Learning Pareto Front and Application to Parameter Synthesis of STL

José Ignacio Requeno, Alexey Bakhirkin, Nicolas Basset, Oded Maler, José-Ignacio Requeno

Learning Pareto Front and Application to Parameter Synthesis of STL

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 an upward-closed set $\overline{X}$ and its downward-closed complement. 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 (STL) expressions where the influence of parameters is monotone. Our method has been implemented in a free and publicly available Python library.

1 Introduction

In *multi-criteria optimization* problems, solutions are evaluated according to several criteria and 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. The Pareto front can be viewed as the boundary of a monotone partition. For a minimization problem, $\underline{X}$ corresponds to infeasible costs and $\overline{X}$ represents the feasible costs. The goal of this paper is to discover the border between $(\underline{X}, \overline{X})$ by learning $(\underline{Y}, \overline{Y})$ (Fig. 1).

Some classes of parametric identification problems are solvable by mapping them to multi-criteria optimisation problems; and benefit of the new strategy we present here. 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

* Oded Maler passed away at the beginning of September 2018. This work was initiated by him [8] continued with and finished by the rest of us.

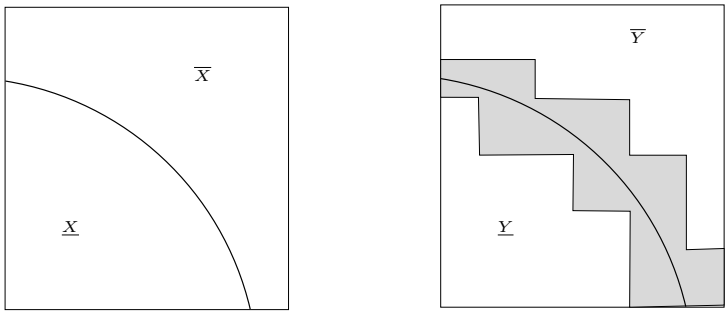


Fig. 1: A monotone partition and its approximation.

in a parameterized predicate $u \leq p$, then for any $p' > p$ and any u, $\varphi_p(u)$ implies $\varphi_{p'}(u)$. When no parameter appears in two constraints in opposing sides of an inequality, and after some pre-processing, the set of parameters that lead to satisfaction is upward-closed. Its set of minimal elements indicates the set of tightest parameters that lead to satisfaction of $\varphi_p(u)$, which is a valuable information about u. This set of minimal elements defines the Pareto front. In [1] we already explored the idea for the domain of real-valued signals $u(t)$ and temporal formulas such as $\exists t < p_1 \ u(t) < p_2$.

In this paper, we present an algorithm that follows the same phylosophy than [7] and provides an alternative way to approximate Pareto fronts with quality metrics. Our algorithm has the additive feature of finding a cloud of exact Pareto points along the boundary (Section 4) instead of under/over estimations. A preliminary version of the procedure was proposed and implemented in [8,11] without the materials of Section 5–7. It was successfully used for learning valuations of parametric Signal Temporal Logic (STL) [9] specifications over time-series data [13,12]. Experiments in Section 7 apply the improvements of the current work to the parameter synthesis of extended STL specifications [3].

2 Related Work

Multi-criteria optimization techniques are commonly categorized in two families [5,6]. 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 point, but the result depends on the choice of the weighted coefficients.

The second family of methods is based on evolutionary algorithms, which generate a set of solutions distributed along the entire Pareto front. However, these methods cannot guarantee the Pareto optimality of the solutions: 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 [14].

In [7] we developed a procedure for computing such an approximation with a explicit guarantee of getting Pareto optimal points in an epsilon-sized interval. The procedure uses a variant of binary search that submits queries to a constraint solver concerning the existence of solutions of a given cost x. The costs used in the queries were selected in order to reduce the distance between the boundaries of $\overline{Y}$ and $\underline{Y}$ (Fig. 1) and improve approximation quality.

3 Preliminaries

3.1 Notation

Multi-objective problems can be formalised as a bipartition of a geometrical space following the next notation. 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 $\overline{X}$, $\underline{X} = X - \overline{X}$ is downward-closed, and we use the term *monotone bi-partition* (or simply partition) for the pair $M = (\underline{X}, \overline{X})$. The Pareto front of the pair $M = (\underline{X}, \overline{X})$ is the set $bd(M) = \min(\overline{X}) = \{x \in M : x \text{ is minimal in } \overline{X}\}$. 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 $\overline{X}$ (Fig. 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. There are two degenerate cases of monotone partitions, $(X, \emptyset)$ and $(\emptyset, X)$ that we ignore from now on, and thus assume that $\mathbf{0} \in \underline{X}$ and $\mathbf{1} \in \overline{X}$, where $\mathbf{r}$ denotes $(r, \ldots, r)$. We adopt the conventions that $bd(M)$ belongs to $\overline{X}$. Our approach is a neat high-dimensional generalization of the problem of locating a boundary point that splits a straight line into two 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.

3.2 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 high-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 $\langle \underline{y}, \overline{y} \rangle$

4 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;*

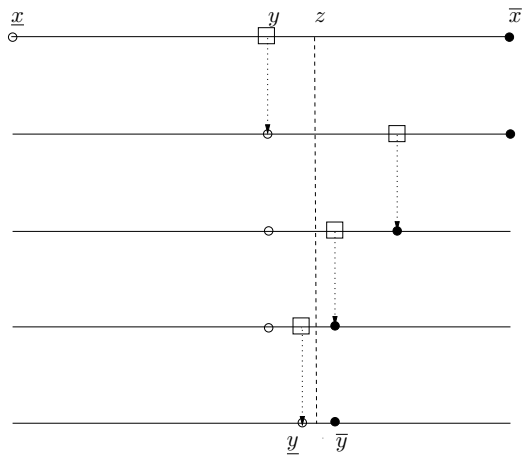


Fig. 2: Binary search and the successive reduction of the uncertainty interval.

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

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 $\texttt{Vol}(L) \leq \delta$

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*

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

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:

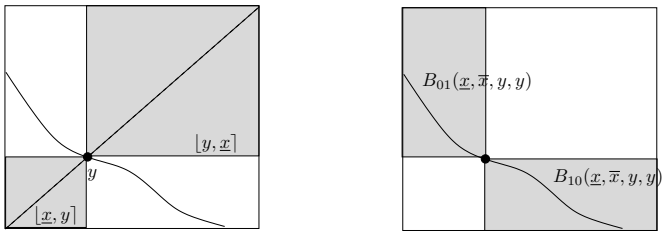
Definition 4 (Sub-boxes). *Let* $\alpha \in \{0, 1\}^n$ *and* 4 *n-dimensional points* $\underline{x} \leq \underline{y} \leq \overline{y} \leq \overline{x}$. *The sub-boxes of* $\lfloor \underline{x}, \overline{x} \rceil$ *induced by* α *and* $\lfloor \underline{y}, \overline{y} \rceil$ *are*

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

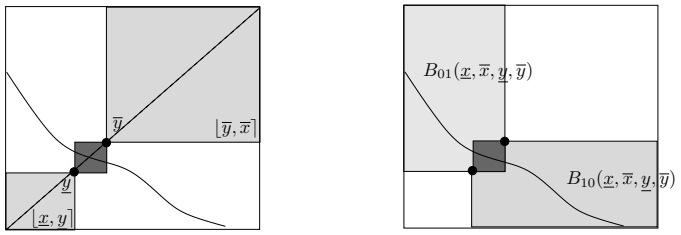
We can now define $I(\overline{x}, \underline{x}, \overline{y}, \underline{y})$.

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*

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



(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.

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.

5 Two Enhacements in Dimensions Higher than 2

The goal of introducing changes to the learning procedure in dimension higher than 2 is twofold. First we decompose into fewer boxes the part that stays in the boundary approximation (new incomparable set $I'(\overline{x}, \underline{x}, \overline{y}, \underline{y})$, Sect. 5.1). Second we avoid a pitfall we identify in Sect. 5.2 by propagating the dominance to the boxes in the boundary approximation every time a point of the Pareto front is discovered. We sum up the new version of the algorithm in Sect. 5.3.

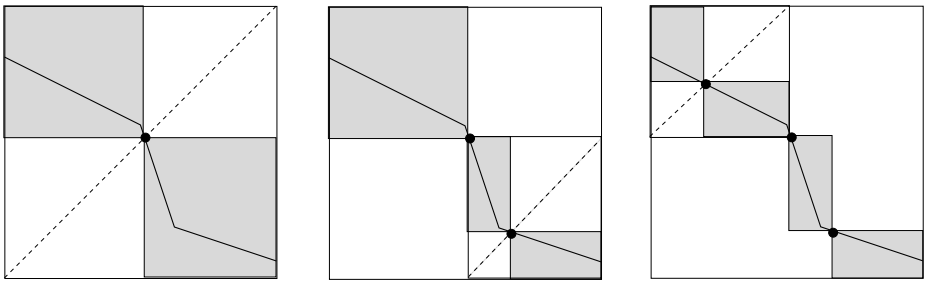


Fig. 4: Successive approximation of the partition boundary by running binary search on diagonals of incomparable boxes.

5.1 Enhanced decomposing step

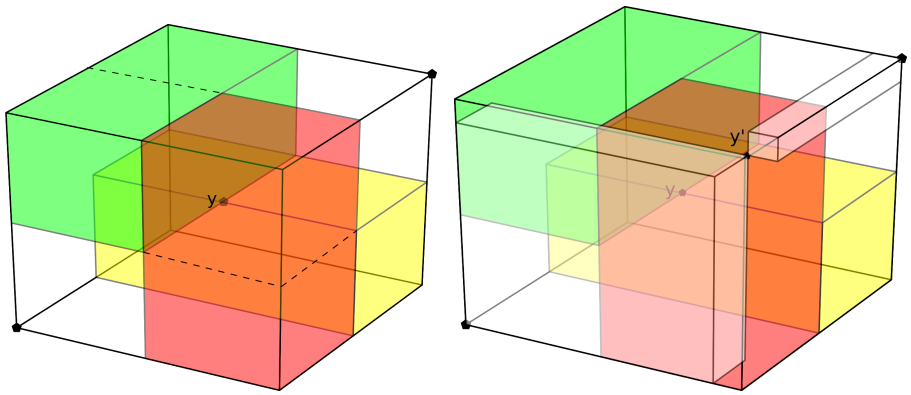
In this section we come back to the treatment of the remainder of the box $\lfloor \underline{x}, \overline{x} \rceil$ when the two boxes above $\overline{y}$ and below $\underline{y}$ are removed. This remainder part was covered by the union of boxes of $I(\overline{x}, \underline{x}, \overline{y}, \underline{y})$. In two dimensions, there are only two boxes in this set 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 $I_\star(\underline{x}, \overline{x}, \underline{y}, \overline{y}) = I_0(\underline{x}, \overline{x}, \underline{y}, \overline{y}) \cup I_1(\underline{x}, \overline{x}, \underline{y}, \overline{y})$), and $B_\beta(\underline{x}, \overline{x}, \underline{y}, \overline{y}) = \prod_{i=1}^{n} I_{\beta_i}(\underline{x}_i, \overline{x}_i, \underline{y}_i, \overline{y}_i)$ with $\beta \in \{0, 1, \star\}^n$ in higher dimension. Examples of boxes created by these expressions are given in Fig. 5.

Now we show how to use the encoding with words on the alphabet $\{0, 1, \star\}$ to define the decomposition of $\lfloor \underline{x}, \overline{x} \rceil \setminus (\lfloor \underline{x}, \underline{y} \rceil \cup \lfloor \overline{y}, \overline{x} \rceil)$ using boxes formed as unions of 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 $\mathbb{A}^3 = \{10\star, \star10, 0\star1\}$ 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:*

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



(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.

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

$$\mathbb{A}^4 = \star\mathbb{A}^3 \cup \{1000\} \cup \{0111\} = \{\star 10\star, \star\star 10, \star 0\star 1, 1000, 0111\}.$$

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

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 $\lfloor \mathbf{0}, y]$ and the upper part $\lfloor y, \mathbf{1}]$, 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).

5.2 Propagating dominance to boxes of the boundary approximation

In dimension higher than 2, it can be the case that when an approximated point $\lfloor \underline{y},\overline{y}]$ is found within a box $\lfloor \underline{x},\overline{x}]$, 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

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

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

$$B_{0\star1\star}(\underline{z},\overline{z},\underline{y},\underline{y}) = [0.0,0.2]\times[0.0,0.5]\times[0.2,0.5]\times[0.0,0.5],$$
$$B_{0\star01}(\underline{z},\overline{z},\underline{y},\underline{y}) = [0.0,0.2]\times[0.0,0.5]\times[0.0,0.2]\times[0.2,0.5].$$

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 $I_{dwc}(\underline{z},\overline{z},\underline{y})$.

5.3 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 search spaces with high dimensions: it concatenates adjacent incomparable boxes and, therefore, adds fewer boxes to 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. Additionally, points $\underline{y}$ and $\overline{y}$ are usually separated by an error $\epsilon > 0$. Consequently, some incomparable boxes $\lfloor\underline{z},\overline{z}\rceil$ in the boundary may overlap. The overlapping areas must be taken into account when moving boxes directly from the frontier to a closure $\underline{Y}$ or $\overline{Y}$, while this peculiarity 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 with respect to the total volume is negligible as long as ϵ is very small (i.e., $\epsilon = 0$ in a ideal case).

6 Computational Cost and Convergence

First of all we show the termination of our algorithms and an upper-bound on the number of steps it takes. The convergence of Algorithms 2–3 is guaranteed by the fact that the volume contained in the boundary decreases at each step by a constant factor as proved in the next lemma.

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 $(1/2)^{n-1}\,\mathtt{Vol}(\lfloor\underline{x},\overline{x}\rceil)$.*

Proof. A point y on the diagonal is of the form $\lambda\underline{x} + (1-\lambda)\overline{x}$. First we notice that $\mathtt{Vol}(\lfloor\underline{x},y\rceil) = (1-\lambda)^n\mathtt{Vol}(\lfloor\underline{x},\overline{x}\rceil)$ and $\mathtt{Vol}\lfloor y,\overline{x}\rceil = \lambda^n\mathtt{Vol}(\lfloor\underline{x},\overline{x}\rceil)$. So the

Algorithm 3 Propagation of dominance.

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

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 derivated $n(\lambda^{n-1} - (1-\lambda)^{n-1})\texttt{Vol}(\lfloor \underline{x}, \overline{x} \rceil)$ 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 $V_m \leq V_0 \prod_{i=1}^{m} \left(1 - \frac{1}{\kappa_n 2^{n-1}} \frac{1}{i}\right)$. The number of boxes within the boundary after m steps is $m \cdot \kappa_n$.*

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: $V_{max,m} \geq V_m / m \cdot \kappa_n$. 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 $V_{m+1} \leq V_m - (1/2)^{n-1} V_{max,m} \leq V_m(1 - (1/2)^{n-1}(1/m \cdot \kappa_n))$.

Proposition 3. *Algorithm ?? terminates within $O((V_0/\delta)^{\kappa_n 2^{n-1}})$ steps.*

Proof. After m steps

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

This quantity is less than δ / V_0 if

$$\left(\frac{1}{\kappa_n 2^{n-1}} \sum_{i=1}^{m} \frac{1}{i}\right) \geq \ln(V_0/\delta).$$

We use the fact that $\sum_{i=1}^{m} \frac{1}{i} \geq \ln m$. So a sufficient condition is $\ln(m) \geq \kappa_n 2^{n-1} \ln(V_0/\delta)$ which holds as soon as $m \geq (V_0/\delta)^{\kappa_n 2^{n-1}}$.

Our algorithm makes multiple calls to an external oracle which spend time answering each queries. A first study of the computational complexity of our algorithm is to count the number of queries made to the oracle. Each round of the main loop of Algorithm **??** or **??**, a call to the one-dimensional binary search along a diagonal is made, it makes $\mathcal{O}(\log_2(1/\epsilon))$ queries to the oracle.

In the version with dominance propagation the worst-case computational complexity is $\theta(N^2 \log_2 N)$ with N the number of boxes added to the boundary but N is bigger here. Each time a point is added in the boundary at most $d \times N + (2d-3)$ boxes are added.

It is of $O(N \log_2 N)$ for the basic algorithm.

No need to sort the list each time. Add elements in a sorted data structure (Binary research tree, heap...) is logarithmic.

The computational cost of Algorithm 2 is determined by the sorting of boxes in the boundary according to their volume (line **??**), and the binary search of the Pareto point on the diagonal (line **??**). Efficient sorting functions have a $\mathcal{O}(m \log m)$ computational cost with m the number of boxes in the border. The binary search of a Pareto point in the diagonal of a box requires $\mathcal{O}(\log_2(1/\epsilon))$ queries to the oracle. The rest of operations such as the addition of boxes to the upper, lower or incomparable closures are constant-time because they correspond to the concatenation of lists. Therefore, each iteration step of the main loop has a maximal cost of either $\mathcal{O}(m \log m) + T_o \log(1/\epsilon))$, with T_o the complexity of the oracle to answer a query.

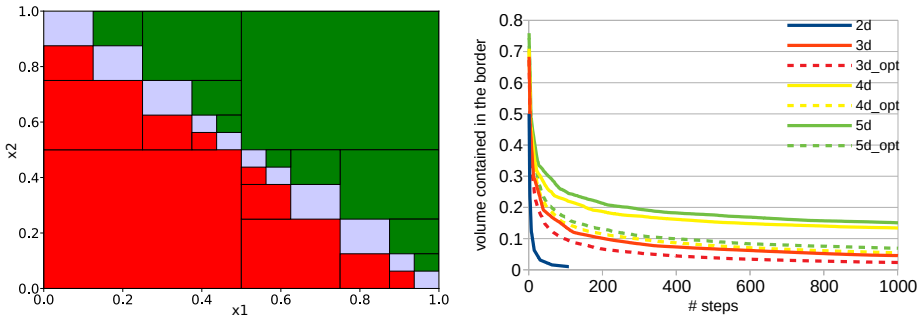
The computational cost of Algorithm 3 is similar to the previous one. The only difference is the addition of an internal loop (line 3) that propagates the dominance to the remaining boxes in the border. It includes list operations and tests of unitary cost, leading to an aggregated cost of $\mathcal{O}(m)$ which is dominated by the cost $\mathcal{O}(m \log m)$ of the sorting function. The number of iteration steps of the main loop is limited by the accuracy δ and the speed of decreasing the volume in the border. The convergence pace is considered in the following paragraphs.

Nevertheless, the convergence rate relies on several factors, whose most prominent one is the geometrical dimension of the problem (i.e., number of parameters). Let's assume a boundary box of initial volume V_0 and dimension n whose Pareto point is discovered in the center of the box. It decomposes the original box in up to 2^n sub-boxes of equal volume. This partition produces the minimal amount of volume that will be removed from the current box and appended to the upper and lower closures, i.e., $2V_0/2^n$. After the first iteration, the volume of the original box is reduced by $V_1 = V_0(1 - 2^{1-n})$, which can be generalized by $V_i = V_0(1 - 2^{1-n})^i$ at the i-th recursion step.

Each border box produces $2n-3$ smaller boxes in the boundary for $n \geq 3$. Processing m boundary boxes approximatively involves reaching a depth $i = \log_{2n-3} m$ for $m = (2n-3)^i$. The volume in the boundary after m iterations is bounded by $\texttt{Vol}(L) \leq kV_{i+1}$ with $1 \leq k \leq 2n-3$ boxes of volume V_{i+1}.

The complexity of our procedure is illustrated by the following artificial 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 closures. Fig. 6a shows the border that is

discovered by our algorithm after running 10 steps in 2D. The green part corresponds to the upper closure ($\sum_{i=1}^{n} x_i \geq 1$), the red part is the lower closure ($\sum_{i=1}^{n} x_i < 1$), 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 $n \in [2, 5]$.



(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.

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 boxes. In conclusion, the propagation of the dominance is recommended in geometrical spaces of high dimension; or 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.

7 Experiments

7.1 ParetoLib library

The procedure explained in this paper has been packaged in a free and publicly available Python library [4]. Our library has designed the oracle as an abstract interface that encapsulates the set of minimum operations that every oracle should provide. Later on, the oracle can be specialized for multiple application domains by simply implementing the abstract interface. Our library currently supports a small number of specialized oracles for inferring the Pareto front. The more prominent ones are **OracleSTL**, that defines the membership of a point x to the upper or lower closures based on the success in evaluating a Signal

Temporal Logic (STL) [9] formula over a signal, and **OracleSTLe**, that works similarly over extended STL specifications [3]. The STL formula is parametrized with a set of variables corresponding to the coordinates of the point x (i.e., the parameters of the STL formula are instantiated by x). Every point x satisfying the STL formula belongs to the upper part of the partition, while every point x falsifying it will fall in the lower part. Internally, OracleSTL asks queries to JAMT [10] and OracleSTLe asks queries to StlEval [2].

7.2 Case studies

In this section we present the feasibility of our approach and the capabilities of our tool. To this end, we apply our procedure for discovering the Pareto front in the context of parameter synthesis for extended STL specifications. We analyze the following parametric STL formulas that are evaluated over a decaying signal.

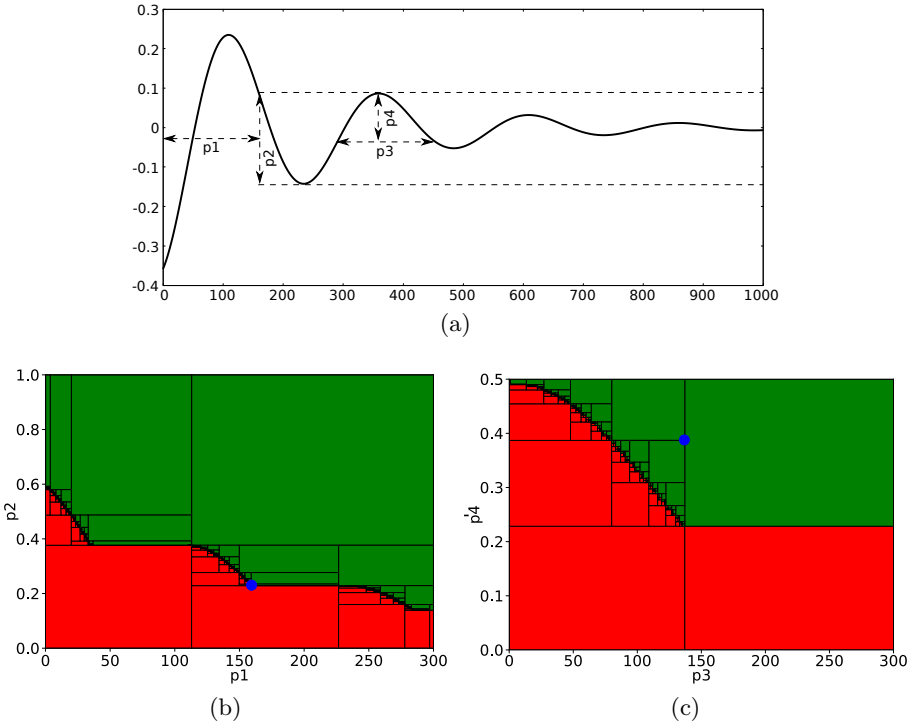
- $\phi_{stab} = \mathrm{F}_{[0,p_1]}\mathrm{G}(\mathrm{On}_{[0,\infty]}\mathrm{Max}\ x - \mathrm{On}_{[0,\infty]}\mathrm{Min}\ x < p_2)$, within p_1 time units, signal x stabilizes and the amplitude will fall below $0.5 \cdot p_2$.
- $\phi_{spike} = |x - \mathrm{D}^0_{p_1} x| \leq \epsilon \wedge (\mathrm{On}_{[0,p_1]}\mathrm{Max}\ x - x \geq p_2)$, signal x has a spike of width p_1 and height at least p_2. Error ϵ appears due to discretization of signal x.

Signal x_{decay} (Fig. 7a) is a damped oscillation with period 250. For $t \in [0, 1000)$, x_{decay} is defined as $x_{decay}(t) = \frac{1}{e} sin(250t + 250)e^{-\frac{1}{250}x}$. Fig. 7b–7c show the image produced by ParetoLib when analyzing the stabilization (ϕ_{stab}) and spikes (ϕ_{spike}) of the signal. The boundary represents the limit between feasible and unfeasible valuations (p_1, p_2), where p_1 and p_2 are parameters. Note that $p_4 = 0.5 - p'_4$ for Fig. 7c in order to guarantee monotonicity. The slope in Fig. 7c captures the valuations of the first spike, which is the more restrictive one (tallest height and narrowest width). Blue dot in Fig. 7c corresponds to the parameter valuation of the second spike. Similarly, blue dot in Fig. 7b shows the stabilization after the first peak.

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}. The experiments are run using a single core of a PC with Intel Core i7-8650U CPU, 32GB RAM and Python 2.7.

8 Conclusions

In this paper, we have presented a novel algorithm for learning the boundary (i.e., Pareto front) between an upward-closed set $\overline{X}$ and its downward-closed complement $\underline{X}$ for a multi-dimensional 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 configurations. To this end, our algorithm divides the multi-dimensional cost space into multiple



(a) (b) (c)

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

smaller boxes and executes a binary search over the diagonal of every box. The multi-dimensional space is exhaustively explored until a stopping criterion is met (e.g. the volume of the unexplored region is smaller than a threshold).

Our algorithm has a polynomial cost with respect to the number of boxes in which the space is divided. The rate of convergence towards the Pareto front depends on the number of boxes, the dimension of the space and the shape of the boundary. At each iteration step of the main algorithm, at least a new Pareto point in the boundary is discovered.

Finally, our method has been applied to the parametric identification of STL formulas. The procedure explained in this paper has been implemented in a free and publicly available Python library. As future work, we plan to apply our tool to various problems including parameter identification of more temporal logics and multicriteria optimisation. In particular, we will study the impact of statistical oracles that answer 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*, 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 Tomáš Vojnar and Lijun Zhang, editors, *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 79–97, Cham, 2019. Springer International Publishing.
4. Nicolas Basset, Oded Maler, and José-Ignacio Requeno Jarabo. ParetoLib library. `https://gricad-gitlab.univ-grenoble-alpes.fr/verimag/tempo/multidimensional_search`, 2018.
5. Kalyanmoy Deb. Multi-objective optimization. In *Search methodologies*, pages 403–449. Springer, 2014.
6. Salvatore Greco, J Figueira, and M Ehrgott. *Multiple criteria decision analysis*. Springer, 2016.
7. 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*, pages 69–83, 2010.
8. 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.
9. Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 152–166. Springer, 2004.
10. Dejan Ničković, Olivier Lebeltel, Oded Maler, Thomas Ferrère, and Dogan Ulus. Amt 2.0: Qualitative and quantitative trace analysis with extended signal temporal logic. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 303–319. Springer, 2018.
11. Marcell Vazquez-Chanlatte. Multidimensional thresholds. `https://github.com/mvcisback/multidim-threshold`, 2018.
12. Marcell Vazquez-Chanlatte, Jyotirmoy V. Deshmukh, Xiaoqing Jin, and Sanjit A. Seshia. Logical clustering and learning for time-series data. In *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, pages 305–325, 2017.
13. 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*, pages 389–405, Limassol, Cyprus, November 2018.
14. 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.