulate the guarantees it provides. We empirically evaluate the efficiency of `RefineAndMine` in Sec. 5 before to discuss its potential improvements.

2   Preliminaries

Input. A *labeled numerical dataset* $(\mathcal{G}, \mathcal{M})$ is given by a finite set (of objects) $\mathcal{G}$ partitioned into two subsets $\mathcal{G}^+$ and $\mathcal{G}^-$ enclosing respectively positive (target) and negative instances; and a sequence of numerical attributes $\mathcal{M} = (m_i)_{1 \leq i \leq p}$ of size $p = |\mathcal{M}|$. Each *attribute* m_i is an application $m_i : \mathcal{G} \to \mathbb{R}$ that associates

to each object $g \in \mathcal{G}$ a value $m_i(g) \in \mathbb{R}$. We can also see $\mathcal{M}$ as a mapping $\mathcal{M} : \mathcal{G} \to \mathbb{R}^p, g \mapsto (m_i(g))_{1 \leq i \leq p}$. We denote $m_i[\mathcal{G}] = \{m_i(g) \mid g \in \mathcal{G}\}$. Fig. 1 (left table) presents a 2-dimensional labeled numerical dataset and its representation in a the Cartesian plane (filled dots represent positive instances).

Interval patterns and their extents. Subgroup discovery is the task of finding *patterns* in labeled datasets with a *high discriminative power*. When dealing with numerical datasets, we generally consider, for intelligibility, *interval patterns* [18]. An *Interval pattern* is a *conjunction of restrictions over the numerical attributes*; that is a set of conditions $m_i \gtreqless v$ with $\gtreqless \in \{=, \leq, <, \geq, >\}$ and m_i is an attribute. Geometrically speaking, interval patterns are *axis-parallel hyper-rectangles*. Fig. 1 (center-left) depicts the interval pattern (non-hatched rectangle) $c_2 = (1 \leq m_1 \leq 4) \wedge (0 \leq m_2 \leq 3) \triangleq [1,4] \times [0,3]$.

Interval patterns are naturally (partially) ordered thanks to "hyper-rectangle inclusion". We denote the *infinite partially ordered set* (*poset*) of all interval patterns by $(\mathcal{D}, \sqsubseteq)$ where $\sqsubseteq$ (same order used in [18]) denotes the dual order $\supseteq$ of hyper-rectangle inclusion. That is pattern $d_1 \sqsubseteq d_2$ *iff* d_1 encloses d_2 $(d_1 \supseteq d_2)$. It is worth mentioning[4] that $(\mathcal{D}, \sqsubseteq)$ is not only a poset, it actually does form a *complete lattice* [27]. For a subset $S \subseteq \mathcal{D}$, the supremum $\bigsqcup S$ (also called join, i.e. the smallest upper bound) is given by the rectangle intersection. Dually, the infimum $\bigsqcap S$ (meet, i.e the largest lower bound) is given by the smallest hyper-rectangle enclosing all patterns in S. Note that the top (resp. bottom) description, that is the most (resp. least) restrictive one, is denoted by $\top$ (resp. $\bot$) and is given by $\emptyset$ (resp. $\mathbb{R}^p$). Fig. 1 (right) depicts two patterns (hatched) $e_1 = [1,5] \times (1,4]$ and $e_2 = [0,4) \times [2,6]$, their meet (non hatched) $e_1 \sqcap e_2 = [0,5] \times (1,6]$ and their join (black) $e_1 \sqcup e_2 = [1,4) \times [2,4]$.

We associate to each object $g \in \mathcal{G}$, the degenerate hyper-rectangle $\delta(g) = \{\mathcal{M}(g)\} = \bigtimes_{i=1}^{p}[m_i(g), m_i(g)]$. We associate also to each *interval pattern* its *extent*; that is the set of objects covered (*i.e.* enclosed) by it. Formally, there is a function $ext : \mathcal{D} \to \wp(\mathcal{G}), d \mapsto \{g \in \mathcal{G} \mid d \sqsubseteq \delta(g)\} = \{g \in \mathcal{G} \mid \mathcal{M}(g) \in d\}$. It is clear that, if for two patterns $d_1 \sqsubseteq d_2$ (i.e. d_1 encloses d_2) then $ext(d_2) \subseteq ext(d_1)$. We define also the positive (resp. negative) extent as follow: $ext^+(d) = ext(d) \cap \mathcal{G}^+$ (resp. $ext^-(d) = ext(d) \cap \mathcal{G}^-$).

With the mapping $\delta : \mathcal{G} \to \mathcal{D}$ and the complete lattice $(\mathcal{D}, \sqsubseteq)$, we denote by $\mathbb{P} = (\mathcal{G}, (\mathcal{D}, \sqsubseteq), \delta)$ the *interval pattern structure* [18,12].

Measuring the discriminative power of a pattern. Remember that the goal of subgroup discovery is to highlight patterns that *well-discriminate* the positive instances in $\mathcal{G}^+$ from those in $\mathcal{G}^-$. This is done using a *quality measure* $\phi : \mathcal{D} \to \mathbb{R}$. Two atomic measures are generally employed to quantify the quality of a pattern d: the *true positive rate* $tpr : d \to |ext^+(d)|/|\mathcal{G}^+|$ and the *false negative rate* $fpr : d \to |ext^-(d)|/|\mathcal{G}^-|$. While many measures exist in the literature [14,22], we are interested in on what we call *discriminant objective quality measure*. A measure is said to be *objective* or *probability based* [14] if it

---

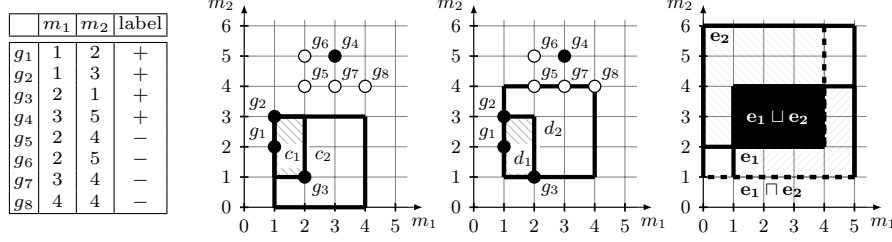[4]This information is necessary for proving the main results of the paper

	m_1	m_2	label
g_1	1	2	+
g_2	1	3	+
g_3	2	1	+
g_4	3	5	+
g_5	2	4	−
g_6	2	5	−
g_7	3	4	−
g_8	4	4	−

Fig. 1: (left to right) (1) a labeled numerical dataset. **(2)** closed c_1 vs non-closed c_2 interval patterns. **(3)** cotp d_1 vs non cotp d_2. **(4)** meet and join of two patterns.

depends only on the number of co-occurrences and non co-occurrences of the pattern and the target label. In other words, those measures depends only on tpr and fpr and potentially other constants (like *dataset size* $|\mathcal{G}|$). Formally, $\exists! \phi^* : [0,1]^2 \to \mathbb{R}$ s.t. $\phi(d) = \phi^*(tpr(d), fpr(d))$. Objective measures depends only on the pattern *extent*. Hence, we will denote interchangeably $\phi(ext(d))$ and $\phi(d)$. An *objective quality measure* ϕ is said to be *discriminant* if its associated measure ϕ^* is *increasing* with tpr (fpr being fixed) and *decreasing* with fpr (tpr being fixed). For instance, $wracc^*(tpr, fpr) = \alpha^+ \cdot \alpha^- \cdot (tpr - fpr)$ or $informedness^*(tpr, fpr) = tpr - fpr$, where $\alpha^+ = |\mathcal{G}^+|/|\mathcal{G}|$, $\alpha^- = |\mathcal{G}^-|/|\mathcal{G}|$ represents respectively positive and negative label prevalence, are discriminant.

Compressing the set of interesting patterns using closure. As we have seen before, discriminant quality measure like $wracc$ and $informedness$ depend only on pattern extent. One common way to reduce the number of outputted patterns is the usage of *closed patterns* [12]. A pattern $c \in \mathcal{D}$ is said to be *(support-)closed* [3] iff $(\forall d \in \mathcal{D}) \, c \sqsubseteq d \Rightarrow ext(d) \subsetneq ext(c)$. That is, any additional restriction on the pattern c will drop at least one object from its extent. Interestingly, since interval patterns poset $(\mathcal{D}, \sqsubseteq)$ is a *complete lattice*, for each pattern $d \in \mathcal{D}$, there exists a unique *closed* pattern $c \in \mathcal{D}$ such that $ext(c) = ext(d)$. Pattern c is called the *closure* of pattern d and is given by $c = int(ext(d))$ ($int \circ ext$ is a closure operator on $(\mathcal{D}, \sqsubseteq)$), where int mapping (called *intent)* associates to each subset of object $A \subseteq \mathcal{G}$, the most restrictive description (i.e. the smallest hyper-rectangle) covering them. The *intent* is defined as follow: $int : \wp(\mathcal{G}) \to \mathcal{D}, A \mapsto \bigsqcap_{g \in A} \delta(g) = \bigtimes_{i=1}^{p} [\min_{g \in A} m_i(g), \max_{g \in A} m_i(g)]$. Note that even if $\mathcal{D}$ is infinite, the set of closed patterns is finite since $\mathcal{G}$ is finite. Fig. 1 (center-left) depicts the closed interval pattern (hatched rectangle) $c_1 = [1,2] \times [1,3]$ which is the closure of $c_2 = [1,4] \times [0,3]$ (non hatched rectangle). The set of all closed patterns is given by $int[\wp(\mathcal{G})]$.

Compressing even more using Relevance theory. Fig. 1 (center-right) depicts two interval patterns, the hatched pattern $d_1 = [1,2] \times [1,3]$ and the non-hatched one $d_2 = [1,4] \times [1,4]$. While both patterns are *closed*, d_1 has certainly better descriminative power than d_2 since they both cover exactly the same positive instances $\{g_1, g_2, g_3\}$; however, d_2 cover more negative instances than d_1. *Relevance theory* formalize this observation and helps us to remove some

clearly uninteresting closed patterns. In a nutshell, a closed pattern $d_1 \in \mathcal{D}$ is said to be *more relevant than* a closed pattern $d_2 \in \mathcal{D}$ iff the positive instances covered by d_2 are covered by d_1 while at the same time the negative instances covered by d_1 are covered by d_2. For ϕ discriminant, if d_1 is more relevant than d_2 then $\phi(d_1) \geq \phi(d_2)$. A closed pattern d is said to be *relevant* iff there is no other closed pattern c that is more relevant than d. Note that if a closed pattern is relevant then it is *closed on the positive* (`cotp` for short). An interval pattern is said to be `cotp` if any smaller interval pattern will at least drop one positive instance (i.e. $d = int(ext^+(d))$). Actually, $int \circ ext^+$ is also a closure operator on $(\mathcal{D}, \sqsubseteq)$. Fig. 1 (center-right) depicts a non `cotp` pattern $d_2 = [1,4] \times [1,4]$ and its closure on the positive $d_1 = int(ext^+(d_2)) = [1,2] \times [1,3]$ which is relevant. Please note that not all `cotp` are *relevant* (see [13] for a full characterization of relevant patterns). The set of all `cotp` patterns is given by $int[\wp(\mathcal{G}^+)]$. We call *relevant (resp. cotp) extent*, any set $A \subseteq \mathcal{G}$ such that $A = ext(d)$ with d is a *relevant (resp. cotp)* pattern. The set of all relevant extents is denoted by $\mathcal{R}$.

3   Anytime Interval Pattern Mining

Discretizations and pattern space. A *discretization* of $\mathbb{R}$ is any *partition* of $\mathbb{R}$ using intervals. In particular, let $C = \{c_i\}_{1 \leq i \leq |C|} \subseteq \mathbb{R}$ be a finite set with $c_i < c_{i+1}$ for $i \in \{1, ..., |C| - 1\}$. Element of C are called *cut points* or *cuts*. We associate to C a *finite discretization* denoted by $dr(C)$ and given by $dr(C) = \{(-\infty, c_1)\} \cup \{[c_i, c_{i+1}) \mid i \in \{1, ..., |C| - 1\}\} \cup \{[c_{|C|}, +\infty)\}$.

Generally speaking, let $p \in \mathbb{N}^*$ and let $C = (C_k)_{1 \leq k \leq p} \in \wp(\mathbb{R})^p$ representing *sets of cut points* associated to each dimension k (i.e. $C_k \subseteq \mathbb{R}$ finite $\forall k \in \{1, ..., p\}$). The partition $dr(C)$ of $\mathbb{R}^p$ is given by: $dr(C) = \prod_{k=1}^{p} dr(C_k)$. Fig. 2 depicts two discretizations. Discretizations are ordered using the natural order between partitions[5]. Moreover, cut-points sets are ordered by $\leq$ as follows: $C^1 \leq C^2 \equiv (\forall k \in \{1, ..., p\}) C_k^1 \subseteq C_k^2$ with $C^i = (C_k^i)_{1 \leq k \leq p}$. Clearly, if $C^1 \leq C^2$ then discretization $dr(C^1)$ *is coarser than* $dr(C^2)$.

Let be the cut-points $C = (C_k)_{1 \leq k \leq p}$. Using the elementary hyper-rectangles (i.e. cells) in the discretization $dr(C)$, one can build a (finite) subset of descriptions $\mathcal{D}_C \subseteq \mathcal{D}$ which is the set of all possible descriptions (hyper-rectangles) that can be built using these cells. Formally: $\mathcal{D}_C = \{\bigsqcap S \mid S \subseteq dr(C)\}$. Note that $\top = \emptyset \in \mathcal{D}_C$ since $\bigsqcap \emptyset = \bigsqcup \mathcal{D} = \top$ by definition. Proposition 1 states an important property on $(\mathcal{D}_C, \sqsubseteq)$ mandatory to demonstrate other results in the paper. *Relevant extents* induced by $\mathcal{D}_C$ are included in the set of relevant extents since $ext[\mathcal{D}_C] \subseteq ext[\mathcal{D}]$. However, the associated relevant pattern (`cotp` in $\mathcal{D}_C$) could are larger than the real one in $\mathcal{D}$.

Proposition 1. $(\mathcal{D}_C, \sqsubseteq)$ is a *finite (complete) sub-lattice* of $(\mathcal{D}, \sqsubseteq)$ that is: $\forall d_1, d_2 \in \mathcal{D}_C : d_1 \sqcup d_2 \in \mathcal{D}_C$ and $d_1 \sqcap d_2 \in \mathcal{D}_C$. Moreover, if $C^1 \leq C^2$ are two cut-points sets, then $(\mathcal{D}_{C^1}, \sqsubseteq)$ is a *(complete) sub-lattice* of $(\mathcal{D}_{C^2}, \sqsubseteq)$.

---

[5]Let E be a set, a partition P_2 of E *is finer than* a partition P_1 (or P_1 *is coarser than* P_2) and we denote $P_1 \leq P_2$ if any subset in P_1 is a subset of a subset in P_2.
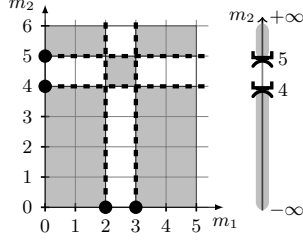
Fig. 2: (left) Discretization $dr((C_1, C_2))$ in $\mathbb{R}^2$ with $C_1 = \{2, 3\}$ and $C_2 = \{4, 5\}$ and **(right)** discretization $dr((C_2))$ in $\mathbb{R}$. Adding a cut point in any C_k will create finer discretization.

Finest discretization for a complete enumeration of relevant extents.
There exist cut points $C \subseteq \wp(\mathbb{R})^p$ such that the space $(\mathcal{D}_C, \sqsubseteq)$ holds all relevant extents (i.e. $ext[\mathcal{D}_C] \supseteq \mathcal{R}$). For instance, if we consider $C = (m_k[\mathcal{G}])_{1 \leq k \leq p}$, the description space $(\mathcal{D}_C, \sqsubseteq)$ holds all relevant extents. However, is there coarser discretization that holds all the relevant extents? The answer is affirmative. One can show that the only interesting cuts are those separating between positive and negative instances (called boundary cut-points by [10]). We call such cuts, *relevant cuts*. They are denoted by $C^{rel} = (C_k^{rel})_{1 \leq k \leq p}$ and we have $ext[\mathcal{D}_{C^{rel}}] \supseteq \mathcal{R}$. Formally, for each dimension k, a value $c \in m_k[\mathcal{G}]$ is a *relevant cut in* C_k^{rel} for attribute m_k iff: $(c \in m_k[\mathcal{G}^+]$ and $prev(c, m_k[\mathcal{G}]) \in m_k[\mathcal{G}^-])$ or $(c \in m_k[\mathcal{G}^-]$ and $prev(c, m_k[\mathcal{G}]) \in m_k[\mathcal{G}^+])$ (see Table 1 for notations). Finding *relevant cuts* C_k^{rel} is of the same complexity of sorting $m_k[\mathcal{G}]$ [10]. In the dataset depicted in Fig. 1, relevant cuts for attribute m_1 are $C_1^{rel} = \{2, 3, 4, 5\} = m_1[\mathcal{G}] \backslash \{1\}$ since 1 does not separate between two distinct classes. Relevant cuts for attribute m_2 are $C_2^{rel} = \{4, 5\} = m_2[\mathcal{G}] \backslash \{1, 2, 3\}$. Fig. 2 (center) depicts the discretization $dr((C_2^{rel}))$.

Anytime enumeration of relevant extents. We design an *anytime* and *interruptible* algorithm devised `RefineAndMine`. This method, presented in Algorithm 1, relies on the enumeration of a chain of discretizations on the data space, from the coarsest to the finest. It begins by searching relevant cuts in pre-processing phase (line 2). Then, it starts by building a *coarse discretization* (line 3) containing a small enough set of relevant cut-points. Once the initial discretization built, `cotp` patterns are mined thanks to `MinIntChange` Algorithm (line 4) [18]. Then as long as the algorithm is not interrupted (or within the computational budget), we add new cut-points (line 6) building finer discretizations. For each added cut-point (line 8), only new interval patterns are searched for (mined descriptions d are new but their extents $ext(d)$ are not necessarily new) . That is `cotp` patterns which left or right bound is *cut* on the considered attribute $attr$ (i.e. $d.I_{attr} \in \{[cut, a), [cut, +\infty), [a, cut), (-\infty, cut) \mid a \in C_{attr}^{cur}\}$). This can be done by a slight modification of `MinIntChange` method. `RefineAndMine` terminates when the set of relevant cuts is exhausted (i.e. $C^{cur} = C^{rel}$) ensuring the *complete* enumeration of relevant extents.

The initial discretization (Line 3) can be done by various strategies (see [29]). A simple, yet efficient, choice is the equal frequency discretization with a fixed number of cuts. Other strategies can be used, e.g. [10]. Adding new

cut-points (Line 6) can also be done in various ways. One strategy is to add a random relevant cut on a random attribute to build the next discretization. Section 4.3 proposes another more elaborated strategy that heuristically guide `RefineAndMine` to rapidly find good quality patterns (observed experimentally).

Algorithm 1: `RefineAndMine`

Input: $(\mathcal{G}, \mathcal{M})$ a numerical datasets with $\{\mathcal{G}^+, \mathcal{G}^-\}$ partition of $\mathcal{G}$

1 procedure `RefineAndMine`()

2   Look for relevant cuts C^{rel}

3   Build an initial set of cut-points $C^{cur} \leq C^{rel}$

4   Mine `cotp` patterns in $\mathcal{D}_{C^{cur}}$ (and their extents) using `MinIntChange`

5   **while** $C^{cur} \neq C^{rel}$ *and within* **computational budget do**

6   Choose the next relevant cut $(attr, cut)$ with $cut \in C^{rel}_{attr} \backslash C^{cur}_{attr}$

7   Add the relevant cut cut to C^{cur}

8   Mine **new cotp patterns** (and their extents) in $\mathcal{D}_{C^{cur}}$

---

4   Anytime (Interval) Pattern Mining with Guarantees

Algorithm `RefineAndMine` starts by mining patterns in a coarse discretization. It continue by mining more patterns in increasingly finer discretizations until the search space is totally explored (final complete lattice being $(\mathcal{D}_{C^{rel}}, \sqsubseteq)$). According to Proposition 1, the description spaces built on discretizations are complete sub-lattices of the total description space. A similar idea involves performing successive enumeration of growing pattern languages (projections) [7]. In our case, it is a successive enumeration of growing complete sub-lattices. For the sake of generality, in the following of this section $(\mathcal{D}, \sqsubseteq)$ denotes a *complete lattice*, and for all $i \in \mathbb{N}^*$, $(\mathcal{D}_i, \sqsubseteq)$ denotes *complete sub-lattices* of $(\mathcal{D}, \sqsubseteq)$ such that $\mathcal{D}_i \subseteq \mathcal{D}_{i+1} \subseteq \mathcal{D}$. For instance, in `RefineAndMine`, the total complete lattice is $(\mathcal{D}_{C^{rel}}, \sqsubseteq)$ while the $(\mathcal{D}_i, \sqsubseteq)$ are $(\mathcal{D}_{C^{cur}}, \sqsubseteq)$ at each step. Following Sec. **??** notation, the outputted set $\mathcal{S}_i$ at a step i contains the set of all *cotp extents* associated to $\mathcal{D}_i$. Before giving the formulas of $\overline{accuracy_\phi}(\mathcal{S}_i)$ and $\overline{specificity}(\mathcal{S}_i)$, we give some necessary definitions and underlying properties. At the end of this section, we show how `RefineAndMine` can be adapted to efficiently compute these two bounds for the case of interval patterns.

Again, similarly to the interval pattern structure [18], we define in the general case a *pattern structure* $\mathbb{P} = (\mathcal{G}, (\mathcal{D}, \sqsubseteq), \delta)$ on the *complete lattice* $(\mathcal{D}, \sqsubseteq)$ where $\mathcal{G}$ is a non empty finite set (partitioned into $\{\mathcal{G}^+, \mathcal{G}^-\}$) and $\delta : \mathcal{G} \to \mathcal{D}$ is a mapping associating to each object its description (recall that in interval pattern structure, δ is the degenerated hyper-rectangle representing a single point). The extent *ext* and intent *int* operators are then respectively given by $ext : \mathcal{D} \to \wp(\mathcal{G}), d \mapsto \{g \in \mathcal{G} \mid d \sqsubseteq \delta(g)\}$ and $int : \wp(\mathcal{G}) \to \wp(\mathcal{G}), A \mapsto \prod_{g \in A} \delta(g)$ with $\prod$ represents the meet operator in $(\mathcal{D}, \sqsubseteq)$ [12].

4.1   Approximating descriptions in a complete sub-lattice

Upper and lower approximations of a pattern. We start by approximating each pattern in $\mathcal{D}$ using two patterns in $\mathcal{D}_i$. Consider for instance Fig. 3 where $\mathcal{D}$ is the space of interval patterns in $\mathbb{R}^2$ while $\mathcal{D}_C$ is the space containing only rectangles that can be built over discretization $dr(C)$ with $C = (\{1, 4, 6, 8\}, \{1, 3, 5, 6\})$. Since the hatched rectangle $d = [3, 7] \times [2, 5.5] \in \mathcal{D}$ does not belong to $\mathcal{D}_C$, two descriptions in $\mathcal{D}_C$ can be used to encapsulate it. The first one, depicted by a gray rectangle, is called the *upper approximation* of d. It is given by the smallest rectangle in $\mathcal{D}_C$ enclosing d. Dually, the second approximation represented as a black rectangle and coined *lower approximation* of d, is given by the greatest rectangle in $\mathcal{D}_C$ enclosed by d. This two denominations comes from Rough Set Theory [26] where lower and upper approximations form together a *rough set* and try to capture the undefined rectangle $d \in \mathcal{D}\backslash\mathcal{D}_C$. Definition 1 formalizes these two approximations in the general case.

Definition 1. The *upper approximation mapping* $\overline{\psi_i}$ and *lower approximation mapping* $\underline{\psi_i}$ are the mappings defined as follows:

$$\overline{\psi_i} : \mathcal{D} \to \mathcal{D}_i, d \mapsto \bigsqcup \{c \in \mathcal{D}_i \mid c \sqsubseteq d\} \qquad \underline{\psi_i} : \mathcal{D} \to \mathcal{D}_i, d \mapsto \bigsqcap \{c \in \mathcal{D}_i \mid d \sqsubseteq c\}$$

The existence of these two mappings is ensured by the fact that $(\mathcal{D}_i, \sqsubseteq)$ is a complete sublattice of $(\mathcal{D}, \sqsubseteq)$. *Theorem 4.1* in [9] provides more properties for the two aforementioned mappings. Proposition 2 restates an important property.

Proposition 2. $\forall d \in \mathcal{D} : \overline{\psi_i}(d) \sqsubseteq d \sqsubseteq \underline{\psi_i}(d)$. The term lower and upper-approximation here are reversed to fit the fact that in term of *extent* we have $\forall d \in \mathcal{D}: ext(\underline{\psi_i}(d)) \subseteq ext(d) \subseteq ext(\overline{\psi_i}(d))$.

A projected pattern structure. Now that we have the upper-approximation mapping $\overline{\psi_i}$, one can associate a new pattern structure $\mathbb{P}_i = (\mathcal{G}, (\mathcal{D}_i, \sqsubseteq), \overline{\psi_i} \circ \delta)^6$ to the pattern space $(\mathcal{D}_i, \sqsubseteq)$. It is worth mentioning, that while extent ext_i mapping associated to $\mathbb{P}_i$ is equal to ext, the intent int_i of $\mathbb{P}_i$ is given by $int_i : \wp(\mathcal{G}) \to \mathcal{D}_i, A \mapsto \overline{\psi_i}(int(A))$. Note that, the set of `cotp` patterns associated to $\mathbb{P}_i$ are given by $int_i[\wp(\mathcal{G}^+)] = \overline{\psi_i}[int[\wp(\mathcal{G}^+)]]$. That is, the upper approximation of a `cotp` pattern in $\mathbb{P}$ is a `cotp` pattern in $\mathbb{P}_i$.

Encapsulating patterns using their upper-approximations. We want to encapsulate any description by knowing only its upper-approximation. Formally, we want some function $f : \mathcal{D}_i \to \mathcal{D}_i$ such that $(\forall d \in \mathcal{D}) \overline{\psi_i}(d) \sqsubseteq d \sqsubseteq f(\overline{\psi_i}(d))$. Proposition 3 define such a function f (called *core*) and states that the *core* is the tightest (w.r.t. $\sqsubseteq$) possible function f.

Proposition 3. The function $core_i$ defined by:

$$core_i : \mathcal{D}_i \to \mathcal{D}_i, c \mapsto core(c) = \underline{\psi_i}\left(\bigsqcup\left\{d \in \mathcal{D} \mid \overline{\psi_i}(d) = c\right\}\right)$$

---

[6]Pattern structure $\mathbb{P}_i$ is said to be a meet-preserving projection by $\overline{\psi_i}$ of $\mathbb{P}$ [8].
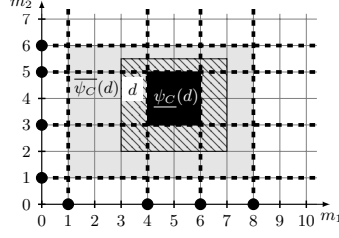
Fig. 3: Description $d = [3,7] \times [2,5.5]$ in $\mathcal{D}$ (blue rectangle) and $C = (\{1,4,6,8\}, \{1,3,5,6\})$. Upper approximation of d in $\mathcal{D}_C$ is $\overline{\psi_C}(d) = [1,8) \times [1,6)$ (green rectangle) while lower approximation of d is $\underline{\psi_C}(d) = [4,6) \times [3,5)$ (red rectangle).

verify the following property: $\forall d \in \mathcal{D} : \overline{\psi_i}(d) \sqsubseteq d \sqsubseteq \underline{\psi_i}(d) \sqsubseteq core_i(\overline{\psi_i}(d))$. Moreover, for $f : \mathcal{D}_i \to \mathcal{D}_i$, $(\forall d \in \mathcal{D})\, d \sqsubseteq f(\overline{\psi_i}(d)) \Leftrightarrow (\forall c \in \mathcal{D}_i)\, core_i(c) \sqsubseteq f(c)$.

Note that, while the *core* operator definition depends clearly on the complete lattice $(\mathcal{D}, \sqsubseteq)$, its computation should be done independently from $(\mathcal{D}, \sqsubseteq)$.

We show here how to compute the *core* in `RefineAndMine`. In each step and for cut-points $C = (C_k) \subseteq \wp(\mathbb{R})^p$, the finite lattice $(\mathcal{D}_C, \sqsubseteq)$ is a sub-lattice of the finest finite lattice $(\mathcal{D}_{C^{rel}}, \sqsubseteq)$ (since $C \leq C^{rel}$). Thereby, the *core* is computed according to this latter as follows: Let $d \in \mathcal{D}_C$ with $d.I_k = [a_k, b_k)$ for all $k \in \{1, ..., p\}$. The left (resp. right) bound of $core_C(d).I_k$ for any k is equal to $next(a_k, C_k)$ (resp. $prev(b_k, C_k)$) if $next(a_k, C_k^{rel}) \notin C_k$ (resp. $prev(b_k, C_k^{rel}) \notin C_k$). Otherwise, it is equal to a_k (resp. b_k). Consider the step $\mathcal{C} = (\{2,3\}, \{4,5\})$ in `RefineAndMine` (its associated discretization is depicted in Fig. 2 (left)) and recall that the relevant cuts set is $\mathcal{C}^{rel} = (\{2,3,4,5\}, \{4,5\})$. Let be the bottom description $d = \perp = \mathbb{R}^2$, the core of d at this step is $core_{\mathcal{C}^{cur}}(d) = (-\infty, 3) \times \mathbb{R}$. Indeed, there is three descriptions in $\mathcal{D}_{\mathcal{C}^{rel}}$ which upper approximation is $d = \perp$, namely d, $c_1 = (-\infty, 4) \times \mathbb{R}$ and $c_2 = (-\infty, 5) \times (-\infty, \infty)$. Their lower approximations are respectively $\perp$, $(-\infty, 3) \times \mathbb{R}$ and $(-\infty, 3) \times \mathbb{R}$. The join (intersection) of these three descriptions is then $core_{\mathcal{C}^{cur}}(d) = (-\infty, 3) \times \mathbb{R}$. Note that particularly for interval patterns, the *core* has *monotonicity*, that is $(\forall c, d \in \mathcal{D}_C)\, c \sqsubseteq d \Rightarrow core_C(c) \sqsubseteq core_C(d)$.

4.2 Bounding accuracy and specificity metrics

At a step i, the outputted extents $\mathcal{S}_i$ consists in the set of `cotp` extents in $\mathbb{P}_i$. Formally, $\mathcal{S}_i = ext[int_i[\wp(\mathcal{G}^+)]]]$ while their associated intents (`cotp` patterns) set is given by $int_i[\mathcal{S}_i] = int_i[\wp(\mathcal{G}^+)]$. Theorem 1 and Theorem 2 gives respectively the *bounds* $\overline{accuracy_\phi}$ and $\overline{specificity}$.

Theorem 1. *Let* $\phi : \mathcal{D} \to \mathbb{R}$ *be a discriminant objective quality measure. The accuracy metric is bounded by:*

$$\overline{accuracy_\phi}(\mathcal{S}_i) = \sup_{c \in int_i[\mathcal{S}_i]} \left[\phi^* \big(tpr(c), fpr(core_i(c)) \big) - \phi^* \big(tpr(c), fpr(c) \big) \right]$$

Moreover $\overline{accuracy_\phi}(\mathcal{S}_{i+1}) \leq \overline{accuracy_\phi}(\mathcal{S}_i)$.

Theorem 2. *The specificity metric is bounded by:*

$$\overline{specificity}(\mathcal{S}_i) = \sup_{c \in int_i[\mathcal{S}_i]} \left((|ext(c)| - |ext(core_i^+(c))|)/(2 \cdot |\mathcal{G}|) \right)$$

where $core_i^+(c) = int_i(ext^+(core_i(c)))$*, that is* $core_i^+(c)$ *is the closure on the positive of* $core_i(c)$ *in* $\mathbb{P}_i$*. Moreover* $\overline{specificity}(\mathcal{S}_{i+1}) \leq \overline{specificity}(\mathcal{S}_i)$*.*

4.3   Computing and updating bounds in `RefineAndMine`

We show below how the different steps of the method `RefineAndMine` (see Algorithm 1) should be updated in order to compute the two bounds $\overline{accuracy}$ and $\overline{specificity}$. For the sake of brevity, we explain here a naive approach to provide an overview of the algorithm. Note that here, *core* (resp. *core*$^+$) refers to $core_{C^{cur}}$ (resp. $core_{C^{cur}}^+$).

Compute the initial bounds (line 4). As `MinIntChange` enumerates all `cotp` patterns $d \in \mathcal{D}_{C^{cur}}$, `RefineAndMine` stores in a key-value structure (i.e. map) called `BoundPerPosExt` the following entries:

$$ext^+(d) : \left(\phi(d), \phi^*\left(tpr(d), fpr(core(d))\right), (|ext(d)| - |ext(core^+(d))|)/(2 \cdot |\mathcal{G}|)\right)$$

The *error-bounds* $\overline{accuracy_\phi}$ and $\overline{specificity}$ are then computed at the end by a single pass on the entries of `BoundPerPosExt` using Theorems 1 and 2.

Update the bounds after adding a new cut-point (line 8). In order to compute the new *error-bounds* $\overline{accuracy_\phi}$ and $\overline{specificity}$ which decrease according to theorems 1 and 2, one need to add/update some entries in the structure `BoundPerPosExt`. For that, only two types of patterns should be looked for:

1. The new `cotp` patterns mined by `RefineAndMine`, that is those which left or right bound on attribute *attr* is the added value *cut*. Visiting these patterns will add potentially new entries in `BoundPerPosExt` or update ancient ones.
2. The old `cotp` which core changes (i.e. becomes less restrictive) in the new discretization. One can show that these patterns are those which left bound is $prev(cut, C_{attr}^{cur})$ or right bound is $next(cut, C_{attr}^{cur})$ on attribute *attr* (see Table 1 for notations). Visiting these patterns will only update ancient entries of `BoundPerPosExt` by potentially decreasing both second and third value.

Adding a new cut-point (line 7). We have implemented for now a strategy which aims to decrease the $\overline{accuracy_\phi}$. For that, we search in `BoundPerPosExt` for the description d having the maximal value $\phi^*\left(tpr(d), fpr(core(d))\right)$. In order to decrease $\overline{accuracy_\phi}$, we increase the size of $core(d)$ (to potentially increase $fpr(core(d))$). This is equivalent to choose a cut-point in the border region $C_{attr}^{rel} \setminus C_{attr}^{cur}$ for some attribute *attr* such that $cut \in d.I_{attr} \setminus core(d).I_{attr}$. Consider that we are in the step where the current discretization C^{cur} is the one depicted in Fig. 2. Imagine that the bottom pattern $d = (-\infty, +\infty) \times (-\infty, +\infty)$ is the

one associated to the maximal value $\phi^*\big(tpr(d), fpr(core(d))\big)$. The new cut-point should be chosen in $\{4,5\}$ for $attr = 1$ (in this case, all remaining cuts are potentially good cuts). Note that if for such description there is no remaining relevant cut in its *border regions* for all $attr \in \{1, ..., p\}$ then $core(d) = d$ ensuring that d is the *top pattern*.

$\wp(E)$	powerset of some set E
$f[E]$	image of set E by a function f (i.e. $f[E] = \{f(e) \mid e \in E\}$)
$(\mathcal{G}, \mathcal{M})$	finite numerical dataset with $|\mathcal{M}| = p$
$\{\mathcal{G}^+, \mathcal{G}^-\}$	partition of $\mathcal{G}$ into positive and negative instances
$(\mathcal{D}, \sqsubseteq)$	complete lattice of interval patterns (p-dimensional)
$\mathbb{P} = (\mathcal{G}, (\mathcal{D}, \sqsubseteq), \delta)$	interval pattern structure with its operators *int* and *ext*
$ext[\mathcal{D}] \subseteq \wp(\mathcal{G})$	finite set of all possible extents provided by $\mathcal{D}$
$\phi : \mathcal{D} \to \mathbb{R}$	discriminant objective quality measure
$\phi^* : [0,1]^2 \to \mathbb{R}$	ϕ associated measure on (tpr, fpr) domain
`cotp`	closed on the positive patterns ($d = int(ext(d) \cap \mathcal{G}^+)$)
$d.I_k$	interval $\subseteq \mathbb{R}$ of $d \in \mathcal{D}$ and attribute $k \in \{1, ..., p\}$ (projection)
$next(c, A)$	following element of $c \in \mathbb{R}$ in A finite (i.e. $\inf\{a \in A \mid c < a\}$)
$prev(c, A)$	preceding element of $c \in \mathbb{R}$ in A finite (i.e. $\sup\{a \in A \mid a < c\}$)

Table 1: Notations

5   Empirical Study

In this section we report quantitative experiments over the implemented algorithms. For reproducibility purpose, the source code is made available in our companion page[7] which also provide a wider set of experiments (see also Appendix B). Experiments were carried out on a variety of datasets (Tab. 2) involving ordinal or continuous numerical attributes from the UCI repository.

First, we study the effectiveness of `RefineAndMine` in terms of the speed of convergence to the optimal solution, as well as regarding the evolution over time of the accuracy of the provided bounding quality's guarantee. To this end, we report in Fig. 4, the behavior of `RefineAndMine` (i.e. quality and bounding

Dataset	num	rows	intervals	class	α	Dataset	num	rows	intervals	class	α
ABALONE_02_M	2	4177	56×10^6	M	0.37	GLASS_02_1	2	214	161×10^6	1	0.33
ABALONE_03_M	3	4177	74×10^9	M	0.37	GLASS_04_1	4	214	5×10^{15}	1	0.33
CREDITA_02_+	2	666	1×10^9	+	0.45	HABERMAN_03_2	3	306	47×10^6	2	0.26
CREDITA_04_+	4	666	3×10^{15}	+							

Table 2: Benchmark datasets and their characteristics: number of numerical attributes, number of rows, number of all possible intervals, the considered class and its prevalence

---

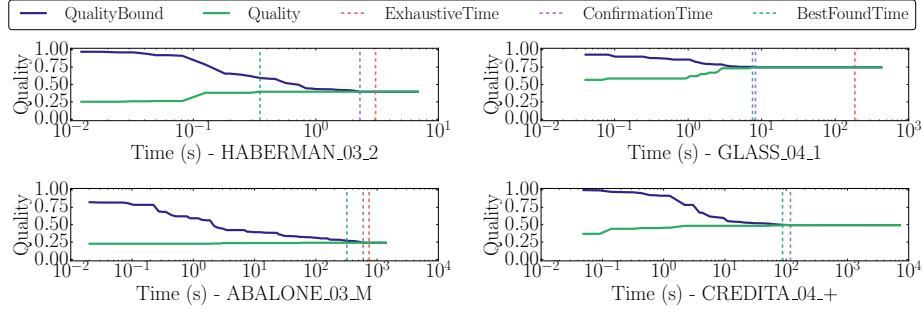[7]Companion page: `https://github.com/Adnene93/RefineAndMine`

Fig. 4: Evolution over time of top pattern quality and its bounding guarantee provided by `RefineAndMine`. Execution time is reported in log scale. The last figure reports that the exhaustive enumeration algorithm was not able to finish within 2 hours

guarantee) according to the execution time to evaluate the time/quality trade-off of the devised approach. $\overline{accuracy}$ as presented in Theorem 1 is the difference between the quality and its bounding measure. The experiments were conducted by running both `RefineAndMine` and the exhaustive enumeration algorithm (`MinIntChange` performed considering $\mathcal{D}_{C^{rel}}$) on the benchmark datasets using informedness measure. The exhaustive algorithm execution time enables the estimation of the computational overhead incurred by `RefineAndMine`. We interrupt a method if its execution time exceeds two hours. Note that, in the experiments, we choose to disable the computation of specificity since the latter is only optional and does not affect the effectiveness of the algorithm. This in contrast to the quality bound computation which is essential as it guides `RefineAndMine` in the cut-points selection strategy. The experiments give evidence of the effectiveness of `RefineAndMine` both in terms of finding the optimal solution as well as in providing stringent bound on the top quality pattern in a prompt manner. Two important milestones achieved by `RefineAndMine` during its execution are highlighted in Fig. 4. The first one, illustrated by the *green dotted line*, points out the required time to find the best pattern. The second milestone (*purple line*) is reached when the quality's and the bound's curves meet which ensures that the best quality was already found by `RefineAndMine`. Interestingly, we observe that for most configurations (including experiments reported in Appendix B) the second milestone is attained by `RefineAndMine` promptly and well before the exhaustive method termination time. This is explained by the fact that the adopted cut points selection strategy aims to decrease as early as possible the $\overline{accuracy}$ metric. Finally, `RefineAndMine` requires in average 2 times of the requested execution time (*red dotted line*) by the exhaustive algorithm. This overhead is mostly incurred by the quality guarantee computation.

We illustrate in Fig. 5 the behavior of `RefineAndMine` in terms of finding diverse set of high quality patterns covering different parts of the dataset. To evaluate how quickly the devised approach finds a diverse patterns set, we run the exhaustive approach over the benchmark datasets to constitute a top-k diverse patterns set heuristically as following: the patterns extracted by the exhaus-

tive search algorithm are sorted according to the quality measure and the best pattern is kept in the returned top-k list. Next, the complete patterns list are iterated over, and the top-k list is augmented by a pattern if and only if its similarity with all the patterns of the current content of the top-k list is lower than a given threshold (a Jaccard index between extents). This process is interrupted if the desired number of patterns of the top-k list is reached or no remaining dissimilar pattern is available. Similar post-processing techniques were used by [21,6]. Once this ground truth top-k list is constituted over some benchmark dataset, we run `RefineAndMine` and measure the *specificity quantity* of the obtained results set with the top-k list. *specificity* metric is rewritten in eq. 1 to accommodate the desired evaluation objective of these experiments. Still, it remains upper-bounded by the general formula of $\overline{specificity}$ given in Theorem 2. This in order to evaluate at what extent the visited patterns by `RefineAndMine` well-cover the ground-truth patterns which are scattered over different parts of some input dataset. We report in Fig. 5 both *specificity* and its bounding guarantee $\overline{specificity}$, as well as, a diversity metric defined in eq. 2. Such metric was defined in [6] to evaluate the ability of an approximate algorithm to retrieve a given ground-truth (i.e. diversified top-k discriminant patterns set). This diversity metric relies on a similarity rather than a distance (as in specificity), and is equal to 1 when all patterns of the top-k list are fully discovered.

$$specificity(\mathcal{D}_C, TopK) = \sup_{d \in TopK} \inf_{c \in \mathcal{D}_C} (|ext(d) \, \Delta \, ext(c)|/|\mathcal{G}|) \qquad (1)$$

$$diversity(\mathcal{D}_C, TopK) = \arg_{d \in TopK} \sup_{c \in \mathcal{D}_C} (Jaccard(ext(d), ext(c))) \qquad (2)$$

In most configurations, we notice that `RefineAndMine` is able to uncover approximately 80% (given by *diversity*) of the ground truth's patterns in less than 20% of the time required by the exhaustive search algorithm. For instance, in ABALONE_02_M, we observe that after 2 seconds (12% of the required time for the exhaustive algorithm), the patterns outputted by `RefineAndMine` approximate 92% of the ground truth. Moreover, we observe that the specificity and $\overline{specificity}$ decrease quickly with time, guaranteeing a high level of diversity.

For a comparative study, we choose to compare `RefineAndMine` with the closest approach following the same paradigm (anytime) in the literature, that is the recent `MCTS4DM` technique [18]. `MCTS4DM` is depicted by the authors as an algorithm which enables the anytime discovery of a diverse patterns set of high quality. While `MCTS4DM` ensures interruptibility and an exhaustive exploration if given enough time and memory budget, it does not ensures any theoretical guarantees on the distance from optimality and on the diversity. We report in Fig. 6 a comparative evaluation between the two techniques. To realize this study, we investigate the ability of the two methods in retrieving the ground truth patterns, this by evaluating the quality of their respective diversified top-k lists against the ground truth using the diversity metric (eq. 2). We observe that `RefineAndMine` outperforms `MCTS4DM` both in terms of finding the best pattern, and of uncovering diverse patterns set of high qualities. This is partially due to the fact that our method is specifically tailored for mining discriminant patterns
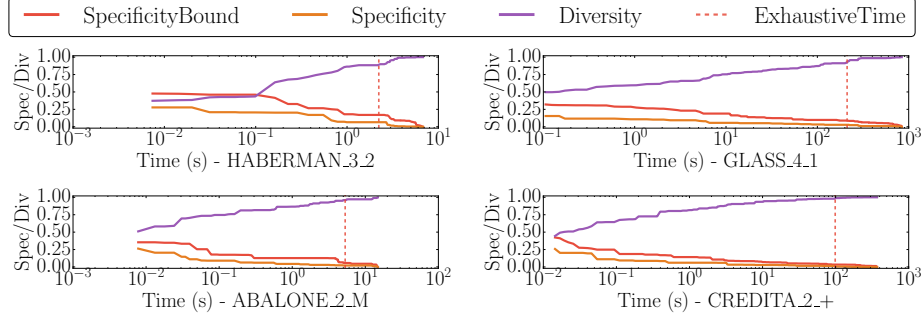
Fig. 5: Efficiency of `RefineAndMine` in terms of retrieving a diverse patterns set. Execution time is reported in log scale. The ground-truth for each benchmark dataset corresponds to the obtained Top10 diversified patterns set with a similarity threshold of 0.25 and a minimum *tpr* of 15% .
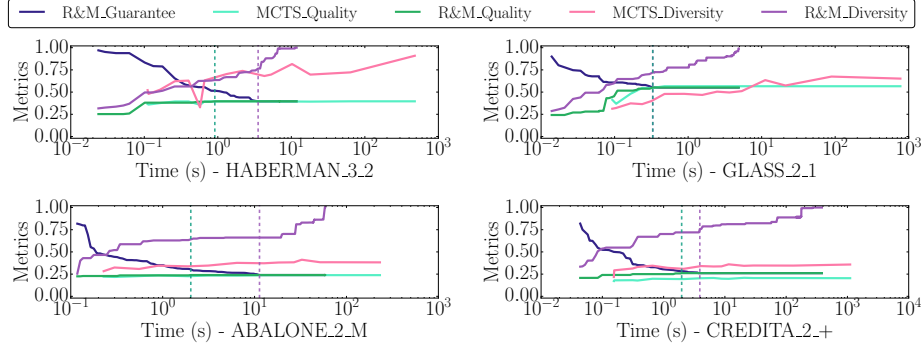

Fig. 6: Comparative experiments between `RefineAndMine` ($R\&M$) and `MCTS4DM`. Execution time is reported in log scale. The ground-truth for each benchmark dataset corresponds to the obtained Top10 diversified patterns set with a similarity threshold of 0.25 and no minimum support size threshold.

in numerical data, in contrast to `MCTS4DM` which is agnostic of the interestingness measure and the description language. Note that, to enable a fair comparison of the two approaches, we report the full time spent by the methods including the overhead induced by the post-computation of the diversified top-k patterns set.

6   Discussions and Conclusion

We introduced a novel anytime pattern mining technique for uncovering discriminant patterns in numerical data. We took a close look to discriminant interestingness measures to focus on hyper-rectangles in the dataset fostering the presence of some class. By leveraging the properties of the quality measures, we defined a guarantee on the accuracy of `RefineAndMine` in approximating the optimal solution which improves over time. We also presented a guarantee

on the specificity of `RefineAndMine` –which is agnostic of the quality measure– ensuring its diversity and completeness. Empirical evaluation gives evidence of the effectiveness both in terms of finding the optimal solution (w.r.t. the quality measure ϕ) and revealing local optimas located in different parts of the data.

This work paves the way for many improvements. `RefineAndMine` can be initialized with more sophisticated discretization techniques [20,10]. We have to investigate additional cut-points selection strategies. While we considered here discriminant pattern mining, the enumeration process (i.e. *successive refinement of discretizations*) can be tailored to various other quality measures in subgroup discovery. For example, the accuracy bound guarantee definition can be extended to handle several other traditional interestingness measures such as Mutual Information, χ^2 and *Gini split* by exploiting their (quasi)-convexity properties w.r.t. *tpr* and *fpr* variables [25,1]. Other improvements include the adaptation of `RefineAndMine` for high-dimensional datasets and its generalization for handling additional types of attributes (categorical, itemsets, etc.). The latter is facilitated by the generic notions from Section 4 and the recent works of Buzmakov et al. [7].

References

1. Abudawood, T., Flach, P.A.: Evaluation measures for multi-class subgroup discovery. In: ECML PKDD. pp. 35–50. Springer (2009)
2. Atzmueller, M., Puppe, F.: Sd-map–a fast algorithm for exhaustive subgroup discovery. In: PKDD. pp. 6–17. Springer (2006)
3. Boley, M., Horváth, T., Poigné, A., Wrobel, S.: Listing closed sets of strongly accessible set systems with applications to data mining. Theor. Comput. Sci. 411(3), 691–700 (2010)
4. Boley, M., Lucchese, C., Paurat, D., Gärtner, T.: Direct local pattern sampling by efficient two-step random procedures. In: KDD. pp. 582–590 (2011)
5. Boley, M., Moens, S., Gärtner, T.: Linear space direct pattern sampling using coupling from the past. In: KDD. pp. 69–77 (2012), `http://doi.acm.org/10.1145/2339530.2339545`
6. Bosc, G., Boulicaut, J.F., Raïssi, C., Kaytoue, M.: Anytime discovery of a diverse set of patterns with monte carlo tree search. DMKD (in press) (2017), `https://doi.org/10.1007/s10618-017-0547-5`
7. Buzmakov, A., Kuznetsov, S.O., Napoli, A.: Fast generation of best interval patterns for nonmonotonic constraints. In: ECML PKDD. pp. 157–172. Springer (2015)
8. Buzmakov, A., Kuznetsov, S.O., Napoli, A.: Revisiting pattern structure projections. In: Formal Concept Analysis. pp. 200–215. Springer (2015)
9. Denecke, K., Wismath, S.L.: Galois connections and complete sublattices. In: Galois Connections and Applications, pp. 211–229. Springer (2004)
10. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. In: IJCAI. pp. 1022–1029 (1993)

11. Ganter, B., Wille, R.: Formal Concept Analysis. Springer (1999)
12. Ganter, B., Kuznetsov, S.O.: Pattern structures and their projections. In: ICCS 2001. LNCS (2120). 129–142.
13. Garriga, G.C., Kralj, P., Lavrac, N.: Closed sets for labeled data. Journal of Machine Learning Research 9, 559–580 (2008)
14. Geng, L., Hamilton, H.J.: Interestingness measures for data mining: A survey. ACM Comput. Surv. 38(3), 9 (2006), `http://doi.acm.org/10.1145/1132960.1132963`
15. Giacometti, A., Soulet, A.: Dense neighborhood pattern sampling in numerical data. In: Proceedings of the 2018 SIAM International Conference on Data Mining. pp. 756–764. SIAM (2018)
16. Grosskreutz, H., Rüping, S.: On subgroup discovery in numerical domains. In: PKDD. Lecture Notes in Computer Science, vol. 5781, p. 30 (2009), `https://doi.org/10.1007/978-3-642-04180-8_15`
17. Hu, Q., Imielinski, T.: Alpine: Progressive itemset mining with definite guarantees. In: Proceedings of the 2017 SIAM International Conference on Data Mining. pp. 63–71. SIAM (2017)
18. Kaytoue, M., Kuznetsov, S.O., Napoli, A.: Revisiting Numerical Pattern Mining with Formal Concept Analysis. In: IJCAI. pp. 1342–1347 (2011)
19. Kaytoue, M., Kuznetsov, S.O., Napoli, A., Duplessis, S.: Mining gene expression data with pattern structures in fca. Inf. Sci. 181(10), 1989–2001 (2011), `https://doi.org/10.1016/j.ins.2010.07.007`
20. Kurgan, L., Cios, K.J.: Discretization algorithm that uses class-attribute interdependence maximization. In: IC-AI. pp. 980–987 (2001)
21. van Leeuwen, M., Knobbe, A.J.: Diverse subgroup set discovery. Data Min. Knowl. Discov. 25(2), 208–242 (2012)
22. Lenca, P., Meyer, P., Vaillant, B., Lallich, S.: On selecting interestingness measures for association rules: User oriented description and multiple criteria decision aid. European Journal of Operational Research 184(2), 610–626 (2008)
23. Lucas, T., Silva, T.C.P.B., Vimieiro, R., Ludermir, T.B.: A new evolutionary algorithm for mining top-k discriminative patterns in high dimensional data. Appl. Soft Comput. 59, 487–499 (2017), `https://doi.org/10.1016/j.asoc.2017.05.048`
24. Mampaey, M., Nijssen, S., Feelders, A., Knobbe, A.J.: Efficient algorithms for finding richer subgroup descriptions in numeric and nominal data. In: ICDM. pp. 499–508 (2012), `https://doi.org/10.1109/ICDM.2012.117`
25. Morishita, S., Sese, J.: Traversing itemset lattice with statistical metric pruning. In: ACM SIGMOD-SIGACT-SIGART. pp. 226–236 (2000)
26. Pawlak, Z.: Rough sets. International Journal of Parallel Programming 11(5), 341–356 (1982)
27. Roman, S.: Lattices and Ordered Sets. Springer New York (2008), `https://books.google.fr/books?id=NZN8aum26LgC`
28. Wrobel, S.: An algorithm for multi-relational discovery of subgroups. In: PKDD. pp. 78–87 (1997)
29. Yang, Y., Webb, G.I., Wu, X.: Discretization methods. In: Data Mining and Knowledge Discovery Handbook, 2nd ed., pp. 101–116. Springer (2010)
30. Zilberstein, S.: Using anytime algorithms in intelligent systems. AI Magazine 17(3), 73–83 (1996)

A   Appendix - Proofs

A.1   Proof of Proposition 1

Proof. By construction, $(\mathcal{D}_C, \sqsubseteq)$ is a **closure system** (preserve meet) on $(\mathcal{D}, \sqsubseteq)$. Indeed, let d_1 and d_2 be in $\mathcal{D}_C$ we have $\exists S_1, S_2 \subseteq dr(C)$ such that $d_1 = \bigsqcap S_1$ and $d_2 = \bigsqcap S_2$. Thus:

$$d_1 \sqcap d_2 = \left(\bigsqcap S_1\right) \sqcap \left(\bigsqcap S_2\right) = \bigsqcap \left(S_1 \cup S_2\right) \in \mathcal{D}_C$$

since $S_1 \cup S_2 \subseteq dr(C)$.

Let us show now that $(\mathcal{D}_C, \sqsubseteq)$ is also a **kernel system** (preserve join) on $(\mathcal{D}, \sqsubseteq)$. We have $C = (C_k)_{1 \le k \le p}$. Consider the following case of d_1 and d_2:

$$d_j = \prod_{k=1}^{p} I_k^j \text{ with } I_k^j = [a_k^j, b_k^j), \ j \in \{1, 2\} \text{ and } a_k^j < b_k^j \in C_k$$

We have:

$$d_1 \sqcup d_2 = \prod_{k=1}^{p} [sup(a_k^1, a_k^2), inf(b_k^1, b_k^2))$$

Clearly, $d_1 \sqcup d_2 \in \mathcal{D}_C$, since the left bound $sup(a_k^1, a_k^2)$ and the right bound $inf(b_k^1, b_k^2)$ remains in C^k for all $k \in p$. Note that if $sup(a_k^1, a_k^2) > inf(b_k^1, b_k^2)$ for at least one $k \in \{1..p\}$, then $d_1 \sqcup d_2 = \top = \emptyset \in \mathcal{D}_C$. The two remaining cases of interval where we use $+\infty$ and $-\infty$ can be handled in, almost, the same way.

We conclude that $(\mathcal{D}_C, \sqsubseteq)$ is a finite (complete) sub-lattice of $(\mathcal{D}, \sqsubseteq)$ (since any finite lattice is by definition complete).

The second part of the proposition is straight-forward since both posets $(\mathcal{D}_{C_1}, \sqsubseteq)$ and $(\mathcal{D}_{C_2}, \sqsubseteq)$ are complete sub-lattice of the same complete lattice $(\mathcal{D}, \sqsubseteq)$ and $\mathcal{D}_{C_1} \subseteq \mathcal{D}_{C_2}$ $(dr(C_1) \le dr(C_2)$ since $C_1 \le C_2)$.   □

A.2   Proof of Proposition 2

Proof. Proposition 2 is a small result from Theorem 4.1 [9]. Since $(\mathcal{D}_i, \sqsubseteq)$ is a complete sub-lattice of $(\mathcal{D}, \sqsubseteq)$, then the mappings $\underline{\psi_i^*}$ and $\overline{\psi_i^*}$ defined below

$$\underline{\psi_i^*}(d) : \mathcal{D} \to \mathcal{D}, d \mapsto \underline{\psi_i}(d) \ \textbf{ and } \ \overline{\psi_i^*}(d) : \mathcal{D} \to \mathcal{D}, d \mapsto \overline{\psi_i}(d)$$

are respectively **join-preserving closure** (thus extensive $d \sqsubseteq \underline{\psi_i^*}(d)$) and **meet-preserving kernel** (thus contractive $\overline{\psi_i^*}(d) \sqsubseteq d$) operators on $(\mathcal{D}, \sqsubseteq)$ such that they ranges in $\mathcal{D}_C$ (i.e. $\underline{\psi_i^*}[\mathcal{D}] = \{\underline{\psi_i^*}(d) \mid d \in \mathcal{D}_i\} = \mathcal{D}_i$ and $\overline{\psi_i^*}[\mathcal{D}] = \mathcal{D}_i$). Note that $\mathcal{D}_i$ is the set of **fix-points** of both mappings according to Theorem 4.1 [9].
Recalling that ext is order-reversing concludes the proof   □

A.3   Proof of Proposition 3

Proof. Let $f : \mathcal{D}_i \to \mathcal{D}_i$ be a function. We want to show the following property:

$$(\forall d \in \mathcal{D})\, d \sqsubseteq f(\overline{\psi_i}(d)) \iff (\forall c \in \mathcal{D}_i)\, core_i(c) \sqsubseteq f(c)$$

We start by implication ($\Longleftarrow$). Let $d \in \mathcal{D}$, we have:

$$core_i(\overline{\psi_i}(d)) = \underline{\psi_i}\left(\bigsqcup\left\{x \in \mathcal{D} \mid \overline{\psi_i}(x) = \overline{\psi_i}(d)\right\}\right)$$

Clearly, we have $d \in \left\{x \in \mathcal{D} \mid \overline{\psi_i}(x) = \overline{\psi_i}(d)\right\}$ by **reflexivity** of equality. Thus, $d \sqsubseteq \bigsqcup\left\{x \in \mathcal{D} \mid \overline{\psi_i}(x) = \overline{\psi_i}(d)\right\}$. By **monotonicity** and **extensivity** of $\underline{\psi_i}$:

$$d \sqsubseteq \underline{\psi_i}(d) \sqsubseteq \underline{\psi_i}\left(\bigsqcup\left\{x \in \mathcal{D} \mid \overline{\psi_i}(x) = \overline{\psi_i}(d)\right\}\right) = core_i(\overline{\psi_i}(d))$$

Since from the hypothesis $core_i(\overline{\psi_i}(d)) \sqsubseteq f(\overline{\psi_i}(d))$, we conclude that $d \sqsubseteq f(\overline{\psi_i}(d))$.

It remains to show the other implication ($\Longleftarrow$). Let $c \in \mathcal{D}_i$ and let be the set $S_c = \{x \in \mathcal{D} \mid \overline{\psi_i}(x) = c\}$. Since, from the hypothesis, $\forall x \in S_c : x \sqsubseteq f(\overline{\psi_i}(x)) = f(c)$ then ($f(c)$ is an upper bound of S_c and the join is the smallest upper bound by definition):

$$\bigsqcup S_c \sqsubseteq f(c)$$

Since $f(c) \in \mathcal{D}_i$ we have $\underline{\psi_i}(f(c)) = f(c)$. We obtain by monotonicity of $\underline{\psi_i}$:

$$core_i(c) = \underline{\psi_i}\left(\bigsqcup S_c\right) \sqsubseteq \underline{\psi_i}(f(c)) = f(c)$$

This concludes the demonstration.   □

A.4   Proof of Theorem 1

Before giving the proof of the theorem, we formulate below the following three Lemmas:

Lemma 1. *The richer* $(\mathcal{D}_i, \sqsubseteq)$ *is, the more constraining are the surrounding approximations. Formally:* $\forall d \in \mathcal{D} : \overline{\psi_i}(d) \sqsubseteq \overline{\psi_{i+1}}(d) \sqsubseteq d \sqsubseteq \underline{\psi_{i+1}}(d) \sqsubseteq \underline{\psi_i}(d)$. *Generally speaking:* $\overline{\psi_i} \circ \overline{\psi_{i+1}} = \overline{\psi_i}$ *and* $\underline{\psi_i} \circ \underline{\psi_{i+1}} = \underline{\psi_i}$.

Proof. Without loss of generality, let $i = 1$. Let us show before that $\overline{\psi_1}\left(\overline{\psi_2}(d)\right) = \overline{\psi_1}(d)$ and $\underline{\psi_1}\left(\underline{\psi_2}(d)\right) = \underline{\psi_1}(d)$ for all $d \in \mathcal{D}$.

Since $\mathcal{D}_1 \subseteq \mathcal{D}_2$ then for all $x \in \mathcal{D}_1$ and for all $d \in \mathcal{D}$ we have (since $\overline{\psi_2}$ and $\underline{\psi_2}$ come respectively from kernel and closure operators - See previous proof):

$$x \sqsubseteq \overline{\psi_2}(d) \Leftrightarrow x \sqsubseteq d \quad \text{and} \quad \underline{\psi_2}(d) \sqsubseteq x \Leftrightarrow d \sqsubseteq x$$

Thus:

$$\overline{\psi_1}\big(\overline{\psi_2}(d)\big) = \bigsqcup \big\{ x \in \mathcal{D}_1 \mid x \sqsubseteq \overline{\psi_2}(d) \big\} = \bigsqcup \big\{ x \in \mathcal{D}_1 \mid x \sqsubseteq d \big\} = \overline{\psi_1}(d)$$

$$\underline{\psi_1}\big(\underline{\psi_2}(d)\big) = \bigsqcap \big\{ x \in \mathcal{D}_1 \mid \underline{\psi_2}(d) \sqsubseteq x \big\} = \bigsqcap \big\{ x \in \mathcal{D}_1 \mid d \sqsubseteq x \big\} = \underline{\psi_1}(d)$$

The first part of the proposition is a straight-forward corollary of the first properties. Indeed, since $\overline{\psi_1}$ and $\underline{\psi_2}$ are respectively **contractive** and **extensive**, we conclude:

$$\overline{\psi_1}(d) = \overline{\psi_1}\big(\overline{\psi_2}(d)\big) \sqsubseteq \overline{\psi_2}(d) \quad \textbf{and} \quad \underline{\psi_2}(d) \sqsubseteq \underline{\psi_1}\big(\underline{\psi_2}(d)\big) = \underline{\psi_1}(d)$$

Thus: $\overline{\psi_1}(d) \sqsubseteq \overline{\psi_2}(d) \sqsubseteq d \sqsubseteq \underline{\psi_2}(d) \sqsubseteq \underline{\psi_1}(d)$. $\qquad\square$

Lemma 2. $\forall d \in \mathcal{D} : core_{i+1}(\overline{\psi_{i+1}}(d)) \sqsubseteq core_i(\overline{\psi_i}(d))$. *That is, the core of the upper approximation is less restrictive in richer spaces.*

Proof. Without loss of generality, let $i = 1$. Let $d \in \mathcal{D}$, we need to show that $core_2(\overline{\psi_2}(d)) \sqsubseteq core_1(\overline{\psi_1}(d))$. According to Lemma 1 we have $\overline{\psi_1} \circ \overline{\psi_2} = \overline{\psi_1}$, thus:

$$\big\{ x \in \mathcal{D} \mid \overline{\psi_2}(x) = \overline{\psi_2}(d) \big\} \subseteq \big\{ x \in \mathcal{D} \mid \overline{\psi_1}(x) = \overline{\psi_1}(d) \big\}$$

Thus:

$$\bigsqcup \big\{ x \in \mathcal{D} \mid \overline{\psi_2}(x) = \overline{\psi_2}(d) \big\} \sqsubseteq \bigsqcup \big\{ x \in \mathcal{D} \mid \overline{\psi_1}(x) = \overline{\psi_1}(d) \big\}$$

Since $\underline{\psi_2}$ is monotonic, we obtain:

$$\underline{\psi_2}\left(\bigsqcup \big\{ x \in \mathcal{D} \mid \overline{\psi_2}(x) = \overline{\psi_2}(d) \big\} \right) \sqsubseteq \underline{\psi_2}\left(\bigsqcup \big\{ x \in \mathcal{D} \mid \overline{\psi_1}(x) = \overline{\psi_1}(d) \big\} \right)$$

Since $\underline{\psi_1}$ is extensive and $\underline{\psi_1} \circ \underline{\psi_2} = \underline{\psi_1}$, we obtain:

$$\underline{\psi_2}\left(\bigsqcup \big\{ x \in \mathcal{D} \mid \overline{\psi_2}(x) = \overline{\psi_2}(d) \big\} \right) \sqsubseteq \underline{\psi_1}\left(\bigsqcup \big\{ x \in \mathcal{D} \mid \overline{\psi_1}(x) = \overline{\psi_1}(d) \big\} \right)$$

We conclude that:

$$core_2(\overline{\psi_2}(d)) \sqsubseteq core_1(\overline{\psi_1}(d))$$

This ends the demonstration. $\qquad\square$

Lemma 3. *Using the same notation of Theorem 1, we have:*

$$\forall d \in \mathcal{D} : \phi(d) \leq \phi^*\left(tpr\big(\overline{\psi_i}(d)\big), fpr\big(core_i(\overline{\psi_i}(d))\big) \right)$$

Proof. By application of Proposition 3 we have with $c = \overline{\psi_i}(d) \in \mathcal{D}_i$:

$$c \sqsubseteq d \sqsubseteq core_i(c)$$

Thus, since ext operator is order-reversing we obtain:

$$ext(core_i(c)) \subseteq ext(d) \subseteq ext(c)$$

Thus, since tpr and fpr increase with the extent. We conclude that:

$$tpr(core_i(c)) \leq tpr(d) \leq tpr(c)$$
$$fpr(core_i(c)) \leq fpr(d) \leq fpr(c)$$

Since ϕ is an objective discriminant quality measure we conclude that:

$$\phi(d) = \phi^*(tpr(d), fpr(d)) \leq \phi^*(tpr(c), fpr(d)) \leq \phi^*(tpr(c), fpr(core_i(c)))$$

This ends the demonstration. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ □

Theorem 1 proof. Below the proof. Without loss of generality $i = 1$:

Proof. According to lemma 3 and by considering d a `cotp` pattern in $int[\wp(\mathcal{G}^+)]$, we have:

$$\phi(d) \leq \phi^*\left(tpr\left(\overline{\psi_1}(d)\right), fpr\left(core_1\left(\overline{\psi_1}(d)\right)\right)\right)$$

Since $\overline{\psi_1}(d)$ is a `cotp` pattern induced by $(\mathcal{D}_1, \sqsubseteq)$, we have:

$$\sup_{int[\wp(\mathcal{G}^+)]} \phi(d) \leq \sup_{c \in \overline{\psi_1}[int[\wp(\mathcal{G}^+)]]} \phi^*\left(tpr\left(c\right), fpr\left(core_1\left(c\right)\right)\right)$$

The set $\overline{\psi_1}[int[\wp(\mathcal{G}^+)]]$ represent the set of `cotp` patterns in $\mathcal{D}_1$, that is $int_1[\mathcal{S}_1]$ ($\mathcal{S}_1$ represents the set of `cotp` extents induced by $\mathcal{D}_1$), we obtain:

$$\sup_{d \in int[\wp(\mathcal{G}^+)]} \phi(d) \leq \sup_{c \in int_1[\mathcal{S}_1]} \phi^*\left(tpr\left(c\right), fpr\left(core_1\left(c\right)\right)\right)$$

Thus, since the left quantity is the same as $\sup_{A \in \mathcal{R}} \phi(A)$, we subtract from both side the quantity $\sup_{B \in \mathcal{S}_1} \phi(B) = \sup_{c \in int_1[\mathcal{S}_1]} \phi^*(tpr(c), fpr(c))$. Hence:

$$\sup_{A \in \mathcal{R}} \phi(A) - \sup_{B \in \mathcal{S}_1} \phi(B) \leq \sup_{c \in int_1[\mathcal{S}_1]} \left[\phi^*\left(tpr\left(c\right), fpr\left(core_1(c)\right)\right) - \phi\left(c\right)\right]$$

That is:

$$\overline{accuracy_\phi}(\mathcal{S}_1) = \sup_{c \in int_1[\mathcal{S}_1]} \left[\phi^*\left(tpr\left(c\right), fpr\left(core_1(c)\right)\right) - \phi^*\left(tpr(c), fpr(c)\right)\right]$$

It remains to show that $\overline{accuracy}$ is "order-reversing". From Lemma 1 and Lemma 2 we have for all $d \in \mathcal{D}$:

$$\overline{\psi_1}(d) \sqsubseteq \overline{\psi_2}(d) \sqsubseteq d \sqsubseteq core_2(\overline{\psi_2}(d)) \sqsubseteq core_1(\overline{\psi_1}(d))$$

Thus, by the property of the discriminant to the positive quality measure ϕ we have for all $d \in \mathcal{D}$:

$$\phi^* \left(tpr\left(\overline{\psi_2}(d) \right), fpr\left(core_2\left(\overline{\psi_2}(d) \right) \right) \right) \leq \phi^* \left(tpr\left(\overline{\psi_1}(d) \right), fpr\left(core_1\left(\overline{\psi_1}(d) \right) \right) \right)$$

Particularly, for $c \in \mathcal{D}_2$ (c is a fix-point for $\overline{\psi_2}$) we have:

$$\phi^* \left(tpr\left(c \right), fpr\left(core_2\left(c \right) \right) \right) \leq \phi^* \left(tpr\left(\overline{\psi_1}(c) \right), fpr\left(core_1\left(\overline{\psi_1}(c) \right) \right) \right)$$

We conclude that:

$$\sup_{c \in int_2[\mathcal{S}_2]} \phi^* \left(tpr(c), fpr\left(core_2(c) \right) \right) \leq \sup_{c \in int_1[\mathcal{S}_1]} \phi^* \left(tpr(c), fpr\left(core_1(c) \right) \right)$$

On the other hand, since $\mathcal{D}_1 \subseteq \mathcal{D}_2$, we have:

$$- \sup_{c \in \mathcal{D}_2} \phi(c) \leq - \sup_{c \in \mathcal{D}_1} \phi(c)$$

that is:

$$- \sup_{c \in int_2[\mathcal{S}_2]} \phi(c) \leq - \sup_{c \in int_1[\mathcal{S}_1]} \phi(c)$$

We conclude that:

$$\overline{accuracy_\phi}(\mathcal{S}_2) \leq \overline{accuracy_\phi}(\mathcal{S}_1)$$

This concludes the demonstration. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

A.5   Proof of Theorem 2

Before giving a demonstration to the theorem, below a necessary lemma:

Lemma 4. *Using the same notation of Theorem 2, we have:*

$$\forall d \in int[\wp(\mathcal{G}^+)] : \inf_{c \in int_i[\mathcal{S}_i]} \left(\frac{|ext(d) \, \Delta \, ext(c)|}{|\mathcal{G}|} \right) \leq \frac{|ext(\overline{\psi_i}(d))| - |ext(core_i^+(\overline{\psi_i}(d)))|}{2 \cdot |\mathcal{G}|}$$

Proof. Since the upper approximations (which is a `cotp` in $\mathbb{P}_i$ and thus is in $int_i[\mathcal{S}_i]$) and the closure of the positive of the core (it is in $int_i[\mathcal{S}_i]$) are already good approximations for d (but not necessarily the bests), let $u = \overline{\psi_i}(d)$, we have $ext(d) \subseteq ext(u)$. On the other hand, since $d \sqsubseteq core_i(u) \sqsubseteq core_i^+(u)$ (Proposition 3), we have $ext(core_i^+(u)) \subseteq ext(d)$. Thus, we have $|ext(d) \, \Delta \, ext(u)| = |ext(u)| - |ext(d)|$ and in the other hand $|ext(d) \, \Delta \, ext(core_i^+(u))| = |ext(d)| - |ext(core_i^+(u))|$. We obtain:

$$\inf_{c \in int_i[\mathcal{S}_i]} |ext(d) \, \Delta \, ext(c)| \leq |ext(u)| - |ext(d)|$$

$$\inf_{c \in int_i[\mathcal{S}_i]} |ext(d) \, \Delta \, ext(c)| \leq |ext(d)| - |ext(core_i^+(u))|$$

We conclude:

$$2 \times \inf_{c \in int_i[\mathcal{S}_i]} |ext(d) \, \Delta \, ext(c)| \leq |ext(u)| - |ext(core_i^+(u))|$$

Since $|\mathcal{G}|$ is a constant, we obtain (with $u = \overline{\psi_i}(d)$):

$$\inf_{c \in int_i[\mathcal{S}_i]} \left(\frac{|ext(d) \, \Delta \, ext(c)|}{|\mathcal{G}|} \right) \leq \frac{|ext(u)| - |ext(core_i^+(u))|}{2 \cdot |\mathcal{G}|}$$

This ends the demonstration. □

Theorem 2 proof. Without loss of generality, let $i = 1$.

Proof. According to Lemma 4, we have for a relevant (thus `cotp`) extent $A \in \mathcal{R}$ $(int_1(A) \in int_1[\wp(\mathcal{G}^+)]]$ and thus $int_1(A) \in int_1[\mathcal{S}_1])$:

$$\inf_{B \in S_1} \left(\frac{|A \, \Delta \, B|}{|\mathcal{G}|} \right) \leq \frac{|ext(int_1(A))| - |ext(core_1^+(int_1(A)))|}{2 \cdot |\mathcal{G}|}$$

We conclude thus:

$$\sup_{A \in \mathcal{R}} \inf_{B \in \mathcal{S}_1} \left(\frac{|A \, \Delta \, B|}{|\mathcal{G}|} \right) \leq \sup_{c \in int_1[\mathcal{S}_1]} \frac{|ext(c)| - |ext(core_1^+(c))|}{2 \cdot |\mathcal{G}|}$$

That is to say:

$$\overline{specificity}(\mathcal{S}_1) = \sup_{c \in int_1[\mathcal{S}_1]} \frac{|ext(c)| - |ext(core_1^+(c))|}{2 \cdot |\mathcal{G}|}$$

It remains to show that $\overline{specificity}$ is also **order reversing**. From Lemma 1 and Lemma 2 we have for all $d \in \mathcal{D}$:

$$\overline{\psi_1}(d) \sqsubseteq \overline{\psi_2}(d) \sqsubseteq d \sqsubseteq core_2(\overline{\psi_2}(d)) \sqsubseteq core_1(\overline{\psi_1}(d))$$

We can conclude directly using Lemma 4 (recall that ext is an order reversing) that for all $d \in \mathcal{D}$:

$$\frac{|ext(\overline{\psi_2}(d))| - |ext(core_2(\overline{\psi_2}(d)))|}{2 \cdot |\mathcal{G}|} \leq \frac{|ext(\overline{\psi_1}(d))| - |ext(core_1(\overline{\psi_1}(d)))|}{2 \cdot |\mathcal{G}|}$$

Particularity, for $c \in \mathcal{D}_2$ (c is a fix-point for $\overline{\psi_2}$) we obtain:

$$\frac{|ext(c)| - |ext(core_2(c))|}{2 \cdot |\mathcal{G}|} \leq \frac{|ext(\overline{\psi_1}(c))| - |ext(core_1(\overline{\psi_1}(c)))|}{2 \cdot |\mathcal{G}|}$$

We conclude that:

$$\sup_{c \in int_2[\mathcal{S}_2]} \frac{|ext(c)| - |ext(core_2^+(c))|}{2 \cdot |\mathcal{G}|} \leq \sup_{c \in int_1[\mathcal{S}_1]} \frac{|ext(c)| - |ext(core_1^+(c))|}{2 \cdot |\mathcal{G}|}$$

In other word:

$$\overline{specificity}(\mathcal{S}_2) \leq \overline{specificity}(\mathcal{S}_1)$$

This conclude the demonstration. □

B   Additional Experiments

This section provides additional materials on the performed empirical study to evaluate the effectiveness our proposed method. For the purpose of reproducibility, we maintain a companion page[8] which includes all scripts utilized for experiments. For MCTS4DM scripts please refer to its corresponding companion page[9].

Note that, for some datasets (CMC and AUTOS), we observe that the exhaustive algorithm execution ends before RefineAndMine guarantees the retrieval of the optimal solution (i.e. second milestone), this stems from the fact that we considered mainly ordinal attributes (i.e. having a small number of dif-

Dataset	num	rows	intervals	class	α	Dataset	num	rows	intervals	class	α
ABALONE_02_M	2	4177	56×10^6	M	0.37	CREDITA_02_+	2	666	1×10^9	+	0.45
ABALONE_03_M	3	4177	74×10^9	M	0.37	CREDITA_03_+	3	666	11×10^{12}	+	0.45
ABALONE_04_M	4	4177	220×10^{15}	M	0.37	CREDITA_04_+	4	666	3×10^{15}	+	0.45
ABALONE_05_M	5	4177	252×10^{21}	M	0.37	GLASS_02_1	2	214	161×10^6	1	0.33
AUTOS_05_0	5	199	46×10^{12}	0	0.32	GLASS_04_1	4	214	5×10^{15}	1	0.33
AUTOS_06_0	6	199	786×10^{15}	0	0.32	GLASS_06_1	4	214	97×10^{21}	1	0.33
BREAST_07_4	7	683	1×10^{12}	4	0.35	HABERMAN_03_2	3	306	47×10^6	2	0.26
BREAST_09_4	9	683	3×10^{15}	4	0.35	SONAR_03_R	3	208	4×10^{12}	R	0.47
CMC_05_1	4	1473	7×10^6	1	0.43	SONAR_04_R	4	208	78×10^{15}	R	0.47
CMC_06_1	5	1473	21×10^6	1	0.43	SONAR_05_R	5	208	1×10^{21}	R	0.47

Table 3: Benchmark datasets and their characteristics: number of numerical attributes, number of rows, number of all possible intervals, the considered class and its prevalence
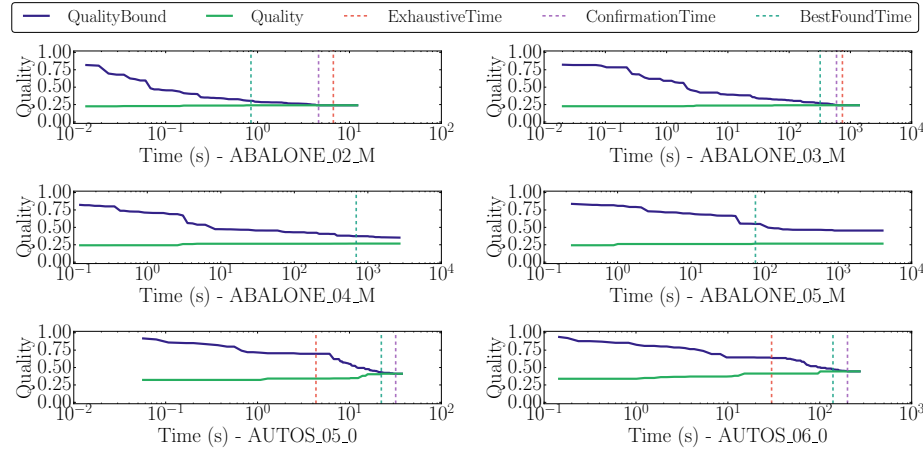

Fig. 7: Evolution over time of top pattern quality and its bounding guarantee provided by RefineAndMine (Part I).

---

[8] https://github.com/Adnene93/RefineAndMine
[9] https://github.com/guillaume-bosc/MCTS4DM

ferent values) in the above datasets. Hence involving a small number of relevant cuts making the exhaustive algorithm already a good option, since the latter makes use of the preprocessing phase computing the relevant cuts.
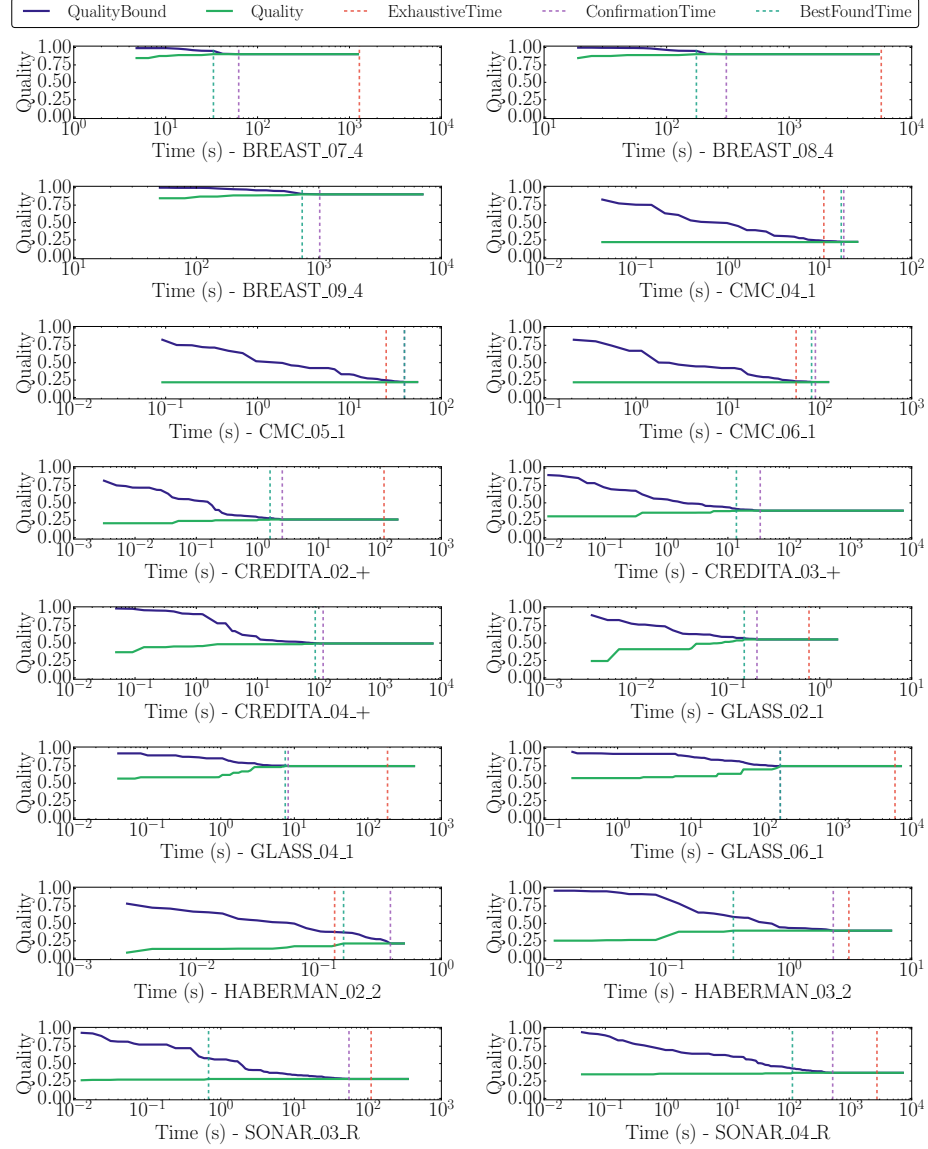

Fig. 8: Evolution over time of top pattern quality and its bounding guarantee provided by `RefineAndMine` (Part II).
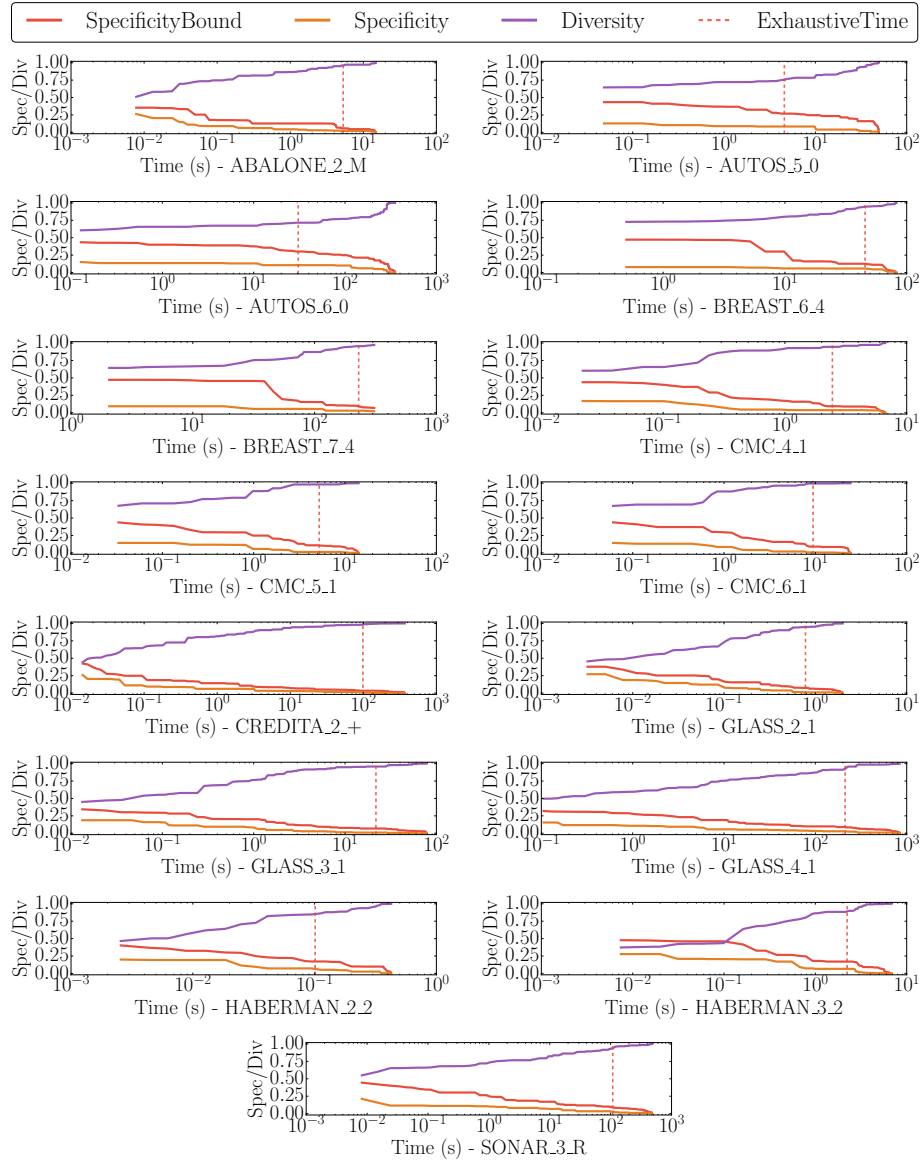
Fig. 9: Efficiency of `RefineAndMine` in terms of retrieving a diverse patterns set.
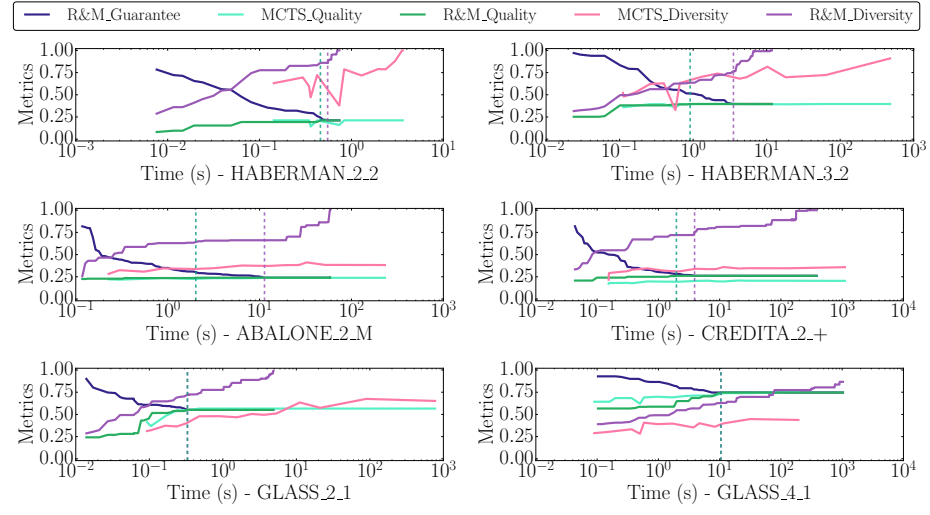
Fig. 10: Comparative experiments between `RefineAndMine` ($R\&M$) and `MCTS4DM`

C   Preliminary Notions on Order Theory

This appendix, largely inspired by [27] and [11] books, gives basic definitions and results from order theory. For any set P, $\wp(P)$ denotes the powerset of P. For a function $f : E \to F$, the image of set $A \subseteq E$ is denoted $f[A] = \{f(a) \mid a \in A\}$.

C.1   Basic definitions

Definition 2 (Poset). A partial order on a non-empty set P is a binary relation $\leq$ on P that is reflexive ($\forall x \in P : x \leq x$), transitive ($\forall x, y, z \in P : x \leq y, y \leq z \Rightarrow x \leq z$), and antisymmetric ($\forall x, y \in P : x \leq y$ and $y \leq x \Rightarrow x = y$). The pair $(P, \leq)$ is called a **partially ordered set** or **poset**. Two elements x and y from P are said to be **comparable** if $x \leq y$ or $y \leq x$. Otherwise, they are said to be **incomparable** and we denote $x \parallel y$.

Note 1. In what follows $(P, \leq)$ denotes a poset and $S \subseteq P$ denotes an arbitrary subset of P.

Definition 3 (Direct neighbor). Let $a, b \in P$, element a is said to be *direct lower neighboor* of element b (dually, element b is called *direct upper neighboor* of element a) and we write $a \prec b$ iff:

$$a < b \ \text{ and } \ \nexists c \in P : a < c < b$$

Note that $<$ is simply the relation $\leq$ from which we omit reflexivity (i.e. $a < b \Leftrightarrow a \leq b, a \neq b$).

Note 2. Any *finite poset* $(P, \leq)$ can be represented by a **line diagram** (**Hasse diagram**) where:

- Elements of P are represented by small symbols in the plane.
- If $a, b \in P$ and $a \prec b$ (a lower neighboor of b), the symbol corresponding to a is depicted below the symbol corresponding to b (order is read bottom-up) and a segment of line is drawn between them.

Fig. 11 **(1)** depicts the Hasse diagram of the poset $(\wp(\{a, b, c\}), \subseteq)$.

Definition 4 (Upper and Lower bound). Let p be an element in P. p is said to be an **upper bound** of S if $\forall s \in S : s \leq p$. Dually, p is said to be a **lower bound** of S if $\forall s \in S : p \leq s$. The set of all upper (resp. lower) bounds of S in P is denoted by S^u (resp. S^ℓ). They are given by:

$$S^u = \{p \in P \mid (\forall s \in S)\, s \leq p\} \quad S^\ell = \{p \in P \mid (\forall s \in S)\, p \leq s\}$$

Note that, particularly, $\emptyset^\ell = \emptyset^u = P$.

Definition 5 (Smallest and Greatest element). An element $s^* \in S$ is said to be the **smallest** element in S **iff**

$$\forall s \in S : s^* \leq s$$

Dually, An element $s^* \in S$ is said to be the **greatest** element in S **iff**

$$\forall s \in S : s \leq s^*$$

The smallest and the greatest elements in S are **unique** if they **exist**.

Definition 6 (Supremum and infimum). If the set of upper bounds S^u has it smallest element it is called the **supremum** or the **join** of S and it is denoted by $sup(S)$ or $\bigvee S \, (= \bigwedge S^u)$. Dually, if the set of lower bounds S^ℓ has the greatest element, it is called the **infimum** or the **meet** of S and it is denoted by $inf(S)$ or $\bigwedge S \, (= \bigvee S^\ell)$.

Definition 7 (Lattice). A poset $(P, \leq)$ is said to be a **meet-semilattice** (resp. **join-semilattice**) if for all $x, y \in P$ the **infimum** (resp. **supremum**) of $\{x, y\}$ exists in P. A poset P is said to be a **lattice** if it is both **meet-semilattice** and **join-semilattice**. A **lattice** $(P, \leq)$ is said to be **complete** if for any subset $S \subseteq P$ has it meet $\bigwedge S$ and a join $\bigvee S$ in P (including the empty set).

Note 3. Any **finite lattice** is by definition **complete**. Moreover, any **complete lattice** $(P, \leq)$ is **bounded**. That is the bottom element $\bigwedge P = \bigvee \emptyset = \bot$ and $\bigvee P = \bigwedge \emptyset = \top$ exists in P. Note also that for $a, b \in P$ with $(P, \leq)$ a lattice, $\wedge\{a, b\} = a \wedge b$ and $\vee\{a, b\} = a \vee b$ are seen as **associative**, **commutative** and **idempotent** binary operator (i.e. lattice are seen also as algebraic structures).

Note 4. For any (finite or infinite) set E, the poset $(\wp(E), \subseteq)$ is a **complete lattice** and it is called the **Boolean lattice**. The **meet** is the set **intersection** while the **join** is the set **union**. For instance, Fig. 11 **(1)** depicts the Hasse diagram of the boolean lattice $(\wp(\{a, b, c\}), \subseteq)$. We have $\{\{a, c\}, \{a, b\}\}^\ell = \{\emptyset, \{a\}\}$ and $\{\{a, c\}, \{a, b\}\}^u = \{\{a, b, c\}\}$. Therefore $\bigwedge\{\{a, c\}, \{a, b\}\} = \{a\}$ while $\bigvee\{\{a, c\}, \{a, b\}\} = \{a, b, c\}$.

C.2   Closure and Kernel Systems

For any subset $S \subseteq P$, $(S, \leq)$ forms a **sub-poset** of $(P, \leq)$ (where the relationship $\leq$ is restricted in S). However, not any sub-poset $(S, \leq)$ keeps properties of it parent poset $(P, \leq)$. That is, a sub-poset of a lattice could be arbitrary (not a lattice). Moreover, even if $(S, \leq)$ is a complete lattice. It could have different meet and join than its parent poset. Fig. 11 shows three examples of sub-poset of the boolean lattice $(\wp(\{a, b, c, d\}), \subseteq)$ (from the second left to the right). The second **(2)** is a sub-poset that is clearly neither a meet-semilattice nor a join-semilattice. The third **(3)** is a sub-poset that is a lattice, however the new meet and the new join are different from set intersection and set union. Indeed, $\{a, b\} \wedge \{b, c\} = \emptyset$ and $\{a, b\} \vee \{b, c\} = \{a, b, c, d\}$. Finally, the forth **(4)** sub-poset inherit both meet and join. We say that such the sub-poset is a **sub-lattice**.
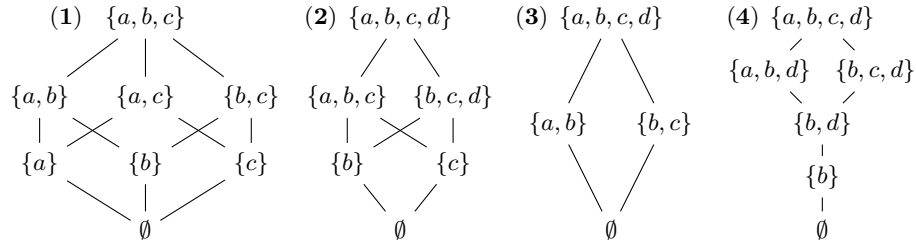
(1) $\{a,b,c\}$ **(2)** $\{a,b,c,d\}$ **(3)** $\{a,b,c,d\}$ **(4)** $\{a,b,c,d\}$

$\{a,b,d\}$ $\{b,c,d\}$

$\{a,b\}$ $\{a,c\}$ $\{b,c\}$ $\{a,b,c\}$ $\{b,c,d\}$

$\{a,b\}$ $\{b,c\}$ $\{b,d\}$

$\{a\}$ $\{b\}$ $\{c\}$ $\{b\}$ $\{c\}$

$\{b\}$

$\emptyset$ $\emptyset$ $\emptyset$ $\emptyset$

Fig. 11: From left to right *Hasse diagram* of: the lattice $(\mathcal{P}(\{a,b,c\}),\subseteq)$ **(1)**, subposet of $(\mathcal{P}(\{a,b,c,d\}),\subseteq)$ that is **not a lattice (2)**, subposet of $(\mathcal{P}(\{a,b,c,d\}),\subseteq)$ that is **a lattice (3)**, **sublattice** of $(\mathcal{P}(\{a,b,c,d\}),\subseteq)$ **(4)**.

Definition 8 (Closure and Kernel System). Let $(P,\leq)$ be a **complete lattice** such that its meet is $\bigwedge_P$ and its join is $\bigvee_P$. Let $S \subseteq P$ be a subset. Poset $(S,\leq)$ is said to be a:

– **Closure system** or a **meet-structure** on $(P,\leq)$ iff:

$$\forall A \subseteq S : \bigwedge_P A \in S$$

Poset $(S,\leq)$ is a **complete lattice** where the meet and the join are given by:

$$\forall A \subseteq S : \bigvee_S A = \bigwedge_P \left\{ s \in S \mid (\forall a \in A)\, a \leq s \right\} \quad \bigwedge_S A = \bigwedge_P A \quad \top_P \in S$$

– **Kernel system** or a **join-structure** on $(P,\leq)$ iff:

$$\forall A \subseteq S : \bigvee_P A \in S$$

Poset $(S,\leq)$ is a **complete lattice** where the meet and the join are given by:

$$\forall A \subseteq S : \bigwedge_S A = \bigvee_P \left\{ s \in S \mid (\forall a \in A)\, s \leq a \right\} \quad \bigvee_S A = \bigvee_P A \quad \bot_P \in S$$

– **Complete sublattice** of $(P,\leq)$ if it is both closure and kernel system. That is both arbitrary meets and arbitrary joins are preserved. Note that top and bottom elements of P $(\top_P, \bot_P)$ are also in S.

Note 5. Reconsider Fig. 11 **(1)** where now $(\wp(\{a,b,c\},\subseteq))$ is seen as a subposet of $(\wp(\{a,b,c,d\},\subseteq))$. Clearly, for **almost** all subsets $S \subseteq \wp(\{a,b,c\}$ the intersection and the union are preserved. However, the $(\wp(\{a,b,c\},\subseteq))$ cannot be seen as a closure system in $(\wp(\{a,b,c,d\},\subseteq))$. Indeed, $\top = \{a,b,c,d\} \notin \wp(\{a,b,c\})$. Thus, $(\wp(\{a,b,c\},\subseteq))$ is only a **kernel system** in $(\wp(\{a,b,c,d\},\subseteq))$ but not a sub-lattice.

Definition 9 (Mapping properties in a poset). A function $f : P \to P$ is said to be:

- **Extensive.** if $(\forall a \in P)\ a \leq f(a)$.
- **Contractive.** if $(\forall a \in P)\ f(a) \leq a$.
- **Monotonic.** if $(\forall a, b \in P)\ a \leq b \Rightarrow f(a) \leq f(b)$.
- **Idempotent.** if $(\forall a \in P)\ f\big(f(a)\big) = f(a)$.

The set of fix-points of a function f is given by $\{a \in P \mid f(a) = a\}$. A function f is said to be a **closure operator** if it is **extensive**, **monotonic** and **idempotent**. Dually, it is said to be a **kernel operator** if it is **contractive**, **monotonic** and **idempotent**.

Note 6. Note that there is a strong relationship between closure (resp. kernel) systems and closure (resp. kernel) operators. For instance, if f is closure (resp. kernel) operator in a complete lattice $(P, \leq)$ then $(f[P], \leq)$ is a closure (resp. kernel) system in $(P, \leq)$. Moreover, $f[P]$ is the set of **fix-points** of f, that is $f[P] = \{p = f(p) \mid p \in P\}$ (if f is a closure operator, elements of $f[P]$ are called **closed elements**). On the other hand, one can always associates a closure operator to a closure system $(S, \leq)$. Indeed, the function

$$\phi : P \to P, P \mapsto \phi(p) = \bigwedge \{s \in S \mid p \leq s\}$$

is a closure operator in $(P, \leq)$ with $\phi[P] = S$. Dually, one can also associates a kernel operator to a kernel system $(S, \leq)$. The function

$$\psi : P \to P, P \mapsto \psi(p) = \bigvee \{s \in S \mid s \leq p\}$$

is a kernel operator in $(P, \leq)$ with $\psi[P] = S$.

There is an elegant theorem (see theorem 3) when $(S, \leq)$ is a complete sub-lattice of $(P, \leq)$. This theorem is the base one we use in the current paper.

Theorem 3 (Theorem 4.1 in [9]). *Let $(P, \leq)$ be a complete lattice and let $(S, \leq)$ be a complete sub-lattice of $(P, \leq)$. Let be the two functions:*

$$\phi : P \to P, P \mapsto \phi(p) = \bigwedge \{s \in S \mid p \leq s\}$$

$$\psi : P \to P, P \mapsto \psi(p) = \bigvee \{s \in S \mid s \leq p\}$$

*Operators ϕ and ψ are respectively **join-preserving closure operator** and **meet-preserving kernel operator**. That is, for all $A \subseteq P$, we have:*

$$\phi\left(\bigvee A\right) = \bigvee \phi[A] \qquad \psi\left(\bigwedge A\right) = \bigwedge \psi[A]$$

Moreover, the set of fix-points is $S = \phi[P] = \psi[P]$.

Note 7. Please note that the closure (resp. kernel) operator is, still, **not necessarily** meet-preserving (resp. join-preserving). For instance consider the complete sub-lattice of $(\wp(\{a, b, c, d\}), \subseteq)$ depicted in Fig. 11 **(4)**. One can show that the associated closure operator is **not meet-preserving**. Indeed, we have $\phi(\{a, b\}) = \{a, b, d\}$ and $\phi(\{b, c\}) = \{b, c, d\}$, but:

$$\phi(\{a, b\} \cap \{b, c\}) = \phi(\{b\}) = \{b\} \neq \{b, d\} = \{a, b, d\} \cap \{b, c, d\} = \phi(\{a, b\}) \cap \phi(\{b, c\})$$

D   Complete lattice of p-dimensional hyper-rectangles

Definition 10. *An **interval** $I \subseteq \mathbb{R}$ is a **convex subset** of $\mathbb{R}$, that is:*

$$\forall x, y \in I, \forall z \in \mathbb{R} : x \leq z \leq y \implies z \in I$$

*The **partially ordered set (poset)** of all intervals in $\mathbb{R}$ is denoted $(\mathfrak{C}(\mathbb{R}), \subseteq)$ [11] and is a **closure-system** in $(\wp(\mathbb{R}), \subseteq)$. That is, the intersection $\bigcap$ of arbitrary set of intervals is an interval in $\mathfrak{C}(\mathbb{R})$.*

The poset $(\mathfrak{C}(\mathbb{R}), \subseteq)$ of intervals is a **complete lattice** where the **meet** $\bigwedge$ is set intersection $\bigcap$ while the **join** $\bigvee$ is the **convex hull** of there **union**, formally:

$$\forall S \subseteq \mathfrak{C}(\mathbb{R}) : \bigwedge S \triangleq \bigcap S$$
$$\bigvee S = \bigcap \left\{ X \in \mathfrak{C}(\mathbb{R}) \mid \bigcup S \subseteq X \right\} = ch \left(\bigcup S \right)$$

There is different types of intervals in $\mathfrak{C}(\mathbb{R})$. For example: $(-\infty, a]$, $(-\infty, a)$, (a, b), $[a, b]$, $[a, b)$,... with $a, b \in \mathbb{R}$. Note that the top (greatest) element of $(\mathfrak{C}(\mathbb{R}), \subseteq)$ is $\mathbb{R}$ while the bottom (smallest) element is $\emptyset$. For two intervals $[a_1, b_1]$ and $[a_2, b_2]$ in $(\mathfrak{C}(\mathbb{R}), \subseteq)$ with $a < b$, we have:

$$[a_1, b_1] \wedge [a_2, b_2] = [sup(a_1, a_2), inf(b_1, b_2)]$$
$$[a_1, b_1] \vee [a_2, b_2] = [inf(a_1, a_2), sup(b_1, b_2)]$$

With sup and inf refer to the greatest and smallest element in $\mathbb{R}$ wrt. natural order $\leq$. This computation of $\wedge$ and $\vee$ is generalizable for other types of intervals.

Definition 11. *Let $p \in \mathbb{N}^*$, an p-dimensional hyper-rectangle $H \subseteq \mathbb{R}^p$ is the result of the product of m intervals in $\mathfrak{C}(\mathbb{R})$. Formally:*

$$H = \prod_{k=1}^{p} I_k \quad with \ \ \forall k \in \{1..m\} : I_k \in \mathfrak{C}(\mathbb{R})$$

*The set $\mathcal{H}_p$ denotes the set of all possible m-dimensional hyper-rectangles in $\mathbb{R}^p$. The poset $(\mathcal{H}_p, \subseteq)$ is again a **closure-system** in $(\wp(\mathcal{R}^p), \subseteq)$.*

The fact of $(\mathcal{H}_p, \subseteq)$ is a complete lattice which meet is **set-intersection** is "inherited" [10] from the **direct product** of m complete lattice $(\mathfrak{C}(\mathbb{R}), \subseteq)$ (the direct product of m complete lattice is a complete lattice [11, p. 130]). We have:

$$\forall S \subseteq \mathcal{H}_p : \bigwedge S \triangleq \bigcap S$$
$$\bigvee S = \bigcap \left\{ X \in \mathcal{H}_p \mid \bigcup S \subseteq X \right\}$$

---

[10]In fact, one can use an order-embedding $f : (\mathfrak{C}(\mathbb{R})^p, \leq) \hookrightarrow (\wp(\mathbb{R}^p), \subseteq)$ (where $\leq$ is the **product order** [27, p. 3]) such that $f((I_k)_{1 \leq k \leq p}) = \prod_{k=1}^{p} I_k$ to build properly the complete lattice $f[\mathfrak{C}(\mathbb{R})^p] = \mathcal{H}_p$.

Thus, the meet $\bigwedge$ and the join $\bigvee$ are formed component-wise. For instance, for hyper-rectangles $H_j = \prod_{k=1}^{p} I_k^j$ with $j \in \{1, 2\}$ and $I_k^j = [a_k^j, b_k^j]$ we have:

$$H_1 \wedge H_2 = \prod_{k=1}^{p} \left(I_k^1 \wedge I_k^2 \right) = \prod_{k=1}^{p} [sup(a_k^1, a_k^2), inf(b_k^1, b_k^2)]$$

$$H_1 \vee H_2 = \prod_{k=1}^{p} \left(I_k^1 \vee I_k^2 \right) = \prod_{k=1}^{p} [inf(a_k^1, a_k^2), sup(b_k^1, b_k^2)]$$

Clearly, $\emptyset$ and $\mathbb{R}^p$ are respectively bottom and top element in $(\mathcal{H}_p, \subseteq)$.

Remark 1. In this paper, we deal with the **dual lattice** of $(\mathcal{H}_p, \subseteq)$ denoted $(\mathcal{H}_p, \sqsubseteq)$. The order $\sqsubseteq$ is given by the **inverse order** $\supseteq$ while the meet $\bigsqcap \triangleq \bigvee$ and the join $\bigsqcup \triangleq \bigwedge$ are inverted. The top element become $\top \triangleq \emptyset$ while the bottom element become $\bot \triangleq \mathbb{R}^p$.

E   Closed Interval Pattern Mining

Algorithm 2 describe the closed interval pattern mining algorithm as proposed in [18]. We slightly change the method to compute the closed on the positives (`cotp` for short) intervals patterns. Please refer to [19,18] for more details.

Algorithm 2: `MinIntChange`

Input: $(\mathcal{G}, \mathcal{M})$ a numerical datasets with $\{\mathcal{G}^+, \mathcal{G}^-\}$ partition of $\mathcal{G}$ and $|\mathcal{M}| = p$
$\qquad$ $C = (C_k)_{1 \leq k \leq p}$ cut points sets per dimension
Output: `cotp` patterns $c \in \mathcal{D}_C$ verifying $d \sqsubseteq c$

1 **procedure** $\mathtt{MinIntChange}_C(d, i, f)$
2 $\quad$ **yield**(d) $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ `// yield cotp pattern`
3 $\quad$ **for** $j \in \{i, ..., p\}$ **do**
4 $\quad\quad$ $f_{start} \leftarrow 0$ **if** $j > i$ **else** f
5 $\quad\quad$ **for** $f_{new} \in \{f_{start}, 1\}$ **do**
6 $\quad\quad\quad$ $d_{new} \leftarrow \mathbf{minChange}_C(d, j, f_{new})$
7 $\quad\quad\quad$ $c \leftarrow \mathbf{int}_C\big(\{g \in ext(d_{new}) \cap \mathcal{G}^+ \mid \mathcal{M}(g) \in d\}\big)$
8 $\quad\quad\quad$ **if** $c \neq \top$ **and** $(\forall k \in \{1, ..., j-1\})\ c.I_k = d.I_k$ **then**
9 $\quad\quad\quad\quad$ $\mathtt{MinIntChange}_C(c, j, f_{new})$

10 **function** $\mathbf{minChange}_C(d, j, f)$
11 $\quad$ $d_{new} \leftarrow d$ $\qquad\qquad\qquad\qquad\qquad$ `//` $d = \prod_{1 \leq i \leq k}[a_k, b_k) \in \mathcal{D}_C$
12 $\quad$ **if** $f = 0$ **then**
13 $\quad\quad$ $d_{new}.I_j \leftarrow [next(a_j, C_j), b_j)$ $\quad$ `//` $next(a_j, C_j) = inf\{c \in C_j \mid a_j < c\}$
14 $\quad$ **else**
15 $\quad\quad$ $d_{new}.I_j \leftarrow [a_j, prev(b_j, C_j))$ $\qquad$ `//` $prev(b_j, C_j) = sup\{c \in C_j \mid c < b_j\}$
16 $\quad$ **return** d_{new} $\qquad\qquad$ `//` $d_{new}.I_j$ `could be empty (i.e.` $d_{new} = \top = \emptyset$`)`
17 **function** $\mathbf{int}_C(S)$
18 $\quad$ **for** $k \in 1, ..., p$ **do**
19 $\quad\quad$ $a_k \leftarrow sup\Big\{c \in C \mid c \leq inf\{m_k(g) \mid g \in S\}\Big\}$ $\qquad$ `//` $sup(\emptyset) = -\infty$
20 $\quad\quad$ $b_k \leftarrow inf\Big\{c \in C \mid c > sup\{m_k(g) \mid g \in S\}\Big\}$ $\qquad$ `//` $inf(\emptyset) = +\infty$
21 $\quad\quad$ $d.I_k \leftarrow [a_k, b_k)$ **if** $a_k \in C$ **else** $(-\infty, b_k)$
22 $\quad$ **return** d
23 $\mathtt{MinIntChange}(d, 1, 0)$ $\quad$ `// Start by a` minLeftChange `on the` 1st `attribute`