Anytime Subgroup Discovery in High Dimensional Numerical Data

Romain Mathonat, Diana Nurbakova, Jean-François Boulicaut, Mehdi Kaytoue

Anytime Subgroup Discovery in High Dimensional Numerical Data

Romain Mathonat[1,4],bon Diana Nurbakova[2], Jean-François Boulicaut[2], Mehdi Kaytoue[1,2]

[1] Infologic, 99 avenue de Lyon, F-26500, Bourg-Lès-Valence, France
[2] Univ Lyon, INSA Lyon, CNRS, LIRIS UMR 5205, F-69621, Lyon, France
[3] Univ Lyon, CNRS, LIRIS UMR 5205, F-69622, Lyon, France
[4] romain.mathonat@gmail.com

***Abstract*—Subgroup discovery (SD) enables one to elicit patterns that strongly discriminate a class label. When it comes to numerical data, most of the existing SD approaches perform data discretizations and thus suffer from information loss. A few algorithms avoid such a loss by considering the search space of every interval pattern built on the dataset numerical values and provide an "anytime" property: at any moment, they are able to provide a result that improves over time. Given a sufficient time/memory budget, they may eventually complete an exhaustive search. However, such approaches are often intractable when dealing with high-dimensional numerical data, for instance, when extracting features from real-life multivariate time series. To overcome such limitations, we propose `MonteCloPi`, an approach based on a bottom-up exploration of numerical patterns with a Monte Carlo Tree Search. It enables to have a better exploration-exploitation trade-off between exploration and exploitation when sampling huge search spaces. Our extensive set of experiments proves the efficiency of `MonteCloPi` on high-dimensional data with hundreds of attributes. We finally discuss the actionability of discovered subgroups when looking for skill analysis from *Rocket League* action logs.**

***Index Terms*—Numerical Subgroup Discovery, Monte Carlo Tree Search**

I. Introduction

Exploratory data analysis is an important task to support knowledge discovery in data. It consists in automatically highlighting hidden structures and relations with data mining algorithms. In the presence of labeled data, many techniques have been proposed for eliciting patterns that characterize a given label. They were introduced using different terms: subgroups, emerging patterns, hypotheses, etc. [1]. In this article, we use the terminology of Subgroup Discovery (SD) [2]. Such patterns also called subgroup descriptions provide understandable and interpretable hypotheses on the phenomena that "produced" the data. Not only these patterns give insights to domain experts but also they can be exploited to build interpretable, say transparent, pattern-based prediction models. Let us take a timely example when we have an object/attribute data table containing both numerical and categorical patient data (age, sex, weight, blood pressure, glycated hemoglobin level HbA1c, etc., and their COVID-19 test results) with a label indicating the severeness level of patient's state. SD can lead to discovery of patterns that can characterise each of these states. For instance, we may discover that a pattern like "$Sex = Male \wedge (52 \leq age \leq 64) \wedge (HbA1c \geq 6.3) \wedge COVID\text{-}19 = True$" discriminates well the label "$Severe\ Case$". A quality measure would indeed quantify how the defined subgroup deviates from the norm, i.e., the distribution of labels in the whole data. Roughly speaking, such a pattern may emphasize that, in this data, even rather "young" male seniors who are close to a diabetes diagnosis tend to develop severe cases of COVID-19 disease.

Among the various applications of SD in numerical data, we investigate more precisely its added-value when processing *time series*. Indeed, we can transform the original time series into potentially high-dimensional numerical data, and take advantage of the interpretability of numerical patterns. For instance, we consider here a use case about the video game data analytics (e-sport). Game activity usually produces sequences of time-tagged data points, i.e., time series. One interesting challenge is then to design efficient and interpretable prediction models for a variety of tasks (e.g., predicting player behaviour for game customization, user profiling for adaptive skill mastery training, etc.).

We consider hereafter only numerical data labeled with discrete values. In other terms, our subgroup descriptions are given by the so-called interval patterns. Mining interval patterns can be seen as finding multi-dimensional intervals containing objects (points) in an n-dimensional Euclidean space. Even though the number of interval patterns appears theoretically infinite, thanks to the concept of *closure*, it can be restricted to a finite set when interval borders correspond to data points. However, even finite, the set of patterns is too large for an exhaustive exploration. To deal with that, different paradigms have been proposed.

Efficient exhaustive algorithms designed for categorical data can be applied on discretized data or directly on numerical data with greedy discretization during the exploration (e.g., `SD-MAP` [3]). Also, greedy algorithms such as `Beam Search` (e.g., `DSSD` [4]) are widely used. Recently, two appealing anytime algorithms have been proposed, attempting to take the most of both paradigms: a best first search that can produce a solution at anytime if interrupted, whose quality improves over time and eventually performs an exhaustive

search if let running long enough. First, the MCTS4DM algorithm exploits a Monte Carlo Tree Search (MCTS, see [5] for a survey) to explore the whole interval pattern search space focusing on both unvisited and promising parts, i.e., performing an exploration-exploitation trade-off [6]. It was the first investigation of the usage of MCTS framework for pattern discovery. Second, the algorithm Refine&Mine proposes an exhaustive interval pattern search that can be operated on increasingly refined data discretizations [7]. In the case of high-dimensional numerical data (hundreds of attributes), both MCTS4DM and Refine&Mine are inefficient. MCTS4DM considers interval patterns in a top-down fashion, sampling the search space with random draws that are costly. Thus, it hardly reaches lower parts of the search space where interesting patterns can be hidden. Refine&Mine provides guarantees and runs fast on very rough data representations, but not when it reaches some finer representations for high-dimensional numerical data.

We want to use the MCTS framework to control the search for subgroups but, in contrast to MCTS4DM, we propose to explore the search space of *extents* in a *bottom-up way*, restricting the search space by exploring only *closed on the positive* elements to reduce its size without pruning interesting patterns. We also define new policies for the EXPAND and ROLLOUT steps of MCTS to deal with high-dimensional data. Our contributions can be summarized as follows:

- We extend the previous works on the *closed on the positive* (COTP) interval patterns [7] by defining and proving a key property, allowing to explore COTPs only, without loss of the relevance of the search space for discriminating the target class.
- We propose a MCTS-based algorithm called MonteCloPi that performs a bottom-up best first search exploration based on the above mentioned mathematical property.
- Through an extensive set of experiments, we demonstrate that MonteCloPi can be applied to collections of multivariate time series of both equal and different lengths and show that the extracted interval patterns are of high value.
- We provide a video game analytics use case, showing how particular strategies can be detected and interpreted. We demonstrate that MonteCloPi is able to classify multivariate time series of different lengths in real time, gaining a competitive advantage over 1NN DTW [8].

We formalize our SD task in Section II. We define COTP interval patterns and discuss their properties in Section III. Section IV describes our algorithm before a survey of related work in Section V. Both quantitative and qualitative experimental studies are reported in Sections VI and VII, respectively. Section VIII concludes.

II. Subgroup Discovery in Numerical Data

Objects of a numerical dataset with n attributes are points in the n-dimensional Euclidean space. Interval patterns are multi-dimensional intervals, i.e., hyper-rectangles with sides parallel to the plane axes . The extent of an interval pattern is the set of data points that fall within this rectangle. Hence, interval patterns are partially ordered by interval inclusion, and they form a lattice, structuring the search space [9].

Definition 1 (Numerical dataset $\mathcal{D}(\mathcal{O}, \mathcal{A}, \mathcal{C})$)**.** *Let $\mathcal{O}$, $\mathcal{A}$ and $\mathcal{C}$ be respectively a set of objects, a set of numerical attributes, and a set of class labels. When an object o takes a value v for an attribute $attr$, we write $attr(o) = v$. The finite domain of a numerical attribute $attr \in \mathcal{A}$ in a given dataset is denoted by $Dom(attr) = \{attr(o) \mid \forall o \in O\}$ The mapping $f : \mathcal{O} \mapsto \mathcal{C}$ associates each object to a unique class label.*

Definition 2. (Interval pattern, extent, support, frequency) *An interval pattern of length n is a vector of intervals $p = \langle I_1, I_2 ..., I_n \rangle$, with $I_i = [a_i, b_i]$, $a_i, b_i \in Dom(attr_i)^2$, $a_i \leq b_i$. The extent of a pattern p is $ext(p) = \{o \in \mathcal{O} \mid attr_i(o) \in I_i, \ \forall I_i \in p\}$. We say that p covers objects of its extent. We have $supp(p) = |ext(p)|$ and $freq(p) = supp(p)/|\mathcal{O}|$.*

Definition 3. (Interval pattern ordering and search space) *$(\mathcal{P}, \sqsubseteq)$ is the poset of all interval patterns ordered with inclusion: $p_1 \sqsubseteq p_2 \iff [a_i^2, b_i^2] \sqsubseteq [a_i^1, b_i^1], \forall i \leq n, [a_i^j, b_i^j]$ referring to the i^{th} interval of the pattern p_j. It is a finite lattice with $\prod_{i=1}^{n} \frac{|Dom(attr_i)| \times (|Dom(attr_i)|+1)}{2}$ elements.*

A subgroup is composed of a pattern (i.e., an intent), its extent, and a score that reflects its interestingness, such as, for instance, the famous Weighted Relative Accuracy ($WRAcc$) [10]. When the context is clear, we use the terms pattern and subgroup interchangeably.

Definition 4. (Subgroup and quality measure) *Given an interval pattern p, $(p, ext(p))$ is a subgroup. Let $\mathcal{S}$ be the set of all possible subgroups. A quality measure φ is an application $\varphi : \mathcal{S} \to \mathbb{R}$ that maps every subgroup from $s \in \mathcal{S}$ to a real number reflecting its interestingness.*

Definition 5. (Weighted Relative Accuracy) *Given an interval pattern p, a dataset $\mathcal{D}$, and $\mathcal{D}_c$ being the set of its elements labeled with class c, $WRAcc(p, \mathcal{D}, \mathcal{D}_c) = \frac{supp(p,\mathcal{D})}{|\mathcal{D}|} \times \left(\frac{supp(p,\mathcal{D}_c)}{supp(p,\mathcal{D})} - \frac{|\mathcal{D}_c|}{|\mathcal{D}|} \right)$. Its range is $[-0.25, 0.25]$, the closer to the bounds the better.*

Example 1. *Table I provides a numerical dataset with 4 objects, 6 attributes and 2 labels. An interval pattern is $p = \langle [1,2], [4,5], [3,4], [7,8], [5,7], [8,9] \rangle$ and its support is $supp(p, \mathcal{D}) = 2$. Accordingly, $(o1, o3)$ is a subgroup w.r.t. p, and $WRAcc(p, \mathcal{D}, \mathcal{D}_+) = 0.25$. As example of a pair belonging to the ordering relation, we have $\langle [1,4], [2,4] \rangle \sqsubseteq \langle [1,3], [3,3] \rangle$.*

The maximum value the WRAcc can take on a dataset $\mathcal{D}$ for a class c, $\mathcal{D}_c$ being the set of its elements labeled by class c, is $WRAcc_{max}(c) = \frac{|\mathcal{D}_c|}{|\mathcal{D}|} \left(1 - \frac{|\mathcal{D}_c|}{|\mathcal{D}|}\right)$.

Proof.

$$WRAcc(p, \mathcal{D}, \mathcal{D}_c) = \frac{supp(p, \mathcal{D})}{|\mathcal{D}|} \times \left(\frac{supp(p, \mathcal{D}_c)}{supp(p, \mathcal{D})} - \frac{|\mathcal{D}_c|}{|\mathcal{D}|} \right)$$

TABLE I: Toy dataset

ID	$attr1$	$attr2$	$attr3$	$attr4$	$attr5$	$attr6$	$class(.)$
o1	1	5	3	8	7	9	+
o2	8	9	0	1	2	1	-
o3	2	4	4	7	5	8	+
o4	5	5	1	2	3	2	-

Note that $supp(p, \mathcal{D}) = supp(p, \mathcal{D}_c) + supp(p, \overline{\mathcal{D}_c})$. We then have:

$$WRAcc(p, \mathcal{D}, \mathcal{D}_c) = \frac{1}{|\mathcal{D}|} \times \left(supp(p, \mathcal{D}_c)(1 - \frac{|\mathcal{D}_c|}{|\mathcal{D}|}) - supp(p, \overline{\mathcal{D}_c}) \frac{|\mathcal{D}_c|}{|\mathcal{D}|} \right)$$

To maximise the $WRAcc$, the rightmost term must be minimized. This is the case for $supp(p, \overline{\mathcal{D}_c}) = 0$: it means the pattern does contain only element with positive class. Notice also that $supp(p, \mathcal{D}) = \lfloor \beta |\mathcal{D}_c| \rfloor$, with $0 \leq \beta \leq 1$ as the number of elements having the target class must be a subset of all elements having the target class. We then have:

$$WRAcc_{max}(\mathcal{D}, \mathcal{D}_c) = \beta \frac{|\mathcal{D}_c|}{|\mathcal{D}|} \left(1 - \frac{|\mathcal{D}_c|}{|\mathcal{D}|} \right)$$

Directly, β needs to be set to 1 to maximize the $WRAcc$. ■

Note that the minimal and maximal values of the $WRAcc$ depend on the proportion of the target class in the dataset. Using directly its value to assess its discriminating power could then bias user interpretation. For example, a $WRAcc$ of 0.09 for a class which is represented at 10% would mean that the pattern covers all elements having this class and only them, i.e., it would be somehow the best pattern. But on a dataset with a representation of this class at 50%, it would be much less relevant. We then propose to normalize the $WRAcc$.

Definition 6. (NORMALIZED WEIGHTED RELATIVE ACCURACY ($NWRAcc$)) *$NWRAcc(p, c)$ takes its values in $[-1, 1]$ and is defined by:*

$$\frac{WRAcc(p, \mathcal{D}, \mathcal{D}_c)}{WRAcc_{max}(\mathcal{D}, \mathcal{D}_c)} = \frac{WRAcc(p, \mathcal{D}, \mathcal{D}_c)}{\frac{|\mathcal{D}_c|}{|\mathcal{D}|} \left(1 - \frac{|\mathcal{D}_c|}{|\mathcal{D}|} \right)}$$

$NWRAcc$ can be used to compare patterns that discriminate classes with different sizes (see supplementary materials for proof).

Definition 7 (Non θ-redundant patterns)**.** *A set of patterns $\mathcal{S}_p \subseteq \mathcal{S}$ is non θ-redundant if given $\theta \in [0; 1]$ and $\forall p1, p2 \in \mathcal{S}_p$, where $p_1 \neq p_2$, we have: $sim(p_1, p_2) \leq \theta$, where sim is a similarity function such as the classical Jaccard index:*

$$sim(p_1, p_2) = \frac{|ext(p_1) \cap ext(p_2)|}{|ext(p_1) \cup ext(p_2)|}.$$

Problem (Non-redundant subgroup set discovery). *Given a numerical dataset $\mathcal{D}$, a target class c, an integer k, a quality measure φ, a similarity measure sim limited by $\theta \in [0; 1]$, find the set of at most k interval patterns that are the best w.r.t φ and non θ-redundant.*

Note that this problem is NP-Hard as proven in [11].

III. CLOSED ON THE POSITIVE INTERVAL PATTERNS

It has been shown that Problem the non-redundant subgroup set discovery problem is equivalent to the discovery of *closed on the positive* patterns (COTP): [12] addresses the general case while [7], [11] consider the discovery of *closed on the positive interval patterns* within numerical data. The key idea is that any sub-interval of a COTP will drop at least one positive object, thus, reducing the *true positive rate*, hence many measures such as the $WRAcc$.

Definition 8 (Closed pattern)**.** *An interval pattern p is closed iff $\nexists p'$ s.t. $p \sqsubset p'$, $ext(p) = ext(p')$.*

Definition 9. (CLOSED ON THE POSITIVE INTERVAL PATTERN)*Let $ext^+(p)$ be the set of objects labeled with $+$ for an interval pattern p. p is said to be a closed on the positive iff $\nexists p'$ s.t. $p \sqsubset p'$, $ext^+(p) = ext^+(p')$.*

Definition 10 (MEET operator [9])**.** *The set of all interval patterns form a lattice $(\mathcal{P}, \sqsubseteq)$. For $\forall p, q \in \mathcal{P}^2$, the MEET operator $\sqcap$ is defined as $\sqcap(p, q) = \langle I_1, ..., I_n, \rangle$ with $I_i = [min(a_i^p, a_i^q), max(b_i^p, b_i^q)]$. For example $\langle [1, 2], [6, 7] \rangle \sqcap \langle [1, 1], [8, 9] \rangle = \langle [1, 2][6, 9] \rangle$.*

Theorem 1. *The MEET of two COTP interval patterns is a COTP.*

Proof. For any interval pattern p' s.t. $p \sqsubset p'$, where p is the MEET of two closed on the positive interval patterns, we have a restriction (i.e., a smaller interval) on at least one interval compared to p. As bounds of those intervals are extracted from values of COTP patterns, i.e., they are as "tight" as possible, any restriction on it would lead to a decrease of the extent of positive elements of p. By definition, p is then a COTP interval pattern. Note that any "widening" of an interval would create an interval pattern p''. If p'' could have the same extent as p, we would have $p \sqsubset p''$ and $extent^+(p) = extent^+(p')$, which is a contradiction with the definition of a COTP pattern for p: the MEET is unique by definition. ■

IV. ALGORITHM MONTECLOPI

Classical lattice exploration methods, like `Beam-Search`, are top-down by nature, going from the most general description (i.e., pattern) down in the search space through successive refinements. In contrast, we suggest to use a *bottom-up* strategy. The idea is to consider positive objects of the database, i.e., objects labeled with the target class, and to compute their MEET with other positive objects, successively climbing the lattice of COTP, containing discriminative patterns. We aim at creating patterns that are the most restrictive descriptions that cover a set of chosen positive elements, i.e., they are by definition discriminating the positive class. We then use a Monte Carlo Tree Search (MCTS) strategy. The idea is to iteratively explore the search space using sampling and provide an *anytime* property. MCTS also guarantees the exhaustiveness on the lattice it explores, if given enough time budget.

Our algorithm is called `MonteCloPi` (MONTE carlo tree search for CLOsed on the PosItives). MCTS typically consists

of 4 steps (i.e., SELECT, EXPAND, ROLLOUT, UPDATE) that are repeated successively. Its idea is to grow an asymmetric tree following an exploration-exploitation trade-off to quickly find interesting areas of the search space. Hereafter, we explain these generic steps and how we instantiate them to our problem. The schema describing `MonteCloPi` is given in Fig. 1, and its pseudocode is given in Alg. 1. Each node of the tree is composed of a set of database objects, i.e., its extent, and a pattern covering those objects.

A. Step 1 SELECT

The first step consists in selecting a non fully-expanded node (i.e., new nodes can be created as its children). The idea is to choose the most relevant area of the search space to explore. We use the UCB function (see, e.g., [5]) to guide us through the tree, achieving an exploration-exploitation trade-off. We begin with the $\perp$ node that has all database objects as children. We compute the UCB of all its children and select the one maximizing it. We perform this step recursively until reaching a non fully-expanded node. UCB gives a better score to nodes (i.e., patterns with their extent) that are good patterns, but also to nodes that are less explored.

We use here $UCB(i) = \bar{x}_i + \sqrt{\frac{2ln(N)}{N_i}}$, with x_i being the mean quality of children node, N the number of times the current node has been chosen, and N_i the number of times the i^{th} child has been chosen.

In Fig. 1, the initial bottom node $\perp$ is given in blue. It is fully expanded, so the SELECT procedure will choose its best child w.r.t. UCB. Suppose it is the node "3" and it has not been fully expanded, then it is selected (given in red).

B. Step 2 EXPAND

Then, the selected node is expanded. This step consists in taking a positive element, i.e., an object labeled with the target class, transforming it to an interval pattern, and computing its MEET with the interval pattern of the current node. Next, the extent of this MEET is computed: we create a pattern with the most restrictive description that covers elements from the selected node plus this positive element. The transformation step from an object to a pattern is straightforward: each attribute $attr_i(o)$ becomes an interval $[attr_i(o), attr_i(o)]$.

In Fig. 1, we take the node covering the positive element "3", add the positive element "2" (node "2,3"). We then compute the MEET of their patterns, its extent, and its quality: it covers elements "3", "2", but also "1" (node "1,2,3"). Note that the pattern obtained from an object labeled with the target class o_+ is COTP, and the first iterations of the current procedure consider single object descriptions, i.e., COTP patterns. From theorem 1, it follows that this EXPAND step creates a new COTP pattern.

Note that to deal with the case of computing an already discovered pattern, we use the permutation-unification principle as described in [6].

Algorithm 1 `MonteCloPi` algorithm

function MONTECLOPI($timeBudget$)
  $\pi \leftarrow PriorityQueue()$
  create $\perp$ initial node having all objects as children
  while within $timeBudget$ **do**
    $p_{sel} \leftarrow Select(\perp)$
    $p_{exp}, qual_{exp} \leftarrow Expand(p_{sel})$
    $p_{roll}, \Delta \leftarrow Rollout(p_{exp})$
    $Update(p_{exp}, \Delta)$
    $\pi.add(p_{exp}, qual_{exp})$
    $\pi.add(p_{roll}, \Delta)$
  end while
  return $\pi.topK()$
end function

function SELECT(p)
  while p is not $\top$ **do** ▷ $\top$ is the most general pattern
    if p is not fully-expanded **then**
      return p
    else
      $p \leftarrow BestChild(p)$
    end if
  end while
  return p
end function

function EXPAND(p)
  $o_+ \leftarrow$ *randomly choose a positive object*
  $p_+ \leftarrow IntervalPattern(o_+)$ ▷ Transform each a_i into $[a_i, a_i]$
  $p_{exp} \leftarrow meet(p, p_+)$
  return $p_{exp}, \varphi(p_{exp})$
end function

function ROLLOUT(p)
  $j \leftarrow \text{Rand}(1, n)$ ▷ Randomly pick an index of one of n intervals constituting p
  for i in $[1, j-1] \cup [j+1, n]$ **do** ▷ Exclude the j^{th} interval I_j of p, i.e. $p \setminus I_j$
    $p[i] \leftarrow [-\infty, +\infty]$
  end for
  return $p, \varphi(p)$
end function

function UPDATE(p, Δ)
  while $p \neq \perp$ **do**
    $\varphi(p) \leftarrow \frac{N(p)*\varphi(p)+\Delta}{N(p)+1}$ ▷ $N(p)$ is the number of times p has been chosen
    $N(p) \leftarrow N(p) + 1$
    $p \leftarrow parent\ of\ p$
  end while
end function

function BESTCHILD(p)
  return $\underset{p' \in \mathcal{C}(p)}{\text{argmax}}\, UCB(p, p')$ ▷ $\mathcal{C}(p)$ is the set of children of p
end function

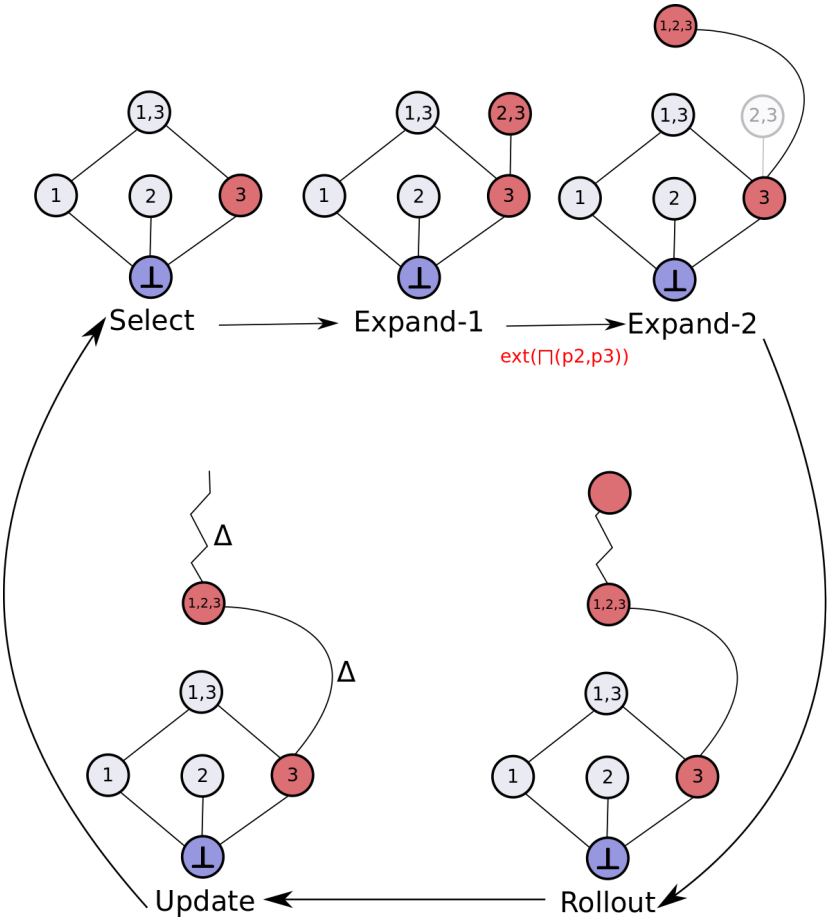

Fig. 1: Simple illustration of `MonteCloPi` steps

C. Step 3 ROLLOUT

Our ROLLOUT step consists in taking the corresponding interval pattern of the expanded node, randomly choosing an interval among n available ones (I_j), widening it by setting its value to random values from the data, and removing restrictions on all others, i.e., setting them to $[-\infty, \infty]$. It corresponds to going "up" in the search space, until reaching the first level of the search space in a top-down approach (see the top red node in Fig. 1). We then compute the quality of this pattern. Note that Monte Carlo strategies consist in drawing a large number of samples, maximizing the sampling to approximate the underlying data distribution, so they require a fast ROLLOUT policy to be efficient.

D. Step 4 UPDATE

Finally, we update parent nodes of the ROLLOUT node with its computed quality Δ. These parent nodes include the expanded node, the selected node and all nodes chosen in the SELECT step until reaching the selected node. The UCB will orient the tree exploration towards the nodes whose ROLLOUTs give good qualities, so they will be prioritized on the next iterations of MCTS, leading to the discovery of interesting patterns.

Steps 1-4 are repeated iteratively until the time budget has not been reached or until the whole search space is explored. The time complexity of `MonteCloPi` is $O(IterNum{*}n|\mathcal{D}|)$, with n the number of attributes. The memory complexity is $O(IterNum)$, as the whole tree needs to be held in memory.

Theorem 2. *`MonteCloPi` explores all COTP interval patterns if given enough time.*

Proof. The EXPAND step creates a COTP. We now need to show that $\forall c \in COTP$, c will be enumerated by our method. By construction: let c be a COTP interval pattern. If we remove positive objects from $ext(c)$ one by one, and compute the corresponding COTP interval pattern, we create a list of COTP interval patterns until we reach a COTP interval pattern containing only one object (or a set of identical objects). This list of successive COTP interval patterns is exactly the list built by our method in reverse order, by adding objects one by one and computing the MEET in EXPAND step. As a COTP interval pattern created from the MEET of two COTP interval patterns is unique (see Proposition 1), `MonteCloPi` will create c. ■

E. Dealing with time series

In many application settings, we have to deal with time series of equal or different lengths, in an univariate or a multivariate case. We can use `MonteCloPi` for mining time series. For that purpose, we can transform such a data type into numerical data similarly to [13]. Basically, a multivariate time series of fixed length dataset is composed of a set of variables X at each timestamp in T. The corresponding numerical dataset is then given by $(\mathcal{O}, X \times T, \mathcal{C})$: each numerical attribute gives the value of a variable at a given time. To deal with time series of different lengths, two modifications of the algorithm are required. First, we adapt the computation of the MEET for two interval patterns: we randomly remove intervals from the longest one to have two interval patterns of the same length, and then compute their MEET. Second, we re-define the notion of *coverage*: a time series transformed into a numerical object o of length n_{ts} is said to be covered by an interval pattern p of length n_p iif: $\exists 1 \leq i_1 \leq \ldots \leq i_{np} \leq n_{ts}$ $s.t.$ $attr_{i1}(o) \in I_1, \ldots, attr_{inp}(o) \in I_{np}$. Note that we loose here the guarantee of exhaustiveness on COTP, as the MEET operator is no more unique.

To summarize, `MonteCloPi` is an *anytime* algorithm. It explores *COTP patterns*, with a guarantee of *exhaustiveness*, if given enough time budget. It is agnostic of the quality measure. In contrast to `MCTS4DM`, it explores the search space in a bottom-up way, exploring only COTPs by redefining in an original way the EXPAND and the ROLLOUT steps.

V. More Related Work

Various subgroup discovery approaches have been proposed in the literature. Top-down approaches using tight optimistic estimates like the recent work of Millot *et al.* [14], or using those optimistic estimates to prune the search space for `Beam-Search` can be considered. However, they are dependent on the quality measure. Besides, using optimistic estimates when looking for a non-redundant pattern set, where non-redundancy is performed as post-processing, is quite difficult. To avoid pruning interesting search space areas, it would require to know the quality of the last top-k non-redundant patterns w.r.t. the quality measure at any time. It would have a high computational cost.

`Misere`, by Egho *et al.* [15], [16] has been proposed to sample interesting rules (originally in the case of sequences of itemsets) in a simple way, giving good results when those rules are used to build a classifier. Genetic algorithm can be considered to tackle this problem, but they do not offer strong guarantees like exhaustiveness or exploring only COTPs. For example, Lucas *et al.* proposed `SSDP` to mine top-k discriminative rules in high-dimensional data [17]. However, they only consider categorical and discrete attributes.

From a time series perspective, Mörchen and Ultsch [18] proposed a new language on interval time series. The problem is different: events have a duration instead of being instantaneous (e.g., injection of a drug in medical conditions), and they look for frequent rather than discriminative patterns. They used a frequent pattern mining approach, which is a different problem: we want to be able to find discriminating patterns. Similarly, Batal *et al.* [19] worked on predictive frequent time-interval patterns.

Atzmueller *et al.* [20] propose a method for the classification of time series. They discretize data with SAX [21], incurring information loss. Features are extracted with the FRESH algorithm from raw time series, and anomaly detection is done to improve the overall workflow. It requires human-in-the-loop. Nanlin Jin *et al.* used SD in Smart Electricity Meter Data [13]. Using the state-of-the-art algorithm in Subgroup Discovery (`Beam-search`), they explored different quality measures on data similar to time series but also containing categorical data. They showed the relevance of a SD approach to create interpretable rules used further in classification in the context of energy consumption. Nguyen *et al.* [22] propose a method that learns a linear classifier on discretized data (SAX or SFA). A feature (pattern) with the largest weight is considered the most discriminating. The goal is first to classify, then to extract patterns from the model. They use a smart pruning strategy to minimize the loss function, dynamically selecting features. However, such an approach ignores the quality measure (e.g., it is impossible to specify to $Precision$ or $Lift$ maximization).

VI. Quantitative Experimental Study

We conduct experiments to assess the performance of `MonteCloPi`. To guarantee reproducibility, all the materials are available online[1].

A. Experimental Setup

a) Datasets: We considered two high dimensional numerical datasets, namely LSVT [23] and gastro [24], coupled with 13 other benchmark datasets of the time series domain [25]. We also used a *RocketLeague* dataset, containing annotated video game traces [26]. The properties of the resulting datasets are summarized in Table II.

[1] www.dropbox.com/s/xhy14vvkr3konqr/monteclopi.zip?dl=0

TABLE II: Dataset statistics.

Dataset	$\lvert\mathcal{D}\rvert$	n	m	$\lvert\mathcal{C}\rvert$	Multi-variate	Space Size
Computers	250	720	1	2	False	$\approx 10^6$
Earthquakes	322	512	1	2	False	$\approx 10^8$
ECG5000	500	140	1	5	False	$\approx 10^9$
Gunpoint	50	150	1	2	False	$\approx 10^7$
Lightning2	60	637	1	2	False	$\approx 10^8$
Worms	182	900	1	5	False	$\approx 10^9$
Yoga	300	426	1	2	False	$\approx 10^9$
Handwriting	150	152	3	26	True	$\approx 10^{23}$
JapaneseVowels	270	29	12	9	True	$\approx 10^{83}$
NATOPS	180	51	24	6	True	$\approx 10^{182}$
RacketSports	151	30	6	4	True	$\approx 10^{39}$
RocketLeague	298	64	8	7	True	$\approx 10^{55}$
StandWalkJump	12	2,500	4	3	True	$\approx 10^{20}$
UWaveGest	2,238	315	3	8	True	$\approx 10^{25}$
LSVT	698	126	-	2	Num	$\approx 10^8$
gastro	309	76	-	3	Num	$\approx 10^8$

b) Baselines: We compare `MonteCloPi` with several algorithms:

- `Beam-Search` (see,e.g., [27]): Starting from the most general subgroup, i.e., containing intervals in the form $[min(Dom(attr)), max(Dom(attr))], \forall attr \in \mathcal{A}$, it explores the search space level-wise, taking the best elements (beam width) at each level. Each restriction (going down by one level) consists in taking an interval and splitting it in 6 equal-sized bins like in [4]. It prevents the search space from exploding, but forfeits exhaustiveness. We set the beam width to 50. Other settings do not impact significantly the reported results.
- `Misere` [15], [16]: It draws uniformly an object from the data, and then generates random generalizations to find the best possible prediction rules for this object. It iterates until the time budget is exhausted.
- `MCTS4DM` [6]: It is a MCTS-based top-down approach that processes numerical data without discretization. Its strategy guides the search following an exploration-exploitation trade-off to compute non-redundant patterns. It explores the lattice of intents, contrary to `MonteCloPi` that explores the lattice of extents.

We tried to use `Refine&Mine` designed for numerical subgroup discovery [7] but it provided only extremely rough discretizations on every dataset: it does not scale with high dimensional numerical data.

c) Evaluation Process: The assessment procedure is as follows. Given a time budget, each algorithm extracts the best patterns. Next, the following post-processing routine is applied for removing redundant patterns: we sort patterns by quality, keep the best, while removing similar patterns (in terms of Jaccard index) of poorer quality. Then we take the second best, etc, until we obtain k patterns. If not specified otherwise, the default parameters for the algorithms are the following: 60s of time budget, $\theta = 0.8$, $top\_k = 5$. Experiments were conducted on a laptop (Intel Core i7-8750H CPU and 16GB RAM).

d) Quality Measure: We explore performances of algorithms on different datasets, and study the impact of varying available time budget. In the following, if not stated otherwise, we use $NWRAcc$ as the quality measure. We average it over top-5 non-redundant patterns of 5 runs of the algorithms. Note that our approach is agnostic of the quality measure, and that any other discriminative quality measure could be considered.

B. Experimental results

1) Overall Performance: First, we look at the overall performance of `MonteCloPi` in terms of $NWRAcc$ compared to `Beam-Search`, `Misere` and `MCTS4DM`. In Fig. 2, we plot the results for all datasets, and one-vs-one comparisons of algorithms are proposed in Fig. 3, 4, 5. As we can see, `MonteCloPi` is the best approach on nearly all of them. `Beam-Search` may also be a good solution and its results are quite stable over datasets. However, our approach clearly outperforms it in the majority of cases. As for `Misere`, its results vary significantly depending on the dataset. For instance, it performs very well on **GunPoint**, while returning patterns having a zero quality on **UWaveGestureLibrary**. Finally, `MCTS4DM` that performs a MCTS-based approach but in a "top-down" way on intents clearly gives less interesting results on such datasets.

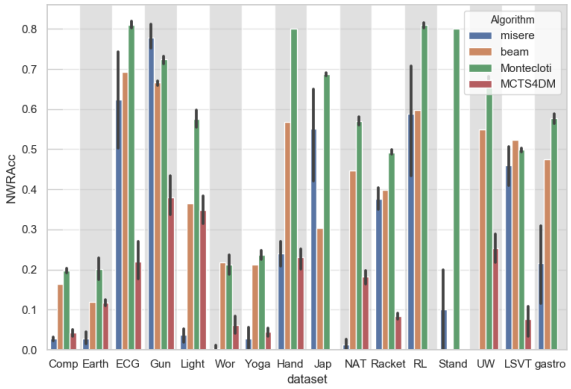

Fig. 2: Mean $NWRAcc$ of top-5 patterns with different algorithms. Bar errors represent the 95% confidence intervals

2) Varying Time Budget.: We illustrate the performance of `MonteCloPi`, `Beam-Search`, and `Misere` versus time budget in Fig. 6. As we can see, `MonteCloPi` quickly gives better results, improving over time. Moreover, those results are quite stable over runs of the same duration, contrary to the `Misere` case. Our hypothesis is that this is due to the exploration-exploitation trade-off, where we focus on interesting areas of the search space, whereas `Misere` performs a pure exploration. Another problem faced by `Misere` is the number of patterns one can create from an high-dimensional numerical object is very large. It results in a strong dependency of the quality of a pattern on the object that was randomly drawn. It explains the large variability of the results for `Misere`.

VII. A Qualitative Assessment in Game Analytics

We propose a use case based on the video game *Rocket League* (https://www.rocketleague.com). It is a competitive online e-sport game where each player controls a car on a football field having a ceiling and walls. The aim is to score goals with a ball more than the opponent team. Additionally to the speed and direction control of the car, players can make the car jump, flip (double jump), rotate in 3-dimensional space, slide, and use boost to accelerate. Those simple mechanics can be combined to create particular skills (like "nutmeg" skills in classical football). *Rocket League* players perform particular "skillshots" to take advantage of their opponent. Consider the "Musty Flick" as an example (see Fig. 7). It consists of a reverse flick well executed to lob the defender. Such action can easily be detected by an expert, such as an e-sport commentator. A question that arises is then: how can we detect it automatically in real time? It is a challenging problem. Indeed, as controls of a player over a car is very

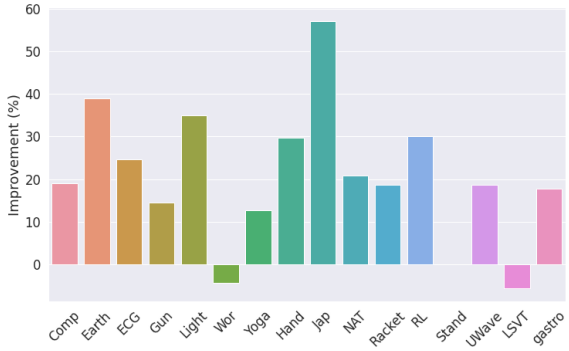

Fig. 3: Monteclopi vs beam-search

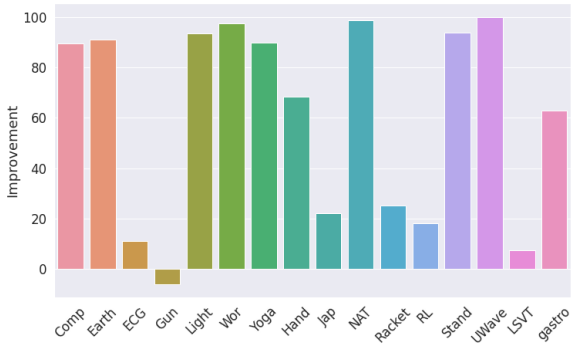

Fig. 4: Monteclopi vs misere

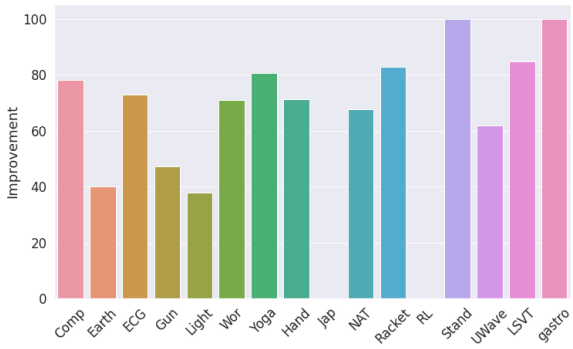

Fig. 5: Monteclopi vs MCTS4DM

precise, each performance of a "Musty Flick" is unique, even for the same player. Most of the variables are instance related: positions of both ball and players, timing, and even inputs of the player, as there are a lot of micro-adjustments made that can be considered as noise. Players are ranked using a system similar to Elo score in chess [28]. As skills have different levels of difficulty, using a detection system could allow assigning different points based on the difficulty level of performed skills. Moreover, it can be interesting *per se* to better understand players behavior, or to create new game modes where the goal would be to perform the best skills, as in figure skating or in gymnastics where, incidentally, real-time automated scoring of skills is being tested. However, instead of needing humans to judge, we may use a system for it, making those competitions available to millions of users.

a) Data: In [26], authors introduced a dataset of sequences of states, each state being composed of contextual numerical values and itemset of events. Those sequences can be encoded into multivariate time series (and then high dimensional numerical data), by setting value of event i to 1 when event i fires, 0 otherwise.

b) Classification Results: To the best of our knowledge, the classification problem we address cannot be easily solved with the state-of-the-art algorithms: the lengths of the multivariate time series being unequal, classification being subject to real-time constraint, and interpretable features being desired. Using `1NN DTW` is unfeasible in practice if we want to classify in real time: the classification on the test set, composed of 59 objects, took 7 minutes, achieving an accuracy of 71.5%. The confusion matrix obtained during cross-validation is given in Table III. Note that the noise is difficult to classify. Indeed, it is composed not only of random movements, but also of failed skills, making this classification task quite difficult to solve.

TABLE III: Confusion matrix

	Noise	Ceiling shot	Power shot	Waving Dash	Air Dribble	Flick	Musty Flick
Noise	0	0	0.5	0	0	0.25	0.25
Ceiling shot	0	1	0	0	0	0	0
Power shot	0	0	1	0	0	0	0
Waving Dash	0	0	0	1	0	0	0
Air Dribble	0	0	0	0	1	0	0
Flick	0	0	0	0	0	0.77	0.23
Musty Flick	0	0	0	0	0	0.43	0.57

We built an instance of the `LeGo` framework [29]: a set of interesting patterns is mined by `MonteCloPi` and then used as features to build a global model (a random forest classifier of 100 trees). With this strategy, we reached a classification accuracy of 80.1%, with 5-fold stratified cross-

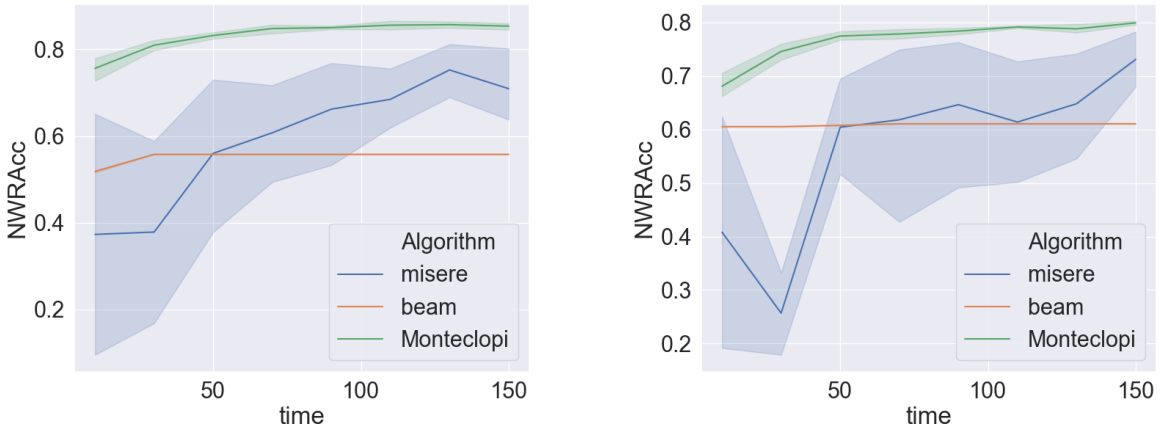

Fig. 6: $NWRAcc$ vs $time\ budget$ on **RocketLeague** (left) and **ECG5000** (right) averaged over 5 runs. The shaded areas represent the 95% confidence interval.

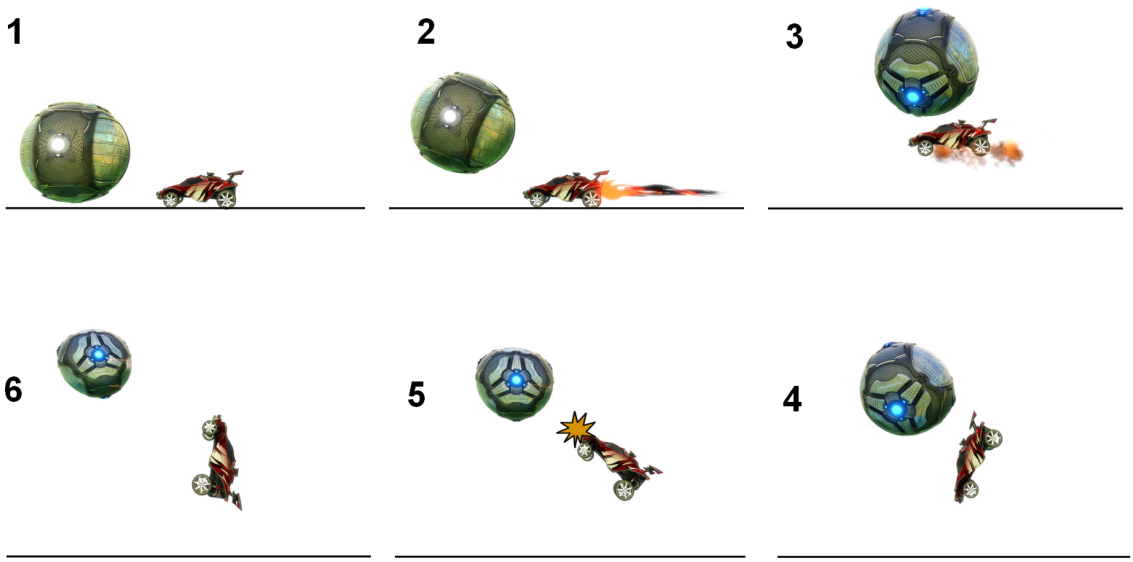

Fig. 7: Decomposition of the "Musty Flick"

validation, a training time of 1 hour, and *nearly immediate* (a few milliseconds) classification during test time.

c) Pattern Interpretability: Let us discuss two computed patterns: the top-1 pattern $\langle\{accelerate\},\{down\ jump\},\{goal\}\rangle$ $(WRAcc = 0.0863)$ corresponds to a "Musty Flick", and $\langle\{accelerate, jump, DistanceCeil = [1.52, 1233.51]\}\rangle$ denotes a "Ceiling Shot" $(WRAcc = 0.0755)$. For the "Musty Flick", discriminating parts are purely based on player's inputs. Note that this sequence of actions cannot be directly used to reproduce a "Musty Flick", but it is indeed discriminating this skill: there is a sequence of inputs to perform before this sequence, but it is common to every "Flick", so it is ignored by the algorithm. For the second pattern, the distance to the ceiling is important. Indeed, the "Ceiling Shot" consists in jumping from the ceiling and then hitting the ball in the air. This experiment emphasizes that `MonteCloPi` gives interesting and *interpretable* patterns.

VIII. Conclusion

We presented `MonteCloPi`, an efficient solution for subgroup discovery in numerical data. It is generic and can be easily adapted to other types of data. To do so, only two modifications are required: the redefinition of the MEET operator and the adaptation of the ROLLOUT step. For instance, we can study the application of `MonteCloPi` to graphs.

We showed that `MonteCloPi` can also deal with multivariate time series of different lengths. Though we reported results on a video game skill classification use case, many real-life mining processes may benefit from our findings.

The property of exhaustiveness on COTP gives the guarantee of finding the best elements and automatically stopping the algorithm, if given enough time. However, this property does not remain valid for the case of time series of different lengths. To overcome this limitation, one may compute all possible maximal common patterns between two elements in the EXPAND step. More investigations are needed, as the number of these patterns could be unreasonably large for some pattern languages.

References

[1] P. K. Novak, N. Lavrac, and G. I. Webb, "Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining," *J. Mach. Learn. Res.*, vol. 10, pp. 377–403, 2009.

[2] S. Wrobel, "An algorithm for multi-relational discovery of subgroups," in *Proc. PKDD*, 1997, pp. 78–87.

[3] M. Atzmüller and F. Puppe, "Sd-map - A fast algorithm for exhaustive subgroup discovery," in *Proc. PKDD*, 2006, pp. 6–17.

[4] M. van Leeuwen and A. J. Knobbe, "Diverse subgroup set discovery," *Data Min. Knowl. Discov.*, vol. 25, no. 2, pp. 208–242, 2012.

[5] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Trans. Comput. Intellig. and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.

[6] G. Bosc, J.-F. Boulicaut, C. Raïssi, and M. Kaytoue, "Anytime discovery of a diverse set of patterns with monte carlo tree search," *Data Min. Knowl. Discov.*, vol. 32, no. 3, pp. 604–650, 2018.

[7] A. Belfodil, A. Belfodil, and M. Kaytoue, "Anytime subgroup discovery in numerical domains with guarantees," in *Proc. ECML/PKDD II*, 2018, pp. 500–516.

[8] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances," *Data Min. Knowl. Discov.*, vol. 31, pp. 606–660, 2017.

[9] M. Kaytoue, S. O. Kuznetsov, and A. Napoli, "Revisiting numerical pattern mining with formal concept analysis," in *Proc. IJCAI*, 2011, p. 1342–1347.

[10] N. Lavrac, P. A. Flach, and B. Zupan, "Rule evaluation measures: A unifying view," in *Proc. ILP*, 1999, pp. 174–185.

[11] A. Belfodil, A. Belfodil, A. Bendimerad, P. Lamarre, C. Robardet, M. Kaytoue, and M. Plantevit, "FSSD - A fast and efficient algorithm for subgroup set discovery," in *Proc. IEEE DSAA*, 2019, pp. 91–99.

[12] G. C. Garriga, P. Kralj, and N. Lavrac, "Closed sets for labeled data," *J. Mach. Learn. Res.*, vol. 9, pp. 559–580, 2008.

[13] N. Jin, P. Flach, T. Wilcox, R. Sellman, J. Thumim, and A. Knobbe, "Subgroup discovery in smart electricity meter data," *IEEE Trans. on Industrial Informatics*, vol. 10, no. 2, pp. 1327–1336, 2014.

[14] A. Millot, R. Cazabet, and J.-F. Boulicaut, "Optimal subgroup discovery in purely numerical data," in *Proc. PaKDD 2020 II*, 2020, pp. 112–124.

[15] E. Egho, D. Gay, M. Boullé, N. Voisine, and F. Clérot, "A parameter-free approach for mining robust sequential classification rules," in *Proc. IEEE ICDM*, 2015, pp. 745–750.

[16] E. Egho, D. Gay, M. Boullé, N. Voisine, and F. Clérot, "A user parameter-free approach for mining robust sequential classification rules," *Knowl. Inf. Syst.*, vol. 52, no. 1, pp. 53–81, 2017.

[17] T. Lucas, T. C. Silva, R. Vimieiro, and T. B. Ludermir, "A new evolutionary algorithm for mining top-k discriminative patterns in high dimensional data," *Applied Soft Computing*, vol. 59, pp. 487 – 499, 2017.

[18] F. Mörchen and A. Ultsch, "Efficient mining of understandable patterns from multivariate interval time series," *Data Min. Knowl. Discov.*, vol. 15, no. 2, pp. 181–215, 2007.

[19] I. Batal, D. Fradkin, J. H. Harrison, F. Mörchen, and M. Hauskrecht, "Mining recent temporal patterns for event detection in multivariate time series data," *Proc. ACM SIGKDD*, pp. 280–288, 2012.

[20] E. Ramirez, M. Wimmer, and M. Atzmueller, "A computational framework for interpretable anomaly detection and classification of multivariate time series with application to human gait data analysis," in *AIME International Workshops KR4HC/ProHealth and TEAAM Revised Selected Papers*, 2019, pp. 132–147.

[21] P. Patel, E. Keogh, J. Lin, and S. Lonardi, "Mining motifs in massive time series databases," in *Proceedings of the 2002 IEEE International Conference on Data Mining*, ser. ICDM '02. USA: IEEE Computer Society, 2002, p. 370.

[22] T. Nguyen, S. Gsponer, I. Ilie, M. O'reilly, and G. Ifrim, "Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations," *Data Min. Knowl. Discov.*, vol. 33, no. 4, p. 1183–1222, 2019.

[23] A. Tsanas, M. Little, C. Fox, and L. Ramig, "Objective automatic assessment of rehabilitative speech treatment in parkinson's disease," *IEEE Trans. on Neural Systems and Rehabilitation Engineering*, vol. 22, no. 1, pp. 181–190, Jan. 2014.

[24] P. Mesejo, D. Pizarro, A. Abergel, O. Rouquette, S. Beorchia, L. Poincloux, and A. Bartoli, "Computer-aided classification of gastrointestinal lesions in regular colonoscopy," *IEEE Transactions on Medical Imaging*, vol. 35, no. 9, pp. 2051–2063, 2016.

[25] A. Bagnall, J. Lines, W. Vickers, and E. Keogh, "The UEA & UCR time series classification repository," 2017, www.timeseriesclassification.com.

[26] R. Mathonat, J.-F. Boulicaut, and M. Kaytoue, "A behavioral pattern mining approach to model player skills in rocket league," in *2020 IEEE Conference on Games (CoG)*, August 2020, in Press. 8 pages.

[27] W. Duivesteijn, A. J. Feelders, and A. Knobbe, "Exceptional model mining," *Data Min. Knowl. Discov.*, vol. 30, no. 1, pp. 47–98, 2016.

[28] R. Herbrich, T. Minka, and T. Graepel, "Trueskill(tm): A bayesian skill rating system," in *Advances in Neural Information Processing Systems*, 2007, pp. 569–576.

[29] A. J. Knobbe, B. Crémilleux, J. Fürnkranz, and M. Scholz, "From local patterns to global models: The {LeGo} approach to data mining," in *From Local Patterns to Global Models: Proc. of the ECML/PKDD-08 Workshop (LeGo-08)*, 2008, pp. 1–16.