Learning Pareto Fronts From Membership Queries

Alexey Bakhirkin, Nicolas Basset, Oded Maler*, José-Ignacio Requeno Jarabo
{alexey.bakhirkin, bassetni, requenoj}@univ-grenoble-alpes.fr

VERIMAG, CNRS and Université Grenoble-Alpes, FRANCE

Abstract. We present a new method for inferring the Pareto front in multi-criteria optimization problems. The approach is grounded on an algorithm for learning the boundary between valid ($\overline{X}$) and invalid ($\underline{X}$) configurations of a multi-dimensional space (X). The algorithm selects sampling points for which it submits membership queries $x \in \overline{X}$ to an oracle. Based on the answers and relying on monotonicity, it constructs an approximation of the boundary. The algorithm generalizes binary search on the continuum from one-dimensional (and linearly-ordered) domains to multi-dimensional (and partially-ordered) ones. The procedure explained in this paper has been applied for the parameter synthesis of extended Signal Temporal Logic (STLe) expressions where the influence of parameters is monotone. Our method has been implemented in a free and publicly available Python library.

1 Introduction

Let X be a bounded and partially ordered set that we consider from now on to be $[0,1]^n$. A subset $\overline{X}$ of X is *upward closed* in X if

$$\forall x, x' \in X \ (x \in \overline{X} \wedge x' \geq x) \rightarrow x' \in \overline{X}.$$

Naturally, the complement of $\overline{X}$, $\underline{X} = X \setminus \overline{X}$ is downward closed, and we use the term *monotone bi-partition* (or simply partition) for the pair $M = (\underline{X}, \overline{X})$. We do not have an explicit representation of M and we want to approximate it based on queries to a membership oracle which can answer for every $x \in X$ whether $x \in \overline{X}$. Based on this information we construct an approximation of M by a pair of sets, $(\underline{Y}, \overline{Y})$ being, respectively, a downward-closed subset of $\underline{X}$ and an upward-closed subset of $\overline{X}$, see Figure 1. This approximation, conservative in both directions, says nothing about points residing in the gap between $\underline{Y}$ and $\overline{Y}$. This gap can be viewed as an over-approximation of $bd(M)$, the boundary between the two sets.

Before presenting the algorithmic solution that we offer to the problem, let us discuss some motivations. To start with, the problem is interesting for its

* Oded Maler passed away at the beginning of September 2018. This work was initiated by him [14] continued with and finished by the rest of us.

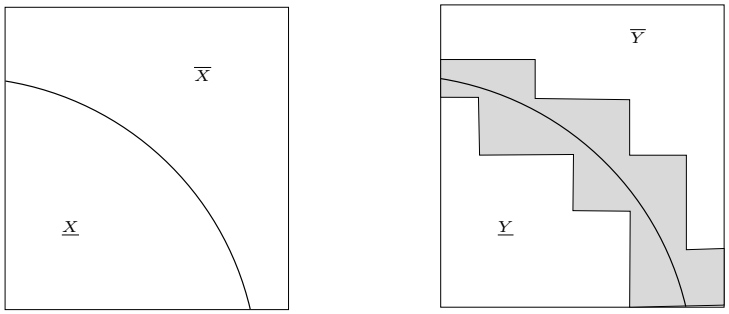


Fig. 1: A monotone partition and its approximation.

own sake as a neat multi-dimensional generalization of the problem of locating a boundary point that splits an interval into two sub-intervals. This problem is solved typically using binary (dichotomic) search, and indeed, the essence of our approach is in embedding binary search in higher dimension.

One major motivation comes from the domain of *multi-criteria optimization* problems where solutions are evaluated according to several criteria. The cost of a solution can be viewed as a point in a multi-dimensional cost space X. The optimal cost of such optimization problems is rarely a single point but rather a set of incomparable points, also known as the *Pareto front* of the problem. It consists of solutions that cannot be improved in one dimension without being worsened in another. For a minimization problem, $\underline{X}$ corresponds to infeasible costs and $\overline{X}$ represents the feasible costs. Here we recall that the set of feasible cost is accessible only via a membership oracle and that we only assume upward-closure of the set of feasible cost and not its convexity.

Another motivation comes from some classes of parametric identification problems. Consider a parameterized family of predicates or constraints $\{\varphi_p\}$ where p is a vector of parameters ranging over some parameter space. Given an element u from the domain of the predicates, we would like to know the range of parameters p such that $\varphi_p(u)$ holds. We say that a parameter p has a fixed (positive or negative) polarity if increasing its value will have a monotone effect on the set of elements that satisfy it. For example if a parameter p appears in a parameterized predicate $u \leq p$, then for any $p' > p$ and any u, $\varphi_p(u)$ implies $\varphi_{p'}(u)$. The set of minimal elements u indicates the set of tightest parameters that lead to satisfaction of $\varphi_p(u)$, which is a valuable information about u. In [1] we already explored the idea for the domain of real-valued signals $u(t)$ and temporal formulas such as $\exists t < p_1\ u(t) < p_2$.

We now sum up the contributions of this article.

– We propose a new method for learning the monotone bi-partition of partially ordered domain based on membership-queries. This can be equivalently formulated as learning the Pareto front associated to a multi-criteria optimisation problem where the set of feasible cost is only accessible from a membership oracle.

– Our method is agnostic of the optimization problem considered; in particular it makes no assumption on the convexity of the Pareto front.
– Our algorithm progressively discovers Pareto optimal points making the boundary approximation (represented as a disjoint union of boxes) smaller and smaller. In a nutshell we make use of classical binary search along the diagonal of the boxes that partitionate the unknown regions.
– We give a lower bound on the volume of the unknown part removed from the approximation after each discovery of a new Pareto optimal point, guaranteeing the convergence of our method.

We proposed in a work-in-progress report [14] a preliminary version of the algorithm without the materials of Section 4–6. An implementation was made in [16] and it was successfully used for learning valuations of parametric Signal Temporal Logic (STL) [15] specifications with standard operators over time-series data [18,17]. In this paper, we complete [14] with new contributions for a) spaces with dimension higher than 2, and b) we prove the convergence of our method. A new software tool [5] based on the current improvements of this paper outperforms the capabilities of the previous implementation in [16]. Additionally, we apply our method for the parameter identification of specification written in the extended version of STL proposed in [3].

1.1 Related Work

Multi-criteria optimization techniques are commonly categorized in two families [8,10]. The first one is associated to mathematical programming methods, which scale a multi-dimensional problem into a one-dimensional utility function by taking a weighted sum of the various costs. An optimal solution for the one-dimensional problem is also a Pareto solution for the original problem. Every invocation to the resolution method discovers a new Pareto optimal point, but the result depends on the choice of the weighted coefficients. One of the most popular algorithm in this family is the Chord Algorithm (see e.g. [7] and reference therein). Methods based on weigthed sums work under the assumption that the Pareto front is convex. Here we make no assumption on the form of the Pareto front and we consider the very general framework where the set to infer is within a black box accessed only by membership queries. In particular, there is no way of solving directly a one-dimensional problem created from weighted sums of costs.

The second family of methods is based on evolutionary algorithms. However, these methods cannot guarantee the Pareto optimality of the set of solutions generated: it is only known that none of the generated solutions dominates the others. Consequently, a major issue in these heuristic techniques consists of finding meaningful measures of quality for the sets of solutions they provide [19].

Our dichotomic searches are made on intervals that cross the Pareto front. It is thus different from searches along interval between points already in the Pareto front (see e.g. [12]). In [9], membership queries are used over interval of the space aligned to the axis to find the Pareto front for a discrete version of the

problem. Here we consider continuous cost space and we treat all the dimensions in a symetric way and simultaneously by searching along the diagonal. In [11], a divide and conquer approach is used for a 2-dimensional problems. Similarly to our approach, points are successively discovered in the Pareto front and the unknown part is then treated recursively. However, the discovery of points was done by minimising a weigted sum via linear programming, something which is not possible when only a membership oracle is provided.

In [13] we developed a procedure for computing such an approximation $(\underline{Y}, \overline{Y})$ of a monotone bi-partion $(\underline{X}, \overline{X})$ with a guarantee on the gap between $\underline{Y}$ and $\overline{Y}$. At the end of the algorithm each point of the border of one part is at distance at most epsilon to the other part. The present paper presents an algorithm that follows the same phylosophy in the sense that it also provides a way to approximate Pareto fronts with quality metrics. Our algorithm has the additive feature of finding a cloud of exact Pareto optimal points.

1.2 Paper Structure

Sections 2–3 introduces the basic notations and our approach for learning the Pareto front in multi-dimensional spaces. In Section 4 we spot phenomena that appears for dimension greater than 2, we enhance our method accordingly to mitigate the curse of dimensionality. Section 5 analyzes the complexity of our method. Finally, Section 6 continues in the same direction than [1] and applies our method to the parameter synthesis of Signal Temporal Logic (STL) [15] specifications with extended operators [3].

Several omitted results and proofs are given in a technical report [4] as well as in the appendix of the present paper.

2 Preliminaries

2.1 Binary search in one dimension

Our major tool is the classical binary search over one-dimensional and totally-ordered domains, where a partition of $[0, 1]$ is of the form $M = ([0, z), [z, 1])$ for some $0 < z < 1$. The outcome of the search procedure is a pair of numbers $\underline{y}$ and $\overline{y}$ such that $\underline{y} < z < \overline{y}$, which implies a partition approximation $M' = ([0, \underline{y}), [\overline{y}, 1])$. The quality of M' is measured by the size of the gap $\overline{y} - \underline{y}$, which can be made as small as needed by running more steps. Note that in one dimension, $\overline{y} - \underline{y}$ is both the volume of $[\underline{y}, \overline{y}]$ and its diameter. We are going to apply binary search to straight lines of arbitrary position and arbitrary positive orientation inside multi-dimensional X, hence we formulate it in terms that will facilitate its application in this context.

Definition 1 (Line Segments in High-Dimension). *The line segment connecting two points $\underline{x} < \overline{x} \in X = [0, 1]^n$ is their convex hull*

$$\langle \underline{x}, \overline{x} \rangle = \{(1 - \lambda)\underline{x} + \lambda\overline{x} : \lambda \in [0, 1]\}.$$

The segment inherits a total order from $[0, 1]$: $x \leq x'$ whenever $\lambda \leq \lambda'$.

The input to the binary search procedure, written in Algorithm 1, is a line segment ℓ and an oracle for a monotone partition $M = (\underline{\ell}, \overline{\ell}) = (\langle \underline{x}, z \rangle, \langle z, \overline{x} \rangle)$, $\underline{x} < z < \overline{x}$. The output is a sub-segment $\langle \underline{y}, \overline{y} \rangle$ containing the boundary point z. The procedure is parameterized by an error bound $\epsilon \geq 0$, with $\epsilon = 0$ representing an ideal variant of the algorithm that runs indefinitely and finds the exact boundary point. Although realizable only in the limit, it is sometimes convenient to speak in terms of this variant. Fig. 2 illustrates several steps of the algorithm.

Algorithm 1 One dimensional binary search: $search(\langle \underline{x}, \overline{x} \rangle, \epsilon)$

Input: A line segment $\ell = \langle \underline{x}, \overline{x} \rangle$, a monotone partition $M = (\underline{\ell}, \overline{\ell})$ accessible via an oracle $member()$ for membership in $\overline{\ell}$ and an error bound $\epsilon \geq 0$.
Output: A line segment $\langle \overline{y}, \underline{y} \rangle$ containing $bd(M)$ such that $\overline{y} - \underline{y} \leq \epsilon$.
$\langle \underline{y}, \overline{y} \rangle = \langle \underline{x}, \overline{x} \rangle$
while $\overline{y} - \underline{y} \geq \epsilon$ **do**
    $y = (\underline{y} + \overline{y})/2$
    if $member(y)$ **then**
        $\langle \underline{y}, \overline{y} \rangle = \langle \underline{y}, y \rangle$ ▷ left sub-interval
    else
        $\langle \underline{y}, \overline{y} \rangle = \langle y, \overline{y} \rangle$ ▷ right sub-interval
    end if
end while
return $\langle \underline{y}, \overline{y} \rangle$

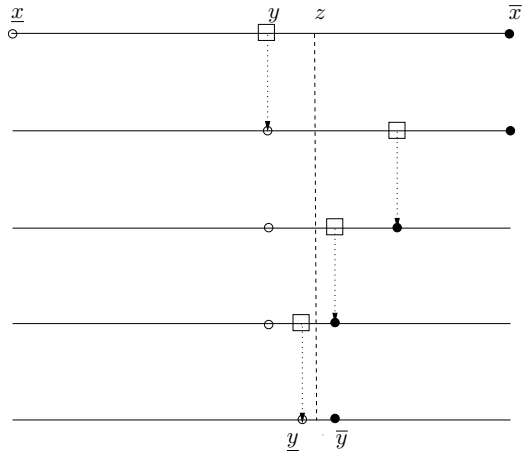


Fig. 2: Binary search and the successive reduction of the uncertainty interval.

3 Learning the Multi-dimensional Pareto Front

The following definitions are commonly used in multi-criteria optimization and in partially-ordered sets in general.

Definition 2 (Domination and Incomparability). *Let $x = (x_1, \ldots, x_n)$ and $x' = (x'_1, \ldots, x'_n)$ be two points. Then*

1. *$x \leq x'$ if $x_i \leq x'_i$ for every i;*
2. *$x < x'$ if $x \leq x'$ and $x_i < x'_i$ for some i. In this case we say that x dominates x';*
3. *$x||x'$ if $x \not\leq x'$ and $x' \not\leq x$, which means that $x_i < x'_i$ and $x'_j < x_j$ for some i and j. In this case we say that x and x' are incomparable.*

Any two points $\underline{x} < \overline{x}$ define a *box* $\lfloor \underline{x}, \overline{x} \rceil = \{x : \underline{x} \leq x \leq \overline{x}\}$ for which they are, respectively, the minimal and maximal corners, as well as the endpoints of the diagonal $\langle \underline{x}, \overline{x} \rangle$. Equivalently the box $\lfloor \underline{x}, \overline{x} \rceil$ is defined as the product of the intervals $\prod_{i=1}^{n} [\underline{x}_i, \overline{x}_i]$. Its volume is $\texttt{Vol}(\lfloor \underline{x}, \overline{x} \rceil) = \prod_{i=1}^{n} (\overline{x}_i - \underline{x}_i)$. Two boxes can intersect (we say also overlap), the intersection being itself a box. We do not take care of the overlap of volume 0 and it will be often the case (even in the ideal version of our algorithms) that the points in the frontier of a box can be found in the frontier of several other boxes we consider.

The procedure for learning a monotone partition is written down in Algorithm 2 and works as follows. It maintains at any moment the current approximation $(\underline{Y}, \overline{Y})$ of the partition as well as a list L of boxes whose union constitutes an over-approximation of the boundary, that is $bd(M) \subseteq \cup_{\lfloor \underline{x}, \overline{x} \rceil \in L} \lfloor \underline{x}, \overline{x} \rceil$. For efficiency reasons, L is maintained in a decreasing order of volume. We successively take the largest box $\lfloor \underline{x}, \overline{x} \rceil$ from L, and find in it parts that we can move to $\underline{Y}$ and $\overline{Y}$. As the algorithm proceeds $\underline{Y}$ and $\overline{Y}$ augments and get closer to $\underline{X}$ and $\overline{X}$; and the total volume of the boundary approximation decreases until some stopping criterion is met (e.g. when going below a threshold δ).

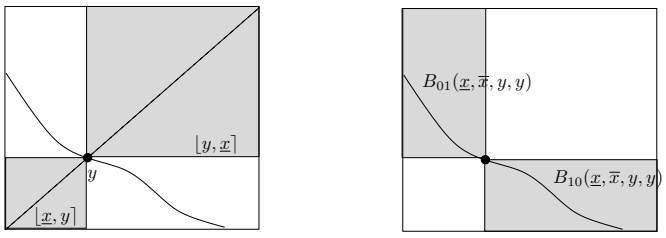
To treat the box $\lfloor \underline{x}, \overline{x} \rceil$ we rely on the observation that any line ℓ of a positive slope inside a box $\lfloor \underline{x}, \overline{x} \rceil$ that admits a monotone partition M, intersects $bd(M)$ at most once. In particular, the diagonal $\ell = \langle \underline{x}, \overline{x} \rangle$ of the box is guaranteed to intersect $bd(M)$ and such intersection is over-approximated by the segment $\langle \underline{y}, \overline{y} \rangle$ returned by the one-dimensional binary search algorithm applied on the diagonal $\langle \underline{x}, \overline{x} \rangle$. In this case $\lfloor \underline{x}, \underline{y} \rceil$ is added to $\underline{Y}$ since its points dominate $\underline{y}$ and the box $\lfloor \overline{y}, \overline{x} \rceil$ is added to $\overline{Y}$ since its points are dominated by $\overline{y}$. The remaining part of $\lfloor \underline{x}, \overline{x} \rceil$ stays in the border approximation and is split into a union of boxes $I(\overline{x}, \underline{x}, \overline{y}, \underline{y})$ described below (Def. 5). Fig. 3 illustrates the interaction between the result of the one-dimensional search process on the diagonal of a box $\lfloor \underline{x}, \overline{x} \rceil$ and the approximation of the higher-dimensional partition.

Definition 3 (Sub-intervals). *The sub-interval of the interval $[\underline{x}, \overline{x}]$ induced by $a \in \{0, 1\}$ and the interval $[\underline{y}, \overline{y}] \subseteq [\underline{x}, \overline{x}]$ is*

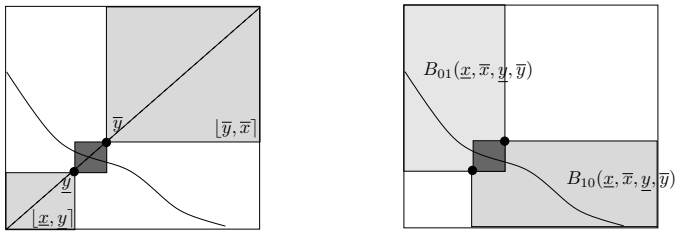
$$I_a(\underline{x}, \overline{x}, \underline{y}, \overline{y}) = \begin{cases} [\underline{x}, \overline{y}] & \text{if } a = 0 \\ [\underline{y}, \overline{x}] & \text{if } a = 1 \end{cases}$$

Algorithm 2 Approximating a monotone partition (and its boundary)

Input: A box $X = \lfloor \mathbf{0}, \mathbf{1} \rceil$, a partition $M = (\underline{X}, \overline{X})$ accessed by a membership oracle for $\overline{X}$ and an error bound δ.
Output: An approximation $M' = (\underline{Y}, \overline{Y})$ of M and an approximation L of the boundary $bd(M)$ such that $|L| \leq \delta$. All sets are represented by unions of boxes.
$L = \{X\}$; $(\underline{Y}, \overline{Y}) = (\emptyset, \emptyset)$ ▷ initialization
repeat
  pop first $\lfloor \underline{x}, \overline{x} \rceil \in L$ ▷ take the largest box from the boundary approximation
  $\langle \underline{y}, \overline{y} \rangle = search(\langle \underline{x}, \overline{x} \rangle, \epsilon)$ ▷ run binary search on the diagonal
  $\overline{Y} = \overline{Y} \cup \{\lfloor \overline{x}, \overline{y} \rceil\}$ ▷ add dominated sub-box
  $\underline{Y} = \underline{Y} \cup \{\lfloor \underline{x}, \underline{y} \rceil\}$ ▷ add dominating sub-box
  $L = L \cup I(\overline{x}, \underline{x}, \overline{y}, \underline{y})$ ▷ insert remainder of $\lfloor \underline{x}, \overline{x} \rceil$ to L
  $\texttt{Vol}(L) = \texttt{Vol}(X) - \texttt{Vol}(\underline{Y}) - \texttt{Vol}(\overline{Y})$
until $\texttt{Vol}(L) \leq \delta$



(a)



(b)

Fig. 3: (a) The effect of finding the exact intersection of the diagonal with the boundary; (b) The effect of finding an interval approximation of that intersection.

In the previous definition a is a boolean variable that evaluates to 1 (resp. 0) when the upper (resp. lower) part of the interval is selected. The previous definition extends to boxes (that is product of intervals) when a binary word $\alpha = \alpha_1 \cdots \alpha_n \in \{0,1\}^n$ is provided:

Definition 4 (Sub-boxes). *Let $\alpha \in \{0,1\}^n$ and 4 n-dimensional points $\underline{x} \leq \underline{y} \leq \overline{y} \leq \overline{x}$. The sub-boxes of $\lfloor \underline{x}, \overline{x} \rceil$ induced by α and $\lfloor \underline{y}, \overline{y} \rceil$ are*

$$B_\alpha(\underline{x}, \overline{x}, \underline{y}, \overline{y}) = \prod_{i=1}^{n} I_{\alpha_i}(\underline{x}_i, \overline{x}_i, \underline{y}_i, \overline{y}_i)$$

We can now define $I(\overline{x}, \underline{x}, \overline{y}, \underline{y})$.

Definition 5 (Incomparable sub-boxes). *The set $I(\overline{x}, \underline{x}, \overline{y}, \underline{y})$ of sub-boxes of $\lfloor \overline{x}, \underline{x} \rceil$ incomparable to $\lfloor \underline{y}, \overline{y} \rceil$ is*

$$I(\overline{x}, \underline{x}, \overline{y}, \underline{y}) = \{B_\alpha(\underline{x}, \overline{x}, \underline{y}, \overline{y}) : \alpha \in \{0,1\}^n \setminus \{0^n, 1^n\}\}$$

Observe that the union of boxes in $I(\overline{x}, \underline{x}, \overline{y}, \underline{y})$ is equal to $\lfloor \underline{x}, \overline{x} \rceil \setminus (\lfloor \underline{x}, \underline{y} \rceil \cup \lfloor \overline{y}, \overline{x} \rceil)$, that is the part that remains in the boundary approximation when we discover $\lfloor \underline{y}, \overline{y} \rceil$ around a point of the boundary in $\lfloor \underline{x}, \overline{x} \rceil$. Note that these boxes overlap but the volume of the overlap is proportional to ϵ (because it is made of boxes each one having a side with length smaller than ϵ). Hence this overlap can be made arbitrarily small by decreasing the parameter ϵ. This is efficient as the number of queries in a one-dimensional binary search is logarithmic in $1/\epsilon$.

Last but not least the stopping criterion is based on the volume $\texttt{Vol}(L)$ of the boundary approximation. Since L is a collection of boxes that can overlap, it is easier to compute its volume as the total volume of the box $X = [0,1]^n$ minus the volume of both parts of the monotone partition approximation $(\underline{Y}, \overline{Y})$. Some steps of the algorithm are illustrated in Fig. 4.

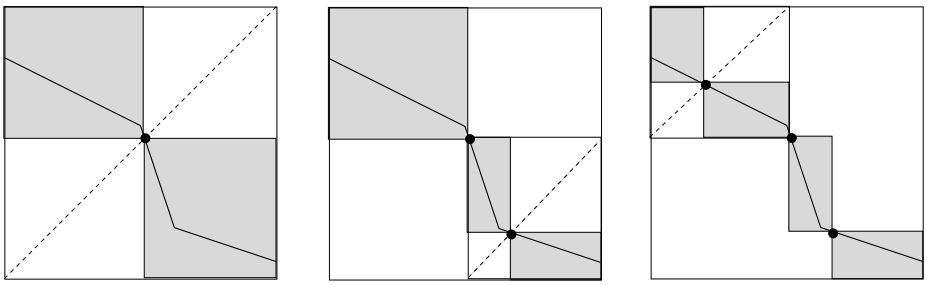


Fig. 4: Successive approximation of the partition boundary by running binary search on diagonals of incomparable boxes.

4 Two Enhancements in Dimensions Higher than 2

The goal of introducing changes to the learning procedure in dimension higher than 2 is to attack the curse of dimensionality. In order to mitigate this problem, we propose two enhancements. First, we minimize the number of incomparable sub-boxes that are generated when a new Pareto point is discovered (i.e., updated definition of $I'(\overline{x}, \underline{x}, \overline{y}, \underline{y})$ in Sect. 4.1). Second, we maximize the volume that is removed from the unexplored region by propagating the dominance of every Pareto point to the remaining boxes in the boundary approximation (Sect. 4.2). We sum up the new version of the algorithm in Sect. 4.3.

4.1 Enhanced decomposing step

In this section we come back to the treatment of the remainder of the box $\lfloor \underline{x}, \overline{x} \rceil$ when the two boxes above $\overline{y}$ and below $\underline{y}$ are removed. The remainder part was covered by the union of boxes of $I(\overline{x}, \underline{x}, \overline{y}, \underline{y})$. In two dimensions, this set only contains two boxes and every point of the box $B_{01}(\underline{x}, \overline{x}, \underline{y}, \overline{y})$ is incomparable to all the points of $B_{10}(\underline{x}, \overline{x}, \underline{y}, \overline{y})$ except for the small overlapping part.

In higher dimension some boxes can be grouped together into bigger boxes. In particular when two sub-boxes differ along only one dimension, they are *adjacent*: their union is again a box. We introduce an extra symbol $\star$ to encode that we do not care of this dimension. Defs 3–4 must be consequently extended for supporting the symbol $\star$ so that $I_\star(\underline{x}, \overline{x}, \underline{y}, \overline{y}) = [\underline{x}, \overline{x}]$ in one dimension or, in other words, $I_\star(\underline{x}, \overline{x}, \underline{y}, \overline{y}) = I_0(\underline{x}, \overline{x}, \underline{y}, \overline{y}) \cup I_1(\underline{x}, \overline{x}, \underline{y}, \overline{y})$. Similarly, $B_\beta(\underline{x}, \overline{x}, \underline{y}, \overline{y}) = \prod_{i=1}^{n} I_{\beta_i}(\underline{x}_i, \overline{x}_i, \underline{y}_i, \overline{y}_i)$ with $\beta \in \{0, 1, \star\}^n$ in higher dimension. Examples of boxes created by these expressions are given in Fig. 5.

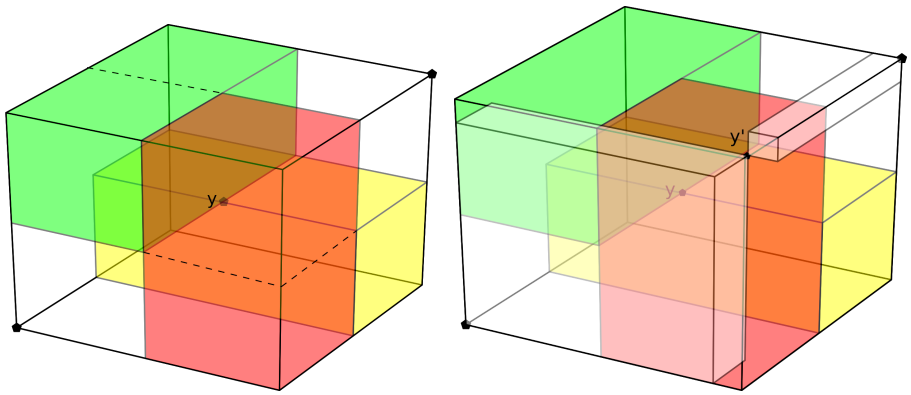
Now we define a better decomposition of $\lfloor \underline{x}, \overline{x} \rceil \setminus (\lfloor \underline{x}, \underline{y} \rceil \cup \lfloor \overline{y}, \overline{x} \rceil)$ using words in the alphabet $\{0, 1, \star\}$ for encoding boxes formed as unions of adjacent boxes of $I(\overline{x}, \underline{x}, \overline{y}, \underline{y})$. A partition of the index set $A = \{\alpha \mid \alpha \in \{0,1\}^n \setminus \{0^n, 1^n\}\}$ is a set $\mathbb{A}$ of non-empty subsets such that $A = \bigcup \mathbb{A}$ and for every two different parts $A_i, A_j \in \mathbb{A}$, $A_i \cap A_j = \emptyset$. A partition $\mathbb{A}$ of the index set A is called *feasible* if for each part $A_i \in \mathbb{A}$, the geometrical union of boxes of A_i is itself a box. Such a box can be described by a word $\beta \in \{0, 1, \star\}^n$, that is $B_\beta(\underline{x}, \overline{x}, \underline{y}, \overline{y}) = \cup_{\alpha \in A_i} B_\alpha(\underline{x}, \overline{x}, \underline{y}, \overline{y})$. It is convenient to overload the notation and write β for summarizing the part A_i. We note $\mathbb{A}^n$ for a partition of dimension n. For instance $\mathbb{A}^3 = \{10\star, \star10, 0 \star 1\}$ is a feasible partition of dimension 3 that encodes the boxes represented in Fig. 5. In the following proposition we use the notation $\star\mathbb{A}^n = \{\star\beta : \beta \in \mathbb{A}^n\}$.

Proposition 1. *Let $n \geq 3$. There exists a feasible partition $\mathbb{A}^n$ of $\{0,1\}^n \setminus \{0^n, 1^n\}$ with $2n-3$ parts defined recursively as follows:*

$$\mathbb{A}^3 = \{10\star, \star10, 0\star1\} \text{ and for } n \geq 3, \mathbb{A}^{n+1} = \star\mathbb{A}^n \cup \{10^n\} \cup \{01^n\}.$$

As an illustration we give $\mathbb{A}^n$ for the dimension 4 and 5:

$$\mathbb{A}^4 = \star\mathbb{A}^3 \cup \{1000\} \cup \{0111\} = \{\star10\star, \star\star10, \star0\star1, 1000, 0111\}.$$



(a) (b)

Fig. 5: A way of decomposing the remainder part into $2n-3=3$ boxes instead of $2^n-2=6$ in dimension $n=3$. The red, yellow and green box correspond to $B_{10\star}(\underline{x},\overline{x},\underline{y},\overline{y})$, $B_{\star 10}(\underline{x},\overline{x},\underline{y},\overline{y})$ and $B_{0\star 1}(\underline{x},\overline{x},\underline{y},\overline{y})$ respectively.

$$\mathbb{A}^5 = \{\star\star 10\star, \star\star\star 10, \star\star 0\star 1, \star 1000, \star 0111, 10000, 01111\}.$$

When updating the boundary approximation in dimension $n>2$, we add a set of $2n-3$ boxes instead of adding the set $I(\overline{x},\underline{x},\overline{y},\underline{y})$ containing 2^n-2 boxes:

$$I'(\overline{x},\underline{x},\overline{y},\underline{y}) = \bigcup_{\beta\in\mathbb{A}^n} \{B_\beta(\underline{x},\overline{x},\underline{y},\overline{y})\}. \tag{1}$$

Example 1. Consider again Fig. 5. We run the ideal version of our algorithm and after the first call to the binary search over the diagonal discover that the point $y=(0.5,0.5,0.5)$ is in the boundary. We remove the lower part $\lfloor \mathbf{0}, y\rceil$ and the upper part $\lfloor y, \mathbf{1}\rceil$, we then decompose the remaining part into the three boxes shown in Fig. 5a. The three boxes having the same volume, we arbitrarily treat the red box first in the next example (Ex. 2 below).

4.2 Propagating dominance to boxes of the boundary approximation

In dimension higher than 2, it can be the case that when an approximated point $\lfloor \underline{y},\overline{y}\rceil$ is found within a box $\lfloor \underline{x},\overline{x}\rceil$, the downward-closure of $\underline{y}$ intersects other boxes of the boundary approximation (see Ex. 2). These parts must also be removed from the boundary approximation to keep $\overline{Y}$ downward-closed. A symmetric remark holds for the upward-closure of $\overline{y}$.

Example 2 (Example 1 continued). When calling the binary search to the red box, we discover that the point $y'=(0.9,0.2,0.9)$ (Fig.5b) is in the boundary. We remove $[0.5,0.9]\times[0,0.2]\times[0,0.9]$ and $[0.9,1]\times[0.2,0.5]\times[0.9,1]$ from the red box and partition the remainder of the red box. Though, there is a part

outside the red box that also should be removed, that is, the intersection of the green box with the downward-closure of y': $[0, 0.5] \times [0, 0.2] \times [0.5, 0.9]$.

As we saw in the previous example, we need to intersect the downward-closure of point y' with other boxes. For this reason, the following proposition will be useful.

Proposition 2. *The downward-closure of a point $\underline{y}$ intersects a box $\lfloor \underline{z}, \overline{z} \rceil$ iff*[1] $\overline{z} > \min(\underline{y}, \overline{z}) \geq \underline{z}$. *When this condition is met then*

$$\lfloor \mathbf{0}, \underline{y} \rceil \cap \lfloor \underline{z}, \overline{z} \rceil = \lfloor \underline{z}, \min(\underline{y}, \overline{z}) \rceil = B_\beta(\underline{z}, \overline{z}, \underline{y}, \underline{y})$$

with $\beta_i = 0$ if $\underline{z}_i \leq \underline{y}_i < \overline{z}_i$ and $\beta_i = \star$ otherwise.

When discovering an approximated point $\lfloor \underline{y}, \overline{y} \rceil$ we check for every box $\lfloor \underline{z}, \overline{z} \rceil$ of the boundary approximation if it intersects $\lfloor \mathbf{0}, \underline{y} \rceil$. If this is the case we add $\lfloor \mathbf{0}, \underline{y} \rceil \cap \lfloor \underline{z}, \overline{z} \rceil$ to $\underline{Y}$ and replace in the boundary approximation $\lfloor \underline{z}, \overline{z} \rceil$ by a set of boxes $I_{dwc}(\underline{z}, \overline{z}, \underline{y})$ whose union equals the difference $\lfloor \underline{z}, \overline{z} \rceil \setminus \lfloor \mathbf{0}, \underline{y} \rceil$.

The set $I_{dwc}(\underline{z}, \overline{z}, \underline{y})$ includes m small boxes, being m the number of coordinates for which $\underline{y}_i < \overline{z}_i$. These boxes have an empty intersection with $B_\beta(\underline{z}, \overline{z}, \underline{y}, \underline{y})$ with β defined in the proposition above. To do so, every box in $I_{dwc}(\underline{z}, \overline{z}, \underline{y})$ selects the complementary interval (i.e., $\beta_i = 1$) for one coordinate. The labeling of the boxes in $I_{dwc}(\underline{z}, \overline{z}, \underline{y})$ needs the definition of a function $\gamma(w, \underline{z}, \overline{z}, \underline{y})$ that creates words $\beta' \in \{0, 1, \star\}^n$ from words $w \in \{0, 1, \star\}^m$. Function $\gamma(w, \underline{z}, \overline{z}, \underline{y})$ selects $[\gamma(w, \underline{z}, \overline{z}, \underline{y})]_i = w_j$ if $\underline{y}_i < \overline{z}_i$ and i is the jth coordinate that satisfies this condition, and $[\gamma(w, \underline{z}, \overline{z}, \underline{y})]_i = \star$ if $\overline{z}_i \leq \underline{y}_i$. Words w must lead disjoint boxes, to do so we take them in the set $\{0^{j-1}1\star^{m-j} \mid 1 \leq j \leq m\}$.

Definition 6. *Given $\underline{y}$ such that $\lfloor \mathbf{0}, \underline{y} \rceil \cap \lfloor \underline{z}, \overline{z} \rceil \neq \emptyset$ we define $I_{dwc}(\underline{z}, \overline{z}, \underline{y})$ by*

$$I_{dwc}(\underline{z}, \overline{z}, \underline{y}) = \bigcup_{j=1}^{m} \{B_{\gamma(0^{j-1}1\star^{m-j}, \underline{z}, \overline{z}, \underline{y})}(\underline{z}, \overline{z}, \underline{y}, \underline{y})\}$$

This gives m boxes where a less elaborate decomposition gives 2^m boxes.

Example 3. Consider the point $\underline{y} = (0.2, 0.6, 0.2, 0.2)$ and the box $\lfloor \underline{z}, \overline{z} \rceil$ defined by $\underline{z} = (0.0, 0.0, 0.0, 0.0)$ and $\overline{z} = (0.5, 0.5, 0.5, 0.5)$. The second coordinate is the only one for which $\overline{z}_i \leq \underline{y}_i$ and hence dimension $n = 4$ with $m = 3$. Then

$$\lfloor \mathbf{0}, \underline{y} \rceil \cap \lfloor \underline{z}, \overline{z} \rceil = B_{0\star00}(\underline{z}, \overline{z}, \underline{y}, \underline{y}) = [0.0, 0.2] \times [0.0, 0.5] \times [0.0, 0.2] \times [0.0, 0.2]$$

and $I_{dwc}(\underline{z}, \overline{z}, \underline{y}) = \cup_{\alpha \in \{1\star\star\star, 0\star1\star, 0\star01\}} \{B_\alpha(\underline{z}, \overline{z}, \underline{y}, \underline{y})\}$ with

$$B_{1\star\star\star}(\underline{z}, \overline{z}, \underline{y}, \underline{y}) = [0.2, 0.5] \times [0.0, 0.5] \times [0.0, 0.5] \times [0.0, 0.5],$$

$$B_{0\star1\star}(\underline{z}, \overline{z}, \underline{y}, \underline{y}) = [0.0, 0.2] \times [0.0, 0.5] \times [0.2, 0.5] \times [0.0, 0.5],$$

$$B_{0\star01}(\underline{z}, \overline{z}, \underline{y}, \underline{y}) = [0.0, 0.2] \times [0.0, 0.5] \times [0.0, 0.2] \times [0.2, 0.5].$$

The same kinds of reasoning can be done for the upward-closure of $\overline{y}$ and $I_{upc}(\underline{z}, \overline{z}, \overline{y})$ can be defined in a similar manner of $I_{dwc}(\underline{z}, \overline{z}, \underline{y})$.

[1] The minimum is defined componentwise $[\min(\underline{y}, \overline{z})]_i = \min(\underline{y}_i, \overline{z}_i)$ for every i.

Algorithm 3 Propagation of dominance.

1: **Input**: An approximation $M' = (\underline{Y}, \overline{Y})$ of M, an approximation L of the boundary and a new discovered interval $\langle \underline{y}, \overline{y} \rangle$.
2: **Output**: Updated $M' = (\underline{Y}, \overline{Y})$ and L.
3: **for** $\lfloor \underline{z}, \overline{z} \rceil \in L$ **do**
4: **if** $\lfloor \underline{z}, \overline{z} \rceil \cap \lfloor \overline{y}, \mathbf{1} \rceil \neq \emptyset$ **then** ▷ if intersects the upward-closure of $\overline{y}$
5: add $\lfloor \underline{z}, \overline{z} \rceil \cap \lfloor \overline{y}, \mathbf{1} \rceil$ to $\overline{Y}$; ▷ add dominated sub-box
6: replace $\lfloor \underline{z}, \overline{z} \rceil$ by $I_{upc}(\underline{z}, \overline{z}, \overline{y})$ in L; ▷ update the boundary
7: **end if**
8: **if** $\lfloor \underline{z}, \overline{z} \rceil \cap \lfloor \mathbf{0}, \underline{y} \rceil \neq \emptyset$ **then** ▷ if intersects the downward-closure of $\underline{y}$
9: add $\lfloor \underline{z}, \overline{z} \rceil \cap \lfloor \mathbf{0}, \underline{y} \rceil$ to $\underline{Y}$; ▷ add dominating sub-box
10: replace $\lfloor \underline{z}, \overline{z} \rceil$ by $I_{dwc}(\underline{z}, \overline{z}, \underline{y})$ in L; ▷ update the boundary
11: **end if**
12: **end for**

4.3 Combining ideas in an updated algorithm

Algorithm 2 must be updated in order to consider all the enhacements explained in this section. In particular, the new version of the code should:

- Replace $I(\overline{x}, \underline{x}, \overline{y}, \underline{y})$ by $I'(\underline{x}, \overline{x}, \underline{y}, \overline{y}))$ in line 9, and
- Add $M', L = \mathit{propagation}(M', L, \langle \underline{y}, \overline{y} \rangle)$ before line 9 (i.e., call to Algorithm 3).

First, the introduction of $I'(\underline{x}, \overline{x}, \underline{y}, \overline{y}))$ mitigates the block explosion in the boundary caused by spaces with high dimensions: it concatenates adjacent incomparable boxes and, therefore, reduces the growing pace of the boundary per iteration. Second, Algorithm 3 shows how to propagate the dominance of a pair $\langle \underline{y}, \overline{y} \rangle$ to the rest of boxes in the boundary. The propagation of the dominance avoids unnecessary calls to the oracle, which could potentially be the slower part of the program. Nevertheless, this feature may introduce fragmentation of the boundary boxes.

Finally, approximated points $\lfloor \underline{y}, \overline{y} \rceil$ are usually separated by an error $\epsilon = \|\overline{y} - \underline{y}\| > 0$, which causes the overlap of some incomparable boxes $\lfloor \underline{z}, \overline{z} \rceil$ in the boundary (recall Fig. 3). The overlapping regions must be taken into account when calling to Algorithm 3 in order to prevent the insertion of the same overlapping box $\lfloor \underline{y}, \overline{y} \rceil$ several times in either $\underline{Y}$ or $\overline{Y}$. This characteristic did not happen in the initial version of Algorithm 2 and need not to be explicitly considered. For the sake of readability and concision, Algorithm 3 omits the code that checks and prevents the insertion of redundant portions of boxes in $\underline{Y}$ ($\overline{Y}$) caused by overlapping. In any case, the accumulated deviation resulting from this fact with respect to the total volume is negligible as long as ϵ is very small (i.e., $\epsilon = 0$ in an ideal case).

5 Computational Cost and Convergence

The following proposition gives theoretical upper-bounds on space and time complexity of our algorithm to discover m points in the Pareto front and to decrease the volume of the boundary approximation below δ. Here we consider two variants of the algorithm depending on whether we enable the enhanced spliting step or not. We denote by n the dimension of the problem and by $\kappa_n \in \{2n-4, 2^n-3\}$ the number of boxes created during the partitioning step depending on the case considered. The propagation of dominance (Section 4.2) makes a precise study of the complexity difficult and is not considered here. We let V_0 denote the volume of the initial box and V_m the volume of the boundary approximation after discovering m points.

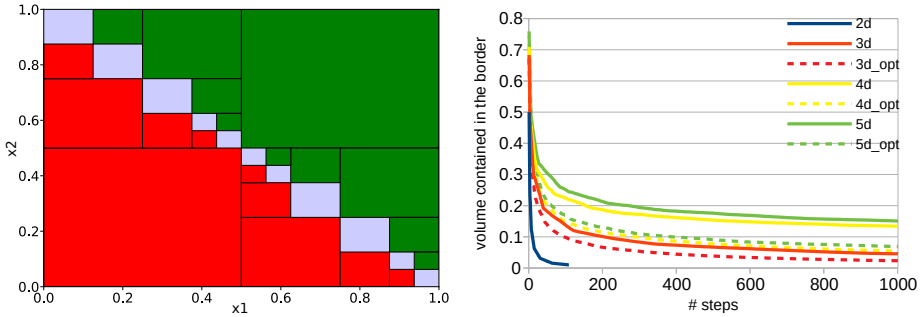
Proposition 3. *For each point discovered on the Pareto front (we call it a step), our algorithm needs $O(\log(1/\varepsilon))$ membership queries. After m steps of the algorithm, the memory consumption is $O(m\kappa_n)$. More precisely, there are m boxes in $\underline{Y}$, m boxes in $\overline{Y}$, $m\kappa_n + 1$ boxes in the boundary approximation L, and $V_m \leq m^{-1/(\kappa_n 2^{n-1})}$. The volume of the boundary approximation decreases below δ within $(V_0/\delta)^{\kappa_n 2^{n-1}}$ steps.*

A full proof of this proposition is given in the appendix. Here we give few words on how to bound the volume. It relies on the fact that when discovering a point in a box a proportion of at least $1/2^{n-1}$ of the box is removed, the worst case being when the point is on the middle of the diagonal. At the ith step we remove a proportion of at most $1/2^{n-1}$ of the box with largest volume, which is greater than the average volume $V_i/(i\kappa_n + 1) \leq V_i/((i+1)\kappa_n)$ (where $i\kappa_n + 1$ is the number of box in the boundary). The sequence of volume $(V_i)_{i\geq 0}$ thus satisfes the recurrence formula $V_i \leq V_{i-1}(1 - 1/(2^{n-1}i\kappa_n))$ which gives after m steps: $V_m \leq V_0 \prod_{i=1}^{m} \left(1 - \frac{1}{\kappa_n 2^{n-1}} \frac{1}{i}\right)$. The rest of the proof is obtained using standard mathematical inequalities involving the logarithmic and exponential functions.

We do not have a matching lower-bound for the complexity of our algorithm nor for the problem under study. We think such a doubly-exponential complexity is unavoidable for pathological cases.

The complexity of our procedure is illustrated by the following example. The surface of a simplex (i.e., points $(x_1, \ldots, x_n)$ such that $\sum_{i=1}^{n} x_i = 1$) splits the geometrical space into two parts. Fig. 6a shows the border that is discovered by our algorithm after running 10 steps in 2D. The green part corresponds to the upper part ($\sum_{i=1}^{n} x_i \geq 1$), the red part is the lower part ($\sum_{i=1}^{n} x_i < 1$), and the blue part is the *don't know* region. Increasing the dimension of the space has a direct impact on the convergence, as shown in Fig. 6b. The range of the geometrical spaces is $[0,1]^n$ and dimension $n \in [2,5]$.

According to Fig. 6b, the convergence is penalized by higher dimensions and slows down as the number of iteration steps increases. However, the method is easily parallelizable. The propagation of dominance in Algorithm 3 partially mitigates the dimensionality problem too, although it inherently involves more operations per iteration and it may introduce fragmentation of the boundary



(a) (b)

Fig. 6: (a) Volume contained in the boundary after 10 steps and (b) variation according to the simplex dimension. Opt. algorithm is not applicable to 2D.

boxes. In conclusion, the propagation of dominance is recommended when the cost of asking a membership query to the oracle consumes a great amount of time, and, therefore, reducing the number of membership queries becomes imperative.

6 Experiments

6.1 ParetoLib library

The procedure explained in this paper has been packaged in a free and publicly available Python library [6,5]. ParetoLib separates the implementation of the multi-dimensional search from the definition of the oracle, which makes the code in the core of the library unique. The oracle is designed as a generic abstract interface that is later on instantiated by the user for every specific optimization problem. Our library currently provides several predefined oracles.

A set of oracles for interacting with Signal Temporal Logic (STL) [15] monitors facilitates the application of our method for the synthesis of parameters in STL specifications. Given a template of a parametrized STL formula, ParetoLib identifies the set of valuations that (in)validate it over a signal. The coordinates of points in the multi-dimensional space are treated by the oracle as particular instances of the parameters in the STL expression. Next section illustrates the usage of our approach for the parameter synthesis for extended STL specifications (see [3,2]).

6.2 Case studies

This section shows the parameter synthesis of extended STL specifications [3,2] using the method developped in this paper. We analyze two parametric properties that exhibit a monotonic behaviour. These parametric STL properties are evaluated over a decaying signal (Fig. 7a) with a damped oscillation of period 250. For $t \in [0, 1000)$, x_{decay} is defined as $x_{decay}(t) = \frac{1}{e} sin(250t + 250)e^{-\frac{1}{250}x}$.

Stabilization. We want to find the tightest parameters p_1 and p_2 such that within p_1 time units, the signal stabilizes around an unknown value within an interval of height p_2. Formally, the parametric STL formula $\phi_{stab} = \mathrm{F}_{[0,p_1]}\mathrm{G}(\mathrm{On}_{[0,\infty]}\mathrm{Max}\ x - \mathrm{On}_{[0,\infty]}\mathrm{Min}\ x < p_2)$ is adapted from [3] by turning constants into parameters. The smaller p_1 and p_2 are, the earlier and tighter the stabilization is. There is a trade-off between stabilizing earlier at the price of higher perturbation, or stabilizing later and allowing less perturbation. The set of best possible trade-offs (i.e. Pareto front) is computed by ParetoLib in Fig. 7b. The boundary represents the limit between feasible and unfeasible valuations (p_1, p_2). The blue dot in Fig. 7b shows the stabilization after the first peak.

Spikes. A spike is defined with two parameters, the width p_3 and the height p_4. Unformally, the thinner and taller a spike is the more spiky it is. Formally, the parametric extended STL formula is $\phi_{spike} = |x - \mathrm{D}^0_{p_3} x| \leq \epsilon \wedge (\mathrm{On}_{[0,p_3]}\mathrm{Max}\ x - x \geq p_4)$. Error ϵ appears due to discretization of signal x. We use $p'_4 = 0.5 - p_4$ instead of p_4 to have a problem of parameter minimization. Fig. 7c shows the image produced by ParetoLib on the same signal as above. The slope captures the valuations of the first oscillation, which yields the more restrictive spikes (tallest height and narrowest width). The blue dot in Fig. 7c corresponds to the parameter valuation of the second oscillation. It is not on the Pareto front because a thinner and higher spike can be found inside the first ocillation of the signal.

Computation time. For these examples, ParetoLib requires less than 2.4 seconds for computing 500 steps of ϕ_{stab}, and less than 1 second for computing 200 steps of ϕ_{spike}. Approximately, 45% of the cost corresponds to the execution time of the oracle (i.e., calls to the STL monitor). Another 45% belongs to printing log traces on the terminal, and the remaining 10% is the overhead introduced by the searching algorithm. The experiments are run using a single core of a PC with Intel Core i7-8650U CPU, 32GB RAM and Python 2.7.

7 Conclusions

This paper presents a novel algorithm for learning the boundary (i.e., Pareto front) between feasible ($\overline{X}$) and infeasible ($\underline{X}$) configurations for multi-criteria optimization problems in a cost space X. The algorithm is based on an external oracle that answers membership queries $x \in \overline{X}$. According to the answers and relying on monotonicity, it discovers a set of Pareto points that partitions the cost space into feasible and unfeasible subspaces. Our algorithm is based on the generalization of a binary search for spaces of any dimension. The method consists of dividing the multi-dimensional cost space into multiple smaller boxes and executing a binary search over the diagonal of every box. The multi-dimensional space is exhaustively explored until a quality criterion is met (e.g. the volume of the unexplored region is smaller than a threshold).

Our method has been applied to the parametric identification of STL formulas that exhibit monotonic behavior. The procedure explained in this paper

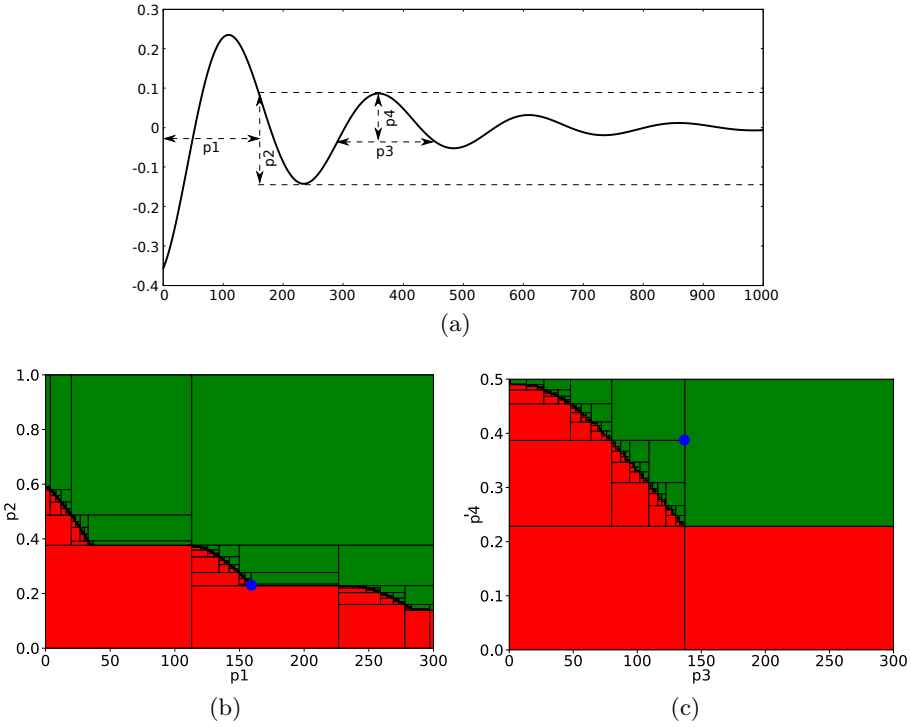


Fig. 7: Decaying signal (a) and Pareto front for ϕ_{stab} (b) and ϕ_{spike} (c).

has been implemented in a free and publicly available Python library. As future work, we plan to apply our approach to various problems including parameter identification of more temporal logics and other multi-criteria optimization problems. In particular, we will study the impact of statistical oracles that answer quantitative membership queries with a confidence interval.

References

1. Eugene Asarin, Alexandre Donzé, Oded Maler, and Dejan Nickovic. Parametric identification of temporal properties. In *International Conference on Runtime Verification (RV)*, pages 147–160, 2011.
2. Alexey Bakhirkin. StlEval, STL Evaluator. `https://gitlab.com/abakhirkin/StlEval`, 2019.
3. Alexey Bakhirkin and Nicolas Basset. Specification and efficient monitoring beyond STL. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 11428 of *LNCS*, pages 79–97. Springer, 2019.
4. Alexey Bakhirkin, Nicolas Basset, Oded Maler, and José Ignacio Requeno. Learning Pareto Front and Application to Parameter Synthesis of STL. Technical report available at `https://hal.archives-ouvertes.fr/hal-02125140`, 2019.

5. Alexey Bakhirkin, Nicolas Basset, Oded Maler, and José Ignacio Requeno. Paretolib: A Python Library for Parameter Synthesis. In *International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, Amsterdam, Netherlands, 2019. Springer. Tool paper submitted to 17th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS).
6. Nicolas Basset, Oded Maler, and José-Ignacio Requeno Jarabo. ParetoLib library. `https://gricad-gitlab.univ-grenoble-alpes.fr/verimag/tempo/multidimensional_search`, 2018.
7. C. Daskalakis, I. Diakonikolas, and M. Yannakakis. How Good is the Chord Algorithm? *SIAM Journal on Computing*, 45(3):811–858, 2016.
8. Kalyanmoy Deb. Multi-objective optimization. In *Search methodologies*, pages 403–449. Springer, 2014.
9. Rüdiger Ehlers. Computing the complete pareto front. *CoRR*, abs/1512.05207, 2015.
10. Salvatore Greco, J Figueira, and M Ehrgott. *Multiple criteria decision analysis*. Springer, 2016.
11. Lu He, Alan M Friedman, and Chris Bailey-Kellogg. A divide-and-conquer approach to determine the pareto frontier for optimization of protein engineering experiments. *Proteins: Structure, Function, and Bioinformatics*, 80(3):790–806, 2012.
12. S. Koziel and A. Bekasiewicz. Pareto-ranking bisection algorithm for expedited multiobjective optimization of antenna structures. *IEEE Antennas and Wireless Propagation Letters*, 16:1488–1491, 2017.
13. Julien Legriel, Colas Le Guernic, Scott Cotton, and Oded Maler. Approximating the Pareto front of multi-criteria optimization problems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 69–83, 2010.
14. Oded Maler. Learning Monotone Partitions of Partially-Ordered Domains (Work in Progress). working paper or preprint available at `https://hal.archives-ouvertes.fr/hal-01556243`, July 2017.
15. Oded Maler and Dejan Ničković. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, Joint International Conferences on Formal Modelling and Analysis of Timed Systems (FORMATS) and Formal Techniques in Real-Time and Fault-Tolerant Systems, (FTRTFT)*, volume 3253 of *LNCS*, pages 152–166. Springer, 2004.
16. Marcell Vazquez-Chanlatte. Multidimensional thresholds. `https://github.com/mvcisback/multidim-threshold`, 2018.
17. Marcell Vazquez-Chanlatte, Jyotirmoy V. Deshmukh, Xiaoqing Jin, and Sanjit A. Seshia. Logical clustering and learning for time-series data. In *International Conference on Computer Aided Verification (CAV)*, volume 10426 of *LNCS*, pages 305–325. Springer, 2017.
18. Marcell Vazquez-Chanlatte, Shromona Ghosh, Jyotirmoy V. Deshmukh, Alberto L. Sangiovanni-Vincentelli, and Sanjit A. Seshia. Time-series learning using monotonic logical properties. In *International Conference on Runtime Verification (RV)*, volume 11237 of *LNCS*, pages 389–405. Springer, 2018.
19. Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M Fonseca, and Viviane Grunert Da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.

A Appendix

Proof of proposition 3. The statements on the number of boxes is quite straightforward to prove: at each operations we remove 1 box and create $\kappa_n + 1$ boxes so there are $\kappa_n m$ boxes after m steps (unless for $m = 0$ where there is a single box). The results on the volume V_m and the upper bound on the convergence speed are more involved. We first state and prove two lemmas.

Lemma 1. *Given a box $\lfloor \underline{x}, \overline{x} \rceil$ and a point y discovered in the diagonal. The volume of the removed part is minimal when y is at the center of the box, this volume is $(1/2)^{n-1}\texttt{Vol}(\lfloor \underline{x}, \overline{x} \rceil)$.*

Proof. A point y on the diagonal is of the form $\lambda\underline{x} + (1-\lambda)\overline{x}$. First we notice that $\texttt{Vol}(\lfloor \underline{x}, y \rceil) = (1-\lambda)^n\texttt{Vol}(\lfloor \underline{x}, \overline{x} \rceil)$ and $\texttt{Vol}\lfloor y, \overline{x} \rceil = \lambda^n\texttt{Vol}(\lfloor \underline{x}, \overline{x} \rceil)$. So the volume of the removed part is $(\lambda^n + (1-\lambda)^n)\texttt{Vol}(\lfloor \underline{x}, \overline{x} \rceil)$. So we want to find λ such that the quantity $\lambda^n + (1-\lambda)^n$ is minimal. Its derivative $n(\lambda^{n-1} - (1-\lambda)^{n-1})\texttt{Vol}(\lfloor \underline{x}, \overline{x} \rceil)$ is negative for $\lambda < 1/2$, null for $\lambda = 1/2$ and positive for $\lambda > 1/2$ which proves the minimality of the volume for $\lambda = 1/2$ with the value $(1/2)^{n-1}\texttt{Vol}(\lfloor \underline{x}, \overline{x} \rceil)$. ▟

Lemma 2. *Let V_m be the volume after m steps and V_0 the initial volume of the boundary, then $V_m \leq V_0 \prod_{i=1}^{m} \left(1 - \frac{1}{\kappa_n 2^{n-1}} \frac{1}{i}\right) \leq V_0 m^{-1/(\kappa_n 2^{n-1})}$.*

Proof. We prove the result by induction on m, the base case for $m = 0$ is trivial satisfied using the usual convention that an empty product is 1. We assume that the property holds at rank m, and we consider the box chosen at step $m+1$ whose volume is denoted $V_{max,m}$. This box is of maximal volume amongst the m box that form the border approximation, so its volume is greater than the average volume: $V_{max,m} \geq V_m/(m \cdot \kappa_n + 1) \geq V_m/((m+1) \cdot \kappa_n)$. Using the lemma above we get that V_{m+1} is obtained from V_m by removing at least $(1/2)^{n-1}V_{max,m}$ to the selected box, so $V_{m+1} \leq V_m(1 - (1/2)^{n-1})(1/((m+1) \cdot \kappa_n)))$. After m steps

$$V_m/V_0 \prod_{i=1}^{m} \left(1 - \frac{1}{\kappa_n 2^{n-1}} \frac{1}{i}\right) \leq \exp\left(-\frac{1}{\kappa_n 2^{n-1}} \sum_{i=1}^{m} \frac{1}{i}\right).$$

We now use the classical fact that the harmonic number $\sum_{i=1}^{m} \frac{1}{i}$ is greater than $\ln(m+1)$ so also greater than $\ln(m)$. The function $t \mapsto \exp(-t)$ is decreasing so

$$\exp\left(-\frac{1}{\kappa_n 2^{n-1}} \sum_{i=2}^{m} \frac{1}{i}\right) \leq \exp\left(-\frac{1}{\kappa_n 2^{n-1}} \ln(m)\right) = m^{-1/(\kappa_n 2^{n-1})}.$$

▟

Now we proceed to the end of Proof of Proposition 3.
The volume upper-bound $V_0 m^{-1/(\kappa_n 2^{n-1})}$ is less than δ iff $m \geq (V_0/\delta)^{\kappa_n 2^{n-1}}$.

▟