Constraint propagation using dominance in interval Branch & Bound for nonlinear biobjective optimization

Benjamin Martin[a], Alexandre Goldsztejn[b], Laurent Granvilliers[c], Christophe Jermann[c]

[a] *bmf.ben.martin@gmail.com*
NOVA-LINCS, Universidade Nova de Lisboa, Lisboa, Portugal
[b] *alexandre.goldsztejn@irccyn.ec-nantes.fr*
IRCCyN, CNRS, Nantes, France
[c] *name.surname@univ-nantes.fr*
LINA, Université de Nantes / CNRS, Nantes, France

Abstract

Constraint propagation has been widely used in nonlinear single-objective optimization inside interval Branch & Bound algorithms as an efficient way to discard infeasible and non-optimal regions of the search space. On the other hand, when considering two objective functions, constraint propagation is uncommon. It has mostly been applied in combinatorial problems inside particular methods. The difficulty is in the exploitation of dominance relations in order to discard the so-called non-Pareto optimal solutions inside a decision domain, which complicates the design of complete and efficient constraint propagation methods exploiting dominance relations.

In this paper, we present an interval Branch & Bound algorithm which integrates *dominance contractors*, constraint propagation mechanisms that exploit an upper bound set using dominance relations. This method discards from the decision space values yielding solutions dominated by some solutions from the upper bound set. The effectiveness of the approach is shown on a sample of benchmark problems.

Keywords: Nonlinear optimization, Biobjective optimization, Constraint propagation, Interval Branch & Bound

1. Introduction

Rigorous numerical global optimization aims at finding all the optimal, with respect to some objectives, and feasible, with respect to some constraints, solutions of a nonlinear continuous optimization problem with some numerical guarantees like a prescribed computational precision or solution existence proof. In single-objective optimization, rigorous global methods such as interval *Branch & Bound* (B&B) have been designed and they are well studied in the literature, see e.g. [19, 23, 38, 49]. These methods subdivide the search space into smaller

parts which are discarded using bounds on the objective and pruning techniques so as to isolate the portion of the feasible space that contains the global optima. The use of interval analysis allows rigorous computations (e.g., verified linear relaxations) and powerful pruning techniques based on constraint propagation. However, the literature on interval B&B for solving nonlinear biobjective optimization is not proficient. In addition, the recent developments [43, 13, 14, 28] do not take full benefits of interval analysis, in particular constraint propagation although it has been used within the B&B-like method PICPA [2]. The difficulty of applying such techniques lies in the exploitation of the dominance relation in the multiobjective case in order to discard *non-optimal* (dominated) solutions of the search space. The method PICPA [2] decomposes the objective space, which eases application of constraint propagation but causes overlapping in the decision space.

We propose in this paper an interval B&B algorithm that integrates constraint propagation through the use of *dominance contractors*. These pruning techniques extend the ideas for multiobjective combinatorial optimization presented in [15, 20] to nonlinear biobjective continuous optimization. This algorithm generalizes the B&B from [43, 14] in which a regular decomposition of the decision space is performed, similar to how it is usually done in the single-objective case. It differs from *inverse* methods [28, 2] in which a decomposition of the objective space masters a decomposition in the decision space.

The paper is organised as follows. Sections 2 and 3 introduce the necessary background on nonlinear biobjective optimization and, respectively, on interval analysis and constraint propagation. Our B&B algorithm with dominance contractors is presented in Section 4. Some experiments validating our proposal are discussed in Section 5. Eventually, the paper is concluded in Section 6.

2. Nonlinear Multiobjective Optimization Problems

In this section we introduce the terminology and notions in use in multiobjective optimization. Though we consider only biobjective problems in this paper, all the definitions given here apply in the general case and are thus expressed for an arbitrary number m of objectives.

Nonlinear MultiObjective Optimization (NLMOO) consists in optimizing several nonlinear conflicting objectives under nonlinear constraints. Such problems arise in many applications, such as engineering design, the need for a compromise being inherent to the decision process (see, e.g., [11, 34]). A NLMOO problem can be written as follows:

$$\left[\begin{array}{ll} \min & f(x) \\ \text{s.t.} & g(x) \leq 0 \\ & h(x) = 0 \end{array}\right] \tag{1}$$

with $x \in \mathbb{R}^n$ the decision variables, $f : \mathbb{R}^n \to \mathbb{R}^m$ the objective functions, $g : \mathbb{R}^n \to \mathbb{R}^p$ the inequality constraints and $h : \mathbb{R}^n \to \mathbb{R}^q$ the equality constraints. The feasible region $\mathcal{X}$ is the set of decision vectors that satisfy all the constraints,

i.e., $\mathcal{X} := \{x \in \mathbb{R}^n : g(x) \leq 0, h(x) = 0\}$. Its image $\mathcal{Y} = f(\mathcal{X})$ in the objective space is called the feasible objective region. In this paper, we consider objective and constraints that are continuously differentiable[1].

Because the objective functions are conflicting, all feasible objective vectors cannot be compared. Still, if one such vector $y \in \mathcal{Y}$ is better according to all the objective functions than another one $y' \in \mathcal{Y}$, y is obviously more desirable than y'. This is formalized by the notion of dominance.

Definition 1 (Dominance relations). *Let y and y' be two vectors in $\mathbb{R}^m$. The following notations are used:*

$$\textit{(i)}\ y < y' \quad \equiv \quad y_i < y'_i \quad \forall i = 1, \ldots, m \qquad (y \text{ strictly dominates } y')$$

$$\textit{(ii)}\ y \lneq y' \quad \equiv \quad y_i \leq y'_i \quad \forall i = 1, \ldots, m, \text{ and } y \neq y' \qquad (y \text{ dominates } y')$$

$$\textit{(iii)}\ y \leq y' \quad \equiv \quad y_i \leq y'_i \quad \forall i = 1, \ldots, m \qquad (y \text{ weakly dominates } y')$$

In a posteriori decision making (i.e., without preferences inducing an aggregation of the objectives), solving problem (1) requires computing its set of Pareto optimal solutions, i.e., optimal trade-offs between the objectives.

Definition 2 (Nondominance, Pareto optimality). *Consider a feasible objective vector $y \in \mathcal{Y}$. It is a* nondominated *(resp.* weakly nondominated*) vector of $\mathcal{Y}$ if there is no other $y' \in \mathcal{Y}$ such that $y' \lneq y$ (resp. $y' < y$). The set of nondominated (resp. weakly nondominated) vectors is denoted $\mathcal{Y}^*$ (resp. $\mathcal{Y}^*_W$).*

A feasible solution $x \in \mathcal{X}$ is Pareto optimal *(resp.* weakly Pareto optimal*) if $f(x)$ is nondominated (resp. weakly nondominated). The set of Pareto optimal (resp. weakly Pareto optimal) solutions is denoted by $\mathcal{X}^*$ (resp. $\mathcal{X}^*_W$).*

As the objectives and constraints are nonlinear (non-convex), locally Pareto optimal solutions may exist.

Definition 3 (Local optimality). *A solution $x \in \mathcal{X}$ is* locally Pareto optimal *if there exists $\delta > 0$ such that x is Pareto optimal in the ball $\mathcal{B}(x, \delta) \cap \mathcal{X}$.*

In the convex case, all locally Pareto optimal solutions are globally Pareto optimal [34, Theorem 2.2.3] and can be found using local approaches, e.g., as a set of Pareto optimal solutions with images well spread upon the nondominated set. Oppositely, the non-convex case requires global search methods like evolutionary algorithms [6], swarm algorithms [42], or interval B&B [43, 14, 28].

Computing all the globally Pareto optimal solutions via interval B&B requires bounding the subproblems issued from the subdivision of the search space. Contrarily to the single-objective case, bounding in multiobjective optimization is not straightforward: the bounds must enclose a whole set of nondominated

[1] Though constraint propagation could apply to evaluable only (blackbox) functions, its effectiveness is reduced in this case.

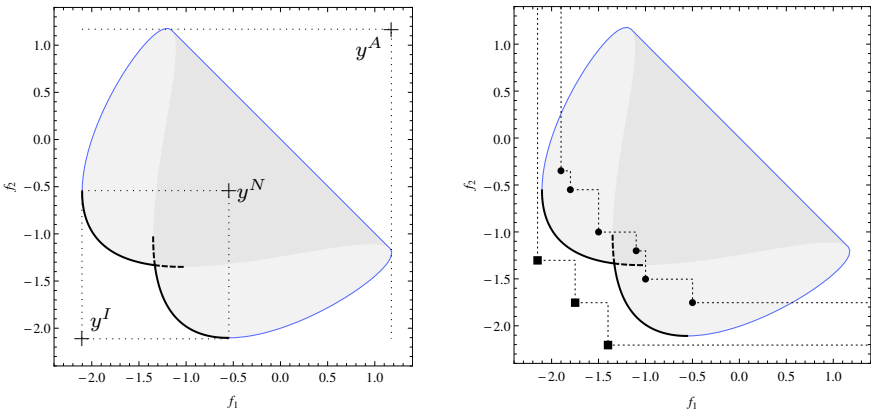


Figure 1: Feasible objective space of a biobjective problem with objective functions $f_1(x) = -(x_1+x_2)^2 - x_1 + x_2$, $f_2(x) = -(x_1+x_2)^2 + x_1 - x_2$ and constraint $g(x) = x_1^2 + 2x_2^2 - 1 \leq 0$. Gray regions correspond to $\mathcal{Y}$. Thick black lines corresponds to $\mathcal{Y}^*$, and thick dashed lines to images of locally Pareto optimal solutions.(left) Ideal, nadir and anti-ideal points, with induced bound sets in dotted lines. (right) An upper bound set represented by six black dots, and a lower bound set by three squared points, the dotted lines deliminating their dominated area.

vectors[2]. Usually, the ideal y^I and nadir y^N (or anti-ideal y^A) points are used to bound, respectively below and above, the nondominated set $\mathcal{Y}^*$:

$$y_i^I = \min_{x \in \mathcal{X}} f_i(x) = \min_{y \in \mathcal{Y}} y_i, \qquad i = 1, \ldots, m \tag{2}$$

$$y_i^N = \max_{x \in \mathcal{X}^*} f_i(x) = \max_{y \in \mathcal{Y}^*} y_i, \qquad i = 1, \ldots, m \tag{3}$$

$$y_i^A = \max_{x \in \mathcal{X}} f_i(x) = \max_{y \in \mathcal{Y}} y_i, \qquad i = 1, \ldots, m \tag{4}$$

As seen on Figure (1), the ideal and nadir bound the nondominated set $\mathcal{Y}^*$: all nondominated points are dominated by the ideal and dominate the nadir, while all feasible objective points dominate the anti-ideal. Those particular points can be "easily" computed in the biobjective case[3] provided solutions of single-objective versions of Problem (1) are known (or can be efficiently obtained). On the other hand, as they are single points, they do not provide good bounds, capturing only poorly the shape of the nondominated set. In order to obtain a more accurate bounding, dominance-free bounding sets have been introduced in [12].

Definition 4 (Dominance-free set). *A set E of vectors in $\mathbb{R}^m$ is* dominance-free *if there is no $y, y' \in E$ such that y dominates y'.*

[2] Assuming this set is actually bounded

[3] Weakly nondominated points increase the difficulty of computing y^N

Intuitively, a dominance free set can serve as a lower (resp. upper) bound if it is "below" (resp. "above") the set of Pareto optimal solutions to the problem. A formal definition follows. The right hand side of Figure 1 depicts one lower and one upper bound set.

Definition 5 (Bound sets). *Consider Problem* (1) *and let* $\mathcal{Y}_L \subset \mathbb{R}^m$ *be a dominance-free set. This set is a* lower bound set *of* $\mathcal{Y}^*$ *if it satisfies:*

$$\mathcal{Y}^* \subseteq \{y : \exists y' \in \mathcal{Y}_L, y \geq y'\}$$

Similarly, let $\mathcal{Y}_U \subset \mathbb{R}^m$ *be a dominance-free set. This set is an* upper bound set *of* $\mathcal{Y}^*$ *if it satisfies:*

$$\mathcal{Y}^* \subseteq \mathbb{R}^m \backslash \{y : \exists y' \in \mathcal{Y}_U, y > y' \}$$

Given this definition, any dominance-free set of feasible objective vectors form a global upper bound set of Problem (1), e.g., the black points in Figure 1. Note also that bound sets can be used to locally bound the Pareto optimal solutions in sub-regions of the search space.

3. Interval analysis

Interval analysis (IA) is a modern branch of numerical analysis born in the 1960's [35]. It replaces computations with real numbers by computations with intervals of real numbers, providing a framework for handling uncertainties and verified computations. It is a powerful tool for dealing reliably with any problems implying real-valued variables such as numerical constraint satisfaction and nonlinear optimization [36, 21, 23, 22].

3.1. Intervals and boxes

An interval $\boldsymbol{x}$ is a closed connected subset of $\mathbb{R}$. It is defined, by a lower and an upper bound $\underline{x}, \overline{x} \in \mathbb{R}$, which gives $\boldsymbol{x} = [\underline{x}, \overline{x}] = \{x \in \mathbb{R} : \underline{x} \leq x \leq \overline{x}\}$. A n-dimensional box $\boldsymbol{x}$ is a vector of n intervals $(\boldsymbol{x}_i)_{1 \leq i \leq n}$, defined by a lower and an upper bound vectors $\underline{x}$ and $\overline{x}$. We can define similarly interval matrices as matrices of intervals: $\boldsymbol{A} = (\boldsymbol{a}_{ij})_{1 \leq i \leq n, 1 \leq j \leq m} = [\underline{A}, \overline{A}]$. A box can be used to enclose tightly any arbitrary subset $\mathcal{U} \subset \mathbb{R}^n$ via the hull operation: $\square\mathcal{U} = [\underline{u}, \overline{u}]$ such that $\forall i \in \{1, \ldots, n\}$, $\underline{u}_i = \inf\{u_i : u \in \mathcal{U}\}$ and $\overline{u}_i = \sup\{u_i : u \in \mathcal{U}\}$.

Given an interval $\boldsymbol{x}$, $\text{mid}(\boldsymbol{x}) := 0.5(\underline{x} + \overline{x})$ is its *center*, $\text{wid}(\boldsymbol{x}) := \overline{x} - \underline{x}$ is its *width*. The width of a box is the maximum of its component-wise widths. Another important operation for a n-dimensional box $\boldsymbol{x}$ is $\text{vol}(\boldsymbol{x}) := \prod_{1 \leq i \leq n} \text{wid}(\boldsymbol{x}_i)$ which defines its volume. Finally, considered as subsets of $\mathbb{R}^n$, boxes accept all set operations.

3.2. Interval arithmetic

Interval arithmetic allows the replacement of real operations by interval ones in numerical computations. It is based on the *containment principle*: the interval resulting from an operation must contain all the possible reals resulting from the corresponding operation applied to any reals from the interval operands. Hence, an arbitrary operation *op* over real numbers $x_1, \ldots, x_n$ is naturally extended to intervals $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ using the hull operation, i.e.,

$$op(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \square\{op(x_1, \ldots, x_n) : x_i \in \boldsymbol{x}_i\}. \tag{5}$$

Simple formulas exist for the most common operators and functions, e.g.,

$$\boldsymbol{x} + \boldsymbol{y} := [\underline{x} + \underline{y}, \overline{x} + \overline{y}], \tag{6}$$

$$\log \boldsymbol{x} := [\log \underline{x}, \log \overline{x}] \text{ if } \underline{x} > 0, \tag{7}$$

$$:= (-\infty, \log \overline{x}] \text{ if } \overline{x} > 0 \geq \underline{x}, \tag{8}$$

$$:= \emptyset \text{ otherwise.} \tag{9}$$

Interval arithmetic hence allows the definition of *interval extensions* $\boldsymbol{f}$ of any (multi-variate) real function f. The natural interval extension simply replaces the different operands of a function by their interval counterparts. When a function is continuous, its natural interval extension is convergent.

Definition 6 (Convergent interval extension). *An interval extension* $\boldsymbol{f}$ *of a function* $f : \mathcal{U} \subseteq \mathbb{R}^n \to \mathbb{R}$ *is* convergent *if for any sequence of boxes* $\boldsymbol{x}^{(k)} \subseteq \mathcal{U}$*,*

$$\lim_{k\to\infty} \operatorname{wid}(\boldsymbol{x}^{(k)}) = 0 \Longrightarrow \lim_{k\to\infty} \operatorname{wid}(\boldsymbol{f}(\boldsymbol{x}^{(k)})) = 0. \tag{10}$$

Consequently, $\boldsymbol{f}(x) = f(x)$ for any degenerated box $[x, x]$. In the following, $\boldsymbol{f}$ will by default designate the natural interval extension of f.

Interval arithmetic inherits many properties of real arithmetic (e.g., associativity, commutativity), but suffers from two specific issues: the *dependency problem* and the *wrapping effect*. Indeed, it does not take into account the dependency between the different occurrences of any variable x in an expression, yielding to an overestimation of the result; for instance, $\boldsymbol{x} - \boldsymbol{x} \neq 0$ and $\boldsymbol{x} \cdot \boldsymbol{x} \neq \boldsymbol{x}^2$ in general. Second, because the result of an operation is an interval, it cannot represent disconnected sets of reals and thus includes many reals which are not the result of an operation from the operands; for example, $\frac{1}{\boldsymbol{x}} = [-\infty, +\infty]$ whenever $0 \in \boldsymbol{x}$, i.e., the *hole* around 0 in the image of the real division operation is lost. In addition, interval arithmetic is in general implemented using floating-point bounds, yielding the necessity for outward rounding which introduces additional overestimation in interval computations.

For these reasons, interval computations are generally included into some branching process which recursively splits operand boxes into smaller ones, making the computation more and more accurate and yielding *pavings*, i.e., a union of non-overlapping, possibly edge-adjacent, boxes.

3.3. Interval constraint propagation

Numerical constraint satisfaction is the problem of finding the set $\mathcal{X}$ of solutions to a constraint system, defined by a set of equalities $h(x) = 0$ and inequalities $g(x) \leq 0$, within a considered box domain $\boldsymbol{x}^{\text{init}} \subseteq \mathbb{R}^n$.

The (interval) constraint programming approach [4] to solving numerical constraint satisfaction problems (NCSPs) follows the *Branch & Prune* scheme and yields a paving of $\mathcal{X}$ down to a prescribed precision : at each iteration, a non-final box of the paving is selected and *contracted* according to the constraints ; if it becomes empty, it is discarded ; if it has reached the prescribed precision, it is declared *final* ; otherwise it is *split* into sub-boxes to be further processed.

Though part of the research in this area considers other topics (e.g., search strategies or applications), most of the work in the interval constraint programming community has concentrated on the definition of efficient *contracting operators* (contractors in short). One reason is that the more efficient the pruning step, the least branching will occur, yielding smaller search trees. It is however a matter of trade-off, since the pruning step can be expensive and may thus turn out counter-productive in some cases.

Cheap and efficient contractors are typically based on some *local consistency* definitions. For example, hull-consistency (aka 2B-consistency) [30] states that each bound of each interval domain $\boldsymbol{x}_i$ must be compatible with the other interval domains with respect to the considered constraints. These definitions give birth to contracting operators that discard boundaries of boxes that are not locally consistent. Typical contractors are HC4-revise [3] which employs operator-wise evaluation and projection in order to enforce hull-consistency, BC3-revise [3] which uses univariate Newton steps to find extremal solutions within an interval domain and enforces box-consistency [5], and MOHC-revise [1] which exploits monotonicity and combines hull and box consistencies. These operators usually consider one constraint at a time and must thus be repeated in sequence, typically following an AC3-like fix-point propagation principle [5]. Because such propagation mechanisms may exhibit slow convergence toward their fix-point, heuristic stopping criteria are often employed, e.g., when the obtained contraction drops below a given (relative) threshold called *improvement factor*.

More global operators comprise: Peeling (or shaving) operators [8, 31], which iteratively discard slices on the boundaries of an interval domain using local consistency based operators on all the constraints ; Constructive interval disjunction (CID) [48, 39] which considers a partition of $\boldsymbol{x}$, propagates other contracting operators (usually HC4-revise) onto each part, and takes the hull of the contracted part as the new domain ; And interval Newton [36] which considers a linear enclosure of the equations of the problem using interval evaluation of their derivatives and applies evaluation/intersection steps until a fix-point, or a maximum number of steps, is reached. The latter has the additional advantage that it can certify the existence of a solution to the equations within a box.

Contractors have proven to be powerful tools for reducing the search space and avoiding large search efforts, allowing to address challenging problems, in particular in control and robotics [21]. They have also been included in interval B&B algorithms to address numerical constrained optimization problems,

Algorithm 1: Interval multiobjective Branch & Bound

Input: multiobjective problem $P = (f, g, h)$; initial box $(\boldsymbol{x}^{\text{init}}, \boldsymbol{y}^{\text{init}})$
Output: nodes $\mathcal{S}_{out}$ containing the Pareto optimal solutions; upper bound set $\mathcal{Y}_U$

1.1 $\mathcal{Y}_L, \mathcal{Y}_U \leftarrow \mathit{InitializeBounds}(\boldsymbol{x}^{\text{init}}, \boldsymbol{y}^{\text{init}}, P)$;
1.2 $\mathcal{S}_{out} \leftarrow \emptyset$;
1.3 $\mathcal{S} \leftarrow \{(\boldsymbol{x}^{\text{init}}, \boldsymbol{y}^{\text{init}}, \mathcal{Y}_L)\}$;
1.4 **while** $\mathcal{S} \neq \emptyset$ **do**
1.5 $\quad (\boldsymbol{x}, \boldsymbol{y}, \mathcal{Y}_L) \leftarrow \mathit{Extract}(\mathcal{S})$;
1.6 $\quad (\boldsymbol{x}, \boldsymbol{y}) \leftarrow \mathit{Prune}(\boldsymbol{x}, \boldsymbol{y}, \mathcal{Y}_L, \mathcal{Y}_U, P)$;
1.7 $\quad$ **if** $(\boldsymbol{x}, \boldsymbol{y}) \neq \emptyset$ **then**
1.8 $\quad\quad \mathcal{Y}_L, \mathcal{Y}_U \leftarrow \mathit{UpdateBounds}(\boldsymbol{x}, \boldsymbol{y}, \mathcal{Y}_L, \mathcal{Y}_U, P)$;
1.9 $\quad\quad$ **if** $\mathit{Final}(\boldsymbol{x}, \boldsymbol{y}, \mathcal{Y}_L, \mathcal{Y}_U)$ **then**
1.10 $\quad\quad\quad \mathcal{S}_{out} \leftarrow \mathcal{S}_{out} \cup \{(\boldsymbol{x}, \boldsymbol{y}, \mathcal{Y}_L)\}$;
1.11 $\quad\quad$ **else**
1.12 $\quad\quad\quad \mathcal{S} \leftarrow \mathcal{S} \cup \mathit{Split}(\boldsymbol{x}, \boldsymbol{y}, \mathcal{Y}_L, \mathcal{Y}_U, P)$;
1.13 $\quad\quad$ **end**
1.14 $\quad$ **end**
1.15 **end**
1.16 **return** $\mathit{CleanUp}(\mathcal{S}_{out}, \mathcal{Y}_U)$

also called (single-objective) nonlinear programs (NLPs) [19, 23, 38, 49]. In this context, they complement other traditional pruning tests, e.g., bounding tests which discard boxes whose best objective value (lower bound) is greater than that of an already computed good solution (upper bound) ; optimality conditions [23, 37, 16] which discard boxes that cannot contain (local) optima according to first or second order optimality criteria ; monotonicity tests [23] which reduce a domain to one of its bound if the objective function is proved to monotonically decrease/increase along the corresponding dimension.

4. A new biobjective Branch & Bound algorithm

Interval B&B methods are well known and most effective methods for solving rigorously and globally NLPs. Although they have been widely developed in the context of single-objective problems, few B&B exist for multiobjective optimization. Moreover, the existing approaches lack some important components, e.g. efficient pruning through constraint propagation.

In this section we propose a generic biobjective B&B algorithm which is a generalization of the single-objective B&B scheme and incorporates constraint programming techniques.

4.1. Generic algorithm

Algorithm 1 introduces a generic multiobjective B&B scheme conforming the tree-search structure used in the literature [26, 27, 28, 14], except that it works

in the product of decision and objective spaces, i.e., on boxes $(\boldsymbol{x}, \boldsymbol{y}) \subset \mathbb{R}^n \times \mathbb{R}^m$ which combine domains for the decision variables x and the objective variables y. This allows considering both sets of variables in all the operations in the algorithm, hence a greater level of generality. Another notable specificity is that Algorithm 1 considers explicitly bound sets (see Definition 5, page 5). The upper bound set $\mathcal{Y}_U$ is global to the whole search tree while a lower bound set $\mathcal{Y}_L$ is associated to each box $(\boldsymbol{x}, \boldsymbol{y})$ in the set $\mathcal{S}$ of search nodes. Finally, Algorithm 1 also explicits a generic pruning step (line 1.6) which allows incorporating any discarding and filtering techniques from the literature as well as new dedicated contractors like the ones we propose in Section 4.4.

In the following, we define the biobjective instance of Algorithm 1 we propose, detailing the implementation of the various functions it involves.

4.2. Inputs and outputs

Algorithm 1 takes as input the considered biobjective problem P defined by the two objective functions $f = (f_1, f_2) : \mathbb{R}^n \to \mathbb{R}^2$, the inequality constraints functions $g : \mathbb{R}^n \to \mathbb{R}^p$ and the equality constraints $h : \mathbb{R}^n \to \mathbb{R}^q$.

It also takes as input the initial search space, defined as a box $(\boldsymbol{x}^{\text{init}}, \boldsymbol{y}^{\text{init}}) \subset \mathbb{R}^{n+2}$ in the decision$\times$objective space. Specifying the decision search space $\boldsymbol{x}^{\text{init}}$ is typical in multiobjective optimization. Specifying the objective search space $\boldsymbol{y}^{\text{init}}$ is more unusual, but it can be useful when one wants to focus the search in a given trade-off area; when no such focus is desired or difficult to obtain (e.g. computing the box whose corners are the ideal and nadir points), $\boldsymbol{y}^{\text{init}}$ can simply be set to $\boldsymbol{f}(\boldsymbol{x}^{\text{init}})$. It must be noted that specifying $\boldsymbol{x}^{\text{init}}$ and/or $\boldsymbol{y}^{\text{init}}$ in fact induces additional constraints in the problem, namely $x \in \boldsymbol{x}^{\text{init}}$ and $y \in \boldsymbol{y}^{\text{init}}$. These bound constraints must be considered in several operations of the algorithm, e.g., optimality conditions checks and pruning.

The algorithm returns a paving $\mathcal{S}_{out}$ in the decision$\times$objective space that covers the weakly Pareto optimal solutions $\mathcal{X}^*_W$ and their corresponding objective values $\mathcal{Y}^*_W$[4]. It also returns the upper bound set $\mathcal{Y}_U$. The global lower bound set $\mathcal{Y}_L$ is not returned explicitly. It can be extracted from $\mathcal{S}_{out}$ in a post-process. The *CleanUp* function post-processes the output data before returning them. In our implementation, it eliminates the boxes in $\mathcal{S}_{out}$ that are dominated by $\mathcal{Y}_U$, which may happen since $\mathcal{Y}_U$ can be updated posterior to the insertion of a box in $\mathcal{S}_{out}$. This process could also be integrated in the *UpdateBounds* operations in order to avoid memory issues when the paving $\mathcal{S}_{out}$ becomes large. Its cost however advocates for its implementation as a post-process.

4.3. Bounding

As explained earlier, our generic B&B uses lower and upper bounding sets. In this section, we discuss how they can be initialized before the algorithm main

[4]Due to interval overestimations, it is in general not possible to distinguish weakly Pareto optimal solutions from Pareto optimal ones

loop and updated during this loop. We also consider their possible exploitations in the other abstract operations of the algorithm.

4.3.1. Initializing the bounds

Function *InitializeBounds* (line 1.1) initializes the bound sets $\mathcal{Y}_L$ and $\mathcal{Y}_U$ on the objective functions $f = (f_1, f_2)$ with respect to the initial domain $(\boldsymbol{x}^{\text{init}}, \boldsymbol{y}^{\text{init}})$ at the beginning of the algorithm. It is typical to initialize the bound sets to singletons, namely the ideal $\mathcal{Y}_L = \{y^I\}$ and the nadir $\mathcal{Y}_U = \{y^N\}$. Since computing these points may be expensive[5], we propose a cheap and safe alternative: $\mathcal{Y}_L = \{\underline{y}^{\text{init}}\}$ and $\mathcal{Y}_U = \{\overline{y}^{\text{init}}\}$. Note that $\underline{y}^{\text{init}} \leq y^I$ and $\overline{y}^{\text{init}} \geq y^N$. The bound sets can be further tighten by means of local optimization and relaxation of the considered problem. We do not study the effectiveness of such techniques in this paper.

4.3.2. Updating the bounds

Function *UpdateBounds* (line 1.8) updates the bound sets $\mathcal{Y}_L$ and $\mathcal{Y}_U$ on the objective functions $f = (f_1, f_2)$ with respect to the currently considered box $(\boldsymbol{x}, \boldsymbol{y})$ at each iteration of the algorithm, provided the box has not been discarded (line 1.7). In this paper, we consider $\mathcal{Y}_L = \{\underline{y}\}$, i.e. the singleton made of the local ideal point at the considered node after the corresponding box has been pruned. In order to update the upper bound set, we propose as in [14] to certify the feasibility of the midpoint $\tilde{x} = \text{mid}(\boldsymbol{x})$ of the decision box of the considered node at each iteration. Certifying the feasibility in the presence of equality constraints requires the use of parametric Hansen-Sengupta [36, 44], with ϵ-Inflation, applied to the system of equation $h(x) = 0$. Such a method is not used in [14]. The parametric Hansen-Sengupta we propose to implement selects the q variables to fix as parameters with a Gauss elimination on the interval matrix $\nabla \boldsymbol{h}(\boldsymbol{x})$ (q is the number of equality constraints, with $q \leq n$, otherwise the Hansen-Sengupta cannot be applied for certifying the solution). Satisfaction of inequality constraints can be checked more easily, using simple interval evaluation. If the feasibility checks succeed, the certified point $\tilde{y} = f(\tilde{x})$ is added to the upper bound set $\mathcal{Y}_U$. Since $\mathcal{Y}_U$ must remain a dominance-free set, all the vectors it contains that are dominated by the newly inserted solution must be removed. This operation takes a time proportional to the size of $\mathcal{Y}_U$ in the worst case. The time complexity of the parametric Hansen-Sengupta is $O(q^2 n)$ for the Gauss elimination; and $O(q^3)$ per iteration of the Hansen-Sengupta (we set a maximum of 15 iterations).

Note that we adopt a "lazy" strategy for eliminating search nodes once the upper bound set is updated, i.e. we do not eliminate directly all the search nodes in $\mathcal{S}$ (and $\mathcal{S}_{out}$) whose lower bound set $\mathcal{Y}_L$ is dominated by the newly inserted element in the upper bound set $\mathcal{Y}_U$. Such nodes are instead discarded in function *Prune*, when they are extracted from $\mathcal{S}$ to be processed. We justify this strategy as the ordering of the search nodes (see in Section 4.5) does not

[5]The ideal requires solving two single-objective NLPs which is itself NP-hard

Algorithm 2: function *Prune* with dominance contractors

Input: node $(\boldsymbol{x}, \boldsymbol{y}, \mathcal{Y}_L)$; upper bound set $\mathcal{Y}_U$; biobjective problem $P = (f, g, h)$

Output: narrowed box $(\boldsymbol{x}, \boldsymbol{y})$

2.1 $(\boldsymbol{x}, \boldsymbol{y}) \leftarrow \mathit{DiscardingTests}(\boldsymbol{x}, \boldsymbol{y}, \mathcal{Y}_L, \mathcal{Y}_U, P)$;

2.2 $\mathcal{S} \leftarrow \mathit{DominanceDecomposition}(\boldsymbol{y}, \mathcal{Y}_U)$;

2.3 $(\boldsymbol{x}', \boldsymbol{y}') \leftarrow \square \bigcup_{\boldsymbol{y}'' \in \mathcal{S}} \mathit{Contract}((\boldsymbol{x}, \boldsymbol{y}''), P)$;

2.4 **return** $(\boldsymbol{x}', \boldsymbol{y}')$

allow to efficiently find the nodes that are dominated by a given element in the upper bound set.

4.3.3. Exploiting the bounds

Consider the box $(\boldsymbol{x}, \boldsymbol{y})$ extracted at a given iteration of the main loop of Algorithm 1, its associated lower bound set $\mathcal{Y}_L$, and the global upper bound set $\mathcal{Y}_U$. A first remark is that since the bound sets are composed of points in the objective space, they only impact directly the objective components $\boldsymbol{y}$ of the considered box. A second remark is that since we have chosen $\mathcal{Y}_L = \{\underline{y}\}$, the lower bound has no impact at all on the considered box.

Points $\hat{y} \in \mathcal{Y}_U$ that dominate the local nadir $\overline{y}$ of the considered box allow removing part of $\boldsymbol{y}$. This removal can take two different forms : either $\boldsymbol{y}$ is split so as to isolate the dominated part, yielding several boxes to be further processed in future iterations ; or else, the dominated part of $\boldsymbol{y}$ is pruned and the impact of this reduction is propagated towards the decision box $\boldsymbol{x}$, yielding a narrowed box $(\boldsymbol{x}', \boldsymbol{y}')$. The upper bound set $\mathcal{Y}_U$ can thus be exploited during the pruning and the splitting operations of Algorithm 1. We detail the exploitation we propose in the following sections.

4.4. Pruning

Function *Prune* aims at discarding parts of the current box $(\boldsymbol{x}, \boldsymbol{y})$ that do not satisfy the constraints or cannot contain (locally) optimal solutions. The implementation we propose combines state of the art discarding tests and constraint propagation using a new contractor based on the dominance relations. It is detailed in Algorithm 2.

This algorithm takes as input a box $(\boldsymbol{x}, \boldsymbol{y})$ and its associated lower bound set $\mathcal{Y}_L$, as well as the global upper bound set $\mathcal{Y}_U$ and the considered problem P. It returns a narrowed box $(\boldsymbol{x}', \boldsymbol{y}')$. This box is obtained applying first discarding tests (line 2.1), then decomposing the objective dimensions of the resulting box using the upper bound set (line 2.2) and then pruning the resulting sub-boxes and taking the hull of all the pruned boxes as a result (line 2.3). This pruning process follows the principles of Constructive Interval Disjunction [48, 39] guided by the dominance relation with respect to the upper bound set. Note that function *Prune* could be interrupted when it becomes obvious no more process is required, e.g., when the whole box is discarded at line 2.1 or when the resulting

narrowed box becomes equal to the input one after a *Contract* at line 2.3. We detail our implementation of the operations composing this function in the following sections.

4.4.1. Discarding tests

Discarding tests are in general cheap operations that allow discarding a whole box at once. For this reason, we propose to apply them first in our implementation of function *Prune*.

The discarding tests we propose to use are:

1. The dominance test which seeks in $\mathcal{Y}_U$ one vector $\hat{y}$ dominating $\mathcal{Y}_L = \{\underline{y}\}$, thus dominating the whole box $\boldsymbol{y}$ (time complexity logarithmic in the size of $\mathcal{Y}_U$ provided it is sorted);
2. The simple monotonicity tests from [14] (time complexity linear in the number of decision variables), which possibly allows some pruning;
3. The generalized monotonicity test from [14] (time complexity quadratic in the number of decision variables);
4. And the three first order tests from [16] (time complexities linear or cubic in the number of decision variables).

These tests are applied in the order above, i.e., from the cheapest to the most expensive, interrupting their application whenever one succeeds.

4.4.2. Dominance decomposition

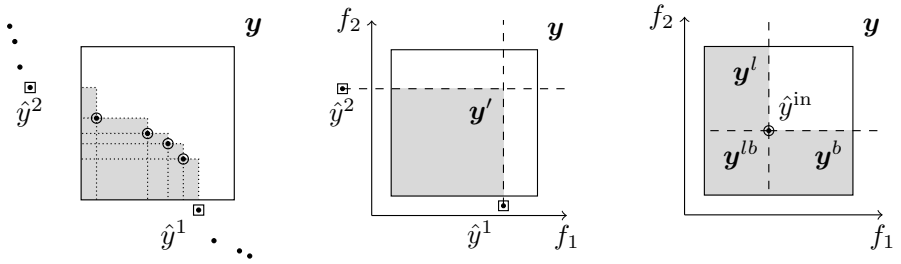
Function *DominanceDecomposition* exploits the upper bound set $\mathcal{Y}_U$ to decompose an objective box $\boldsymbol{y}$ into several sub-boxes.

In single-objective optimization, the interval $\boldsymbol{y}$ would be intersected with $[-\infty, y_U]$, where y_U is the (scalar) upper bound value. In the biobjective case as it has been seen in the discrete case [15, 20], the upper bound set induces a non-dominance relation which can be formalized as follows:

$$\bigwedge_{\hat{y} \in \mathcal{Y}_U} \big(y_1 \leq \hat{y}_1 \vee y_2 \leq \hat{y}_2\big). \tag{11}$$

Then only the vectors in $\mathcal{Y}_U$ such that $\hat{y} < \overline{y}$ are of interest to contract a box using constraint (11). Among them, vectors such that $\hat{y} < \underline{y}$ allow discarding the whole box but are already exploited in Function *DiscardingTests* (see above) and are thus disregarded here.

Vectors such that $\hat{y}_i < \underline{y}_i$ and $\hat{y}_j \in \boldsymbol{y}_j$, $i \neq j$, and in particular the *tightest vectors* as defined in [20] and depicted in square on Figure 2(a), allow contracting the objective box $\boldsymbol{y}$ from the outside. Indeed, for such vectors, one of the inequalities in the corresponding disjunction in (11) is not satisfied and thus the disjunction is reduced to the other inequality. Figure 2(b) depicts as $\boldsymbol{y}'$ the portion of $\boldsymbol{y}$ satisfying (11) given two such vectors.



(a) Selection from $\mathcal{Y}_U$ (b) Decomposition by $\hat{y}^1, \hat{y}^2$ (c) Decomposition by $\hat{y}^{\text{in}}$

Figure 2: Cutting $\boldsymbol{y}$ in the biobjective case from $\mathcal{Y}_U$.

Eventually, vectors such that $\underline{y} < \hat{y}$ (i.e., $\hat{y} \in \boldsymbol{y}$), like those circled in Figure 2(a), impose to consider the corresponding disjunction of inequalities in (11). In that case, one can decompose the box $\boldsymbol{y}$ into three nondominated sub-boxes ($\boldsymbol{y}^{lb}$, $\boldsymbol{y}^{l}$, $\boldsymbol{y}^{b}$) as shown on Figure 2(c) .

By exploiting the ordering of the upper bound set $\mathcal{Y}_U$, finding the interesting points can be done in logarithmic time in the size of $\mathcal{Y}_U$ (in order to identify the tightest outer points) plus an additional term linear in the number of interesting points (which are comprised between the outer points in $\mathcal{Y}_U$). The complexity of the overall Algorithm 2 depends on the number of sub-boxes produced by *DominanceDecomposition*: the finer is the decomposition, the better is the expected pruning (see Figure 2(a) where the gray part corresponds to the finest decomposition into 15 sub-boxes exploiting all the interesting points in this example) but also the higher is the computational cost. Because the number of interesting points can be very large[6], we propose to restrict the decomposition to at most three points from $\mathcal{Y}_U$, namely:

- $\hat{y}^1$ the tightest outer point with respect to objective f_1;
- $\hat{y}^2$ the tightest outer point with respect to objective f_2;
- $\hat{y}^{\text{in}}$ the inner point inducing the largest dominated volume within $\boldsymbol{y}$;

and we suggest three variants of the function *DominanceDecomposition*:

Dominance Peeler This variant uses only $\hat{y}^1$ and $\hat{y}^2$ and reduces $\boldsymbol{y}$ to a single box $\boldsymbol{y}'$ as shown in Figure 2(b);

Dominance Divider This variant uses the three points $\hat{y}^1$, $\hat{y}^2$ and $\hat{y}^{\text{in}}$, reduces $\boldsymbol{y}$ with the first two just like *Dominance Peeler* and then decomposes it into two sub-boxes $\boldsymbol{y}^l \cup \boldsymbol{y}^{lb}$ and $\boldsymbol{y}^b$ as shown in Figure 2(c);

[6]The size of $\mathcal{Y}_U$ is not bounded and all its points may allow decomposing a given box.

Dominance Full Divider This variant is similar to *Dominance Divider* except it decomposes $\boldsymbol{y}$ into three sub-boxes $\boldsymbol{y}^l$, $\boldsymbol{y}^{lb}$ and $\boldsymbol{y}^b$ as shown in Figure 2(c).

Note that some of the interesting points considered in each variant may not exist in $\mathcal{Y}_U$, in which case they are ignored. In order to avoid unnecessary work on poor decompositions, we also propose to use $\hat{y}^{\text{in}}$ only when the volume it dominates within $\boldsymbol{y}$ exceeds a given ratio:

$$\text{vol}([\hat{y}^{\text{in}}, \overline{y}]) \geq \rho \cdot \text{vol}(\boldsymbol{y}), \tag{12}$$

where $0 \leq \rho < 1$ is a parameter. When $\hat{y}^{\text{in}}$ cannot be used (or does not exist), all the proposed variants work the same. When $\hat{y}^1$ and $\hat{y}^2$ also do not exist, no decomposition happen and $y'' = y$. Still, the contraction in Algorithm 2 operates once, since the other constraints in the problem may still allow some reduction of the considered box.

4.4.3. Contraction

Given a box $(\boldsymbol{x}, \boldsymbol{y}'')$ resulting from the dominance decomposition, the following numerical constraint satisfaction problem (NCSP) can be built from the biobjective problem P:

$$\left[\begin{array}{c} f(x) = y,\ g(x) \leq 0,\ h(x) = 0 \\ x \in \boldsymbol{x}, y \in \boldsymbol{y}''. \end{array} \right]. \tag{13}$$

Applying classical contractors with respect to this NCSP on a box $(\boldsymbol{x}, \boldsymbol{y}'')$ allows pruning its decision domains *and* its objective domains. The constraint $f(x) = y$ indeed establishes the link between decision and objective variables and allows propagating reductions from one space to the other. Indeed, if $f(\boldsymbol{x}) \subsetneq \boldsymbol{y}''$, this constraint allows narrowing the objective domain $\boldsymbol{y}''$. Conversely, if $f(\boldsymbol{x}) \supsetneq \boldsymbol{y}''$, it allows narrowing $\boldsymbol{x}$.

Following the principle of CID, function *Contract* (line 2.3) applies on each sub-box resulting from the dominance decomposition an AC3-like fix-point algorithm using HC4 and BC3 contractors, and takes the hull of the resulting contractions. This fix-point loop may be interrupted considering an improvement factor.

4.4.4. Alternative pruning techniques

Other, potentially complementary, pruning techniques are proposed in [14, 27]: in [14], a contractor similar to BC3-revise extracts from the "middle" of the domain of a variable the values yielding objective vectors dominated by a selected point $\hat{y} \in \mathcal{Y}_U$. It prunes only one variable domain at a time and is thus less global than the contractor we propose. In [27], an interval Newton method exploiting the first order optimality conditions of multiobjective problems is proposed . This method can be used directly for unconstrained problems. However for constrained problems, and in particular those containing inequalities, one has to take care of changes of constraints activity as in [33], otherwise the interval Newton cannot converge. Since we consider general constrained biobjective problems, we don't use this specific contractor.

4.5. Search Strategy

The search strategy in Algorithm 1 is implemented within the three functions *Extract* (line 1.5), which selects the next node to be processed, *Final* (line 1.9), which determines if a node must not be further processed, and *Split* (line 1.12), which decomposes a non-final node into sub-nodes to be further processed. Since we do not aim at studying the effect of different search strategies in this paper[7], we propose to use a standard setting of the search strategy that has been proved to be experimentally stable for the different problems used in the numerical experiment in Section 5.

The three components of our search strategy are based on the following criteria, Ordering Criterion (OC), Termination Criterion (TC) and Splitting Criterion (SC):

OC $\frac{\underline{y_1}-y_1^{\text{init}}}{\text{wid}(\boldsymbol{y}_1^{\text{init}})} + \frac{\underline{y_2}-y_2^{\text{init}}}{\text{wid}(\boldsymbol{y}_2^{\text{init}})}$.

TC $\text{wid}(\boldsymbol{x}_i) \leq \epsilon_{x_i}$.

SC Max Domain, i.e. the variable x_i whose $\text{wid}(\boldsymbol{x}_i)$ is the largest.

4.5.1. Ordering Criterion

Nodes in $\mathcal{S}$ are ordered by increasing OC, which is defined as the normalized (with respect to $\boldsymbol{y}^{\text{init}}$) sum of objectives. The search hence focuses first on the regions of the Pareto front minimizing OC. Technically, $\mathcal{S}$ is implemented as a binary search tree, with a constant time complexity for extracting a node (as the first one is extracted) and a $O(\log(|\mathcal{S}|))$ time complexity for insertion of nodes in the set.

This criterion is a particular case of the weighted sum criteria from [46], in which $\boldsymbol{y}^{\text{init}}$ is used for the normalization instead of the nadir and the ideal points (hence avoiding their computations[8]). Other criteria from [46, 14] consider other weights including adaptive ones that change over the iterations. This latter criterion enables the search to focus more homogeneously along the Pareto front, but requires reordering $\mathcal{S}$ whenever the weights are changed.

4.5.2. Termination Criterion

Setting a search node final in Algorithm 1 is problem dependent. For example, one may wish to stop the B&B process once a global precision on the upper bound set, with respect to the merged lower bound sets, is obtained. Criteria from [14] considers a precision on the decision variables or a relative precision on the objectives. The criterion we are using here can be seen as a safeguard for avoiding infinite decomposition of the search space when any other termination criteria is considered.

[7] A comparison of some search strategies is presented in [32].

[8] If those points can be computed efficiently, they can be used to initialize $\boldsymbol{y}^{\text{init}}$

4.5.3. Splitting Criterion

This criterion is used to determine which variable domain is halved. The procedure *Split* hence splits a node in two sub-nodes along the variable with largest width. This is a common and robust criterion in many interval B&B algorithms, and the only one to our knowledge used for multiobjective optimization. This criterion is moreover balanced, which is necessary for proving the theoretical convergence of the B&B (see below).

Another splitting criterion used in single-objective optimization is the smear-based criterion, see e.g. [47]. It is in general an efficient criterion, but theoretically unbalanced. Finally, it is interesting to note that since objective variables and associated domains are considered in Algorithm 1, these domains could also be selected for splitting, yielding to mixed method in-between direct B&B [43, 14] and inverse B&B [28]. Since this would require a study of its own, we leave this possibility as a future direction of research.

4.5.4. Convergence

The convergence proof of the interval B&B from [14] considers an equivalent termination criterion as TC with $\epsilon_x = 0$ and a slight modification[9] of the ordering criterion similar to OC. Hence, the nodes in $\mathcal{S}$ whose decision domains reduce asymptotically to points (due to the 0 precision) can be proved to converge to the set $\mathcal{X}^*_W$ of weakly Pareto optimal solutions. This proof however supposes that the upper bound set $\mathcal{Y}_U$ can be updated in order to cover the whole set $\mathcal{Y}^*$, which requires to ensure that the feasibility of the decisions in $\mathcal{X}^*$ can be numerically guaranteed. This requires additional regularity assumptions on the problems (e.g. constraint qualifications on solutions in $\mathcal{X}^*$), in particular for dealing with equality constraints not considered in [14]. If we consider filtering the nodes in $\mathcal{S}$ each time the upper bound set $\mathcal{Y}_U$ is updated (i.e. not using the lazy strategy we propose to use in practice), our algorithm has similar convergence results as in [14], since adding a constraint propagation process with dominance contractors does not permit to discard nodes containing weakly Pareto optimal solutions.

5. Numerical Experiment

We have implemented our interval B&B algorithm with dominance contractor in C++ using the RealPaver [18] API, which features routines for constraint propagation and contractors, and using the Gaol [17] library for interval arithmetic. These experiments have been run on a computer under Linux Ubuntu version 14.10 64-bit, with processor Intel i7-4702MQ 2.20GHz and 8Gb of RAM.

The implementation follows the description given in Section 4. The portion ρ used in the *dominance divider* and *dominance full divider* is set to 0.125, which

[9] Every K (a positive integer) iterations of the algorithm, the extracted node is the one with the decision box of largest width.

Table 1: Problem characteristics

Problem	n	p	q	ϵ_x
Non-scalable				
Binh [7]	2	2	0	0.00125
KIM [24]	2	0	0	0.00125
NBI [9]	5	1	2	0.00125
SR [52]	7	11	0	0.00125
OSY [40]	6	6	0	0.00125
WD [50]	4	4	0	0.00125
TAN [7]	2	2	0	0.00125
Scalable				
CF3 [51]	$\{2, 3\}$	1	0	0.0125
CTP1 [10]	$\{3, 5\}$	2	0	0.00125
CTP2 [10]	$\{3, 5\}$	1	0	0.00125
CTP6 [10]	$\{3, 4\}$	1	0	0.005
CTP7 [10]	$\{4, 5\}$	1	0	0.0025
MOP [45]	$\{7, 10, 13\}$	0	0	0.005

Note: n = nb variables, p = nb inequalities, q = nb equalities.

proved to be an appropriate default setting for our benchmark. The improvement factor used to interrupt long-running fix-point constraint propagation is set to 0.75 and also considers the prescribed precision on the decision variables given by the termination criterion in order to avoid unnecessary pruning of small enough variable domains.

We consider a benchmark of 13 problems from the literature, with various characteristics detailed in Table 1. These problems are fully described in Appendix A. The last 6 problems are scalable and we have thus considered various dimensions to study the scalability of the methods. Each problem is given a specific precision ϵ_x on the decision variables for the termination criterion TC. This precision has been fixed so as to allow precise enough outputs in reasonable computational times.

For each problem, an initial decision box $\boldsymbol{x}^{\text{init}}$ is given as bound constraints, see Appendix A. The initial objective box is set to $\boldsymbol{y}^{\text{init}} = \boldsymbol{f}(\boldsymbol{x}^{\text{init}})$, which is not unbounded on the tested problems. Hence, the objective vector $\overline{y}^{\text{init}}$ can be used as a reference point for computing hypervolumes [53]. Therefore, for each tested implementation, we measure both the CPU time (an efficiency indicator) and the normalized, with respect to the width of $\boldsymbol{y}^{\text{init}}$, difference Δ_H between the hypervolume of the global lower bound set (obtained by combining and filtering by dominance the lower bounds $\mathcal{Y}_L$ of all nodes in $\mathcal{S}$) and the hypervolume of the global upper bound set $\mathcal{Y}_U$ (a quality indicator). This difference measures the *distance* between the lower bound set and the upper bound set: the smaller the difference, the better the enclosure of the weakly Pareto optimal solutions. We also provide the number of output boxes, the total number of nodes treated (number of iterations), the size of the final upper bound set $\mathcal{Y}_U$ and the maximum number of nodes stored in $\mathcal{S}$ at any time of

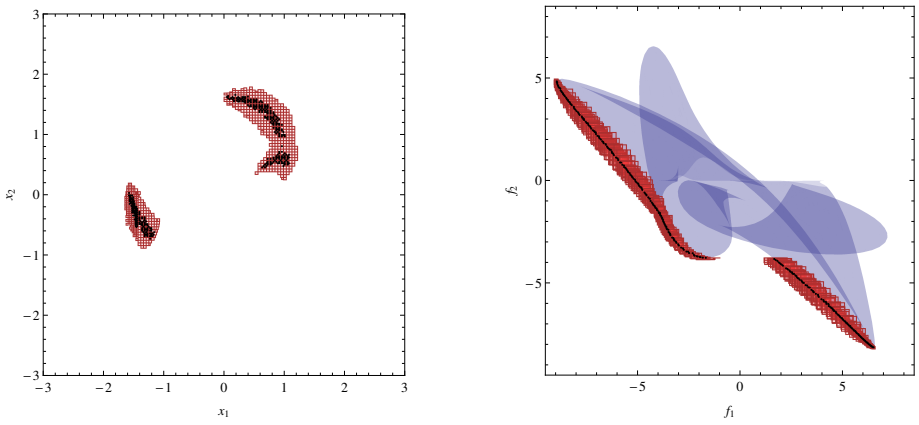


Figure 3: Illustration on the problem KIM with $\epsilon_x = 0.05$. On the left, the produced decomposition in the decision space, on the right in the objective space. Black dots are points from the computed upper bound set $\mathcal{Y}_U$. Red boxes are the obtained paving $\mathcal{S}_{out}$. The blue shape on the right represents the image of the initial decision box.

the algorithm. In addition, we provide for each statistic a "ratio to the best method" allowing to quantitatively rank all methods on a problem according to each indicator. This eases the appreciation of the efficiency/quality trade-off for each method. For all the methods and problems, we have used a timeout of 3600 seconds beyond which no statistics are reported.

In order to assess the relative merits of the various contractors we have proposed with respect to the literature, we compare the following implementations of our interval B&B:

basic the B&B described in Section 4 without dominance contractor, i.e., function *DominanceDecomposition* returns the input box as it is. Pruning by constraint propagation is performed on the inequality and equality constraints only, and monotonicity and first-order rejection tests are used;

basic+FT basic B&B using the pruning/splitting method from Fernández and Tóth [14] discussed in section 4.4.4;

basic+peel basic B&B using dominance peeler (i.e., only outer points $\hat{y}^1$ and $\hat{y}^2$ from $\mathcal{Y}_U$);

basic+div basic B&B using dominance divider (i.e., both outer points $\hat{y}^1$, $\hat{y}^2$, and inner point $\hat{y}^{\text{in}}$ from $\mathcal{Y}_U$, decomposition in at most 2 sub-boxes);

basic+full basic B&B using dominance full-divider (i.e., like dominance divider but with full decomposition, at most 3 sub-boxes).

A paving obtained with basic+full is shown on Figure 3.

We also compare all our variants with our implementation[10] of the B&B from [14] in two different versions:

FT implementation of [14], i.e., similar to basic+FT but without constraint propagation nor first-order optimality discarding test;

FT+FO same as FT but using the first-order optimality discarding test, i.e., similar to basic+FT but without constraint propagation.

FT+FO and basic+FT are equivalent on unconstrained problems, with the exception that in FT+FO, nodes in $\mathcal{S}$ are filtered by dominance each time $\mathcal{Y}_U$ is successfully updated (following the implementation from [14]). Note also that equality constraints are not considered in [14], thus FT and FT+FO are not applied on Problem NBI.

In the following we discuss the experimental results, first on the non-scalable problems, then on the scalable ones.

5.1. Non-scalable problems

Results on non-scalable problems are detailed in Table 3. We can see that overall, dominance contractors, and in particular the method *basic+peel*, are giving the best results with the exception of problem KIM. This bound-constrained problem has only two variables but the objectives contain many multi-occurrences of them, making overestimations of their interval valuation large. Propagating on the objectives turns out to be too computationally expensive with respect to the slight gain in terms of number of explored nodes.

We can also note that for higher dimensional and highly constrained problems WB, OSY and SR, constraint propagation performs well compared to FT. Although on problem WB and OSY, the use of first-order rejection test with FT allows an efficient pruning (FT+FO competes with some of the proposed methods, while FT may reach the timeout[11]), this is not the case on SR and in general dominance contractors help the solving process yielding better timings, precision and less explored nodes. Comparatively, FT performs poorly on WB, OSY and SR. This is not surprising since the constraints of those problems can be efficiently treated by first-order test and constraint propagation.

As the problem NBI contains equality constraints, it can not be treated by FT or FT+FO, and only the proposed methods are compared. It appears that the pruning technique from [14] here greatly deteriorates the search in terms of timing. Dominance contractors based on peeling appear to be the most efficient approach.

On the lower dimensional and constrained problems Binh and TAN, all methods compete well with each other, with a slight advantage for the proposed tech-

[10] No open-source implementation of this method is available at the moment, and no other method can deal with the same spectrum of problems in a direct way.

[11] With FT on OSY: the experiment encounters a memory issue terminating before the timeout. The timeout can be effectively reached using a lower precision $\epsilon_x = 0.0125$

niques which give better timings and precisions and explore the decision space more efficiently.

These experiments overall validate the efficiency of constraint propagation and dominance contractors. However, we can see that in general peeling is enough to obtain good results and that the stronger *divider* and *full-divider*, although they bring more accurate results (Δ_H) and eliminate more search nodes ($\#Nodes$), are not worth the additional computational efforts.

5.2. Scalable problems

The results for scalable problems are presented in Tables 4 and 5.

The problem CF3 appears to be very difficult as it is poorly handled by all of the tested approaches. This can be seen as the number of final nodes represent a large portion of the decomposed space. This is due to the nested and intricate trigonometric functions that appear in both objectives and constraint making interval evaluations imprecise. In addition, we have observed that a large part of the decision space has an image close to the Pareto front. Hence we could not solve CF3 for dimensions $n > 3$ before the timeout with any method. Still, the results for dimensions $n = 2$ and $n = 3$ are interesting. The methods *FT*, *FT+FO* and *basic + FT* perform worse than the other methods on CF3 for both dimensions $n = 2$ and $n = 3$ (about 4 times worse on all indicators, i.e., both in efficiency and quality, for $n = 3$ compared to the best obtained results). The use of dominance contractors seems to ease the scaling from $n = 2$ to $n = 3$, since *basic* alone which is competitive for $n = 2$ is drastically degraded for $n = 3$. It is also worth mentioning that the divider strategy is the best on this problem. This could partly be explained by the large number of feasible points in $\mathcal{Y}_U$, whose discovery seems to be facilitated by dominance contractors which in turn benefit from them in future pruning steps, yielding a mutual reinforcement mechanism.

On the CTP test suite, no method clearly dominates the others, though the proposed methods with dominance contractors globally provide the most stable, and relatively good, results across all these problems. On CTP1 for instance, the dominance contractors allow to discard infeasible or suboptimal nodes efficiently enough at dimension $n = 3$ but become too expensive at dimension $n = 5$, though they still outperform FT and FT+FO. A similar observation holds for CTP2, but in this case FT and FT+FO are the most efficient methods at the largest dimension $n = 5$ (though faster by less than 21%). Again, the quality achieved using stronger contractors comes at a too high computational price in this case. On CTP6, appart on the number of explored nodes, all methods are quite competitive. We observe, as expected, that dominance contractors help improving the exploration by reducing the number of treated and output nodes. Eventually, CTP7 requires the use of pruning on objectives, as can be seen by the poor performances of *basic*. This is understandable as the Pareto front is made of several disconnected components in this problem. Constraint propagation is hence also important for performing well on the higher dimensional instance. Again, the method *basic+peel* appears to overall perform best.

Table 2: Average and max ratios obtained for all tested problems on a subset of measures.

Method	CPU			Δ_H			# Nodes			# TO
	avg.	max	# b	avg.	max	# b	avg.	max	# b	
FT	1.52	19.38	3	1.42	4.37	0	1.72	9.02	0	4
FT+FO	1.92	19.26	1	1.58	5.79	0	1.84	9.02	0	1
basic	1.63	50.70	3	1.39	6.38	2	1.85	74.08	0	1
basic+FT	1.85	11.41	0	1.21	4.37	0	1.30	9.02	0	1
basic+peel	**1.14**	7.86	**9**	1.04	1.30	5	1.11	11.00	6	1
basic+div	1.18	**1.91**	4	1.04	1.75	**12**	**1.00**	**1.02**	**14**	**0**
basic+full	1.24	2.42	1	**1.01**	**1.14**	11	**1.00**	1.03	**14**	**0**

On the last problem MOP, which is bound-constrained, it is interesting to see that solely divider and full-divider are scaling well. This problem has simple objectives, which can be efficiently handled by the stronger dominance contractors. They are indeed helping reducing drastically the number of explored nodes. It is also worth noting *divider* and *full-divider* behave totally identically at all dimensions, thus the result is in favor of *divider* due to its cheapest computational cost.

5.3. Summary

We summarize our experimental results of the previous sections in Table 2 with respect to the ratios of CPU times and #Nodes (efficiency), and Δ_H (quality). The table provides the harmonic average and maximum ratio for each indicator, as well as the number of times #b a method has obtained ratio 1.0, i.e., it has been the best on the corresponding measure. The number of timeout *# TO* attained by each method is also reported.

This underlines the global performance and robustness of our proposed algorithm with *dominance contractors*, and in particular with the use of *divider* and *full-divider*, in terms of both efficiency and quality. Those contractors, based on traditional constraint programming techniques, outperform overall the pruning technique from [14].

Eventually, from the previous results and analyzing the problems specifically, we can draw the following conclusions:

1. complex objective expressions (i.e. with numerous occurrences of each variable) cannot be efficiently handled by the dominance contractors. This is also true for complex constraints expressions and constraint propagation in general.

2. highly constrained problems (i.e., OSY, SR and WD) are solved efficiently with the usage of constraint propagation and the proposed dominance contractors (dominance peeler and dominance divider). They generalize the classical pruning step of single-objective interval B&B and perform overall better than using specific pruning techniques alone, such as the one from [14].

3. *dominance divider*, although having stable performance from one problem to the other, is not often performing well compared to the single use of

peeling. But for some problems with simple objective expressions and a high number of variables, they can help a lot in reducing the exploration cost. Nevertheless, *full-divider* has never been worth the effort. It advocates for a parsimonious decomposition of the objective boxes with dominance contractors.

6. Conclusion

The proposed interval Branch & Bound with dominance contractors offers a generic framework for applying constraint propagation in a biobjective optimization context. It allows exploiting efficiently the upper bound set collected during the search, similarly to what is done in the single-objective case. The implementation we propose clearly helps the global solving of nonlinear biobjective optimization by discarding efficiently parts of the search space not containing Pareto optimal solutions. As it is the case of many interval-based techniques, the performance of dominance contractors depends on the complexity of the objective functions and constraints expressions, in particular in terms of number of multi-occurrences of each variable (dependency problem).

The directions for future research are numerous. First we could explore the use of other contractors, e.g., based on interval linearization that may help reducing the influence of complex constraints/objectives expressions, like the Quad filtering proposed for quadratic constraints in [29]. Linearization can also be used for computing (using parametric simplex algorithm) better lower bound sets than the ones used in this paper based on local approximated ideal points. This in turn would yield the exploration of different search strategies.

Improving the update of the upper bound set via local search is also a promising search direction. In particular, we would like to embed the certified continuation method ParCont [33] inside our interval B&B so as to quicken the verified computations of enclosures of Pareto optimal solutions, and avoid exploring already discovered regions of the search space.

The fact the interval B&B algorithm we propose uses explicit domains for the objective functions makes it possible to develop an inverse approach similar to [28] (see [32] for preliminary experiments), or even hybrid approaches which split intervals in both the decision and objective spaces, combining the strengths of both approaches.

Eventually, the extension of the dominance contractor to more than 2 objectives can be done without theoretical difficulties. More objectives means more ways to divide objective boxes into nondominated parts given a point from the upper bound set, see [32]. The difficulty is then to maintain and search elements in the upper bound set efficiently. The techniques developed in [41, 25] could help solve this issue.

Acknowledgment

The authors would like to thanks Brice Chevalier, whose earlier work on interval B&B in nonlinear multiobjective optimization during his master thesis

has been the foundation of the developments proposed here. The first author is also grateful to the Portuguese Foundation for Science and Technology. for having partially granted this work through the project PROCURE (Probabilistic Constraints for Uncertainty Reasoning in Science and Engineering Applications), ref. PTDC/EEI-CTP/1403/2012.

References

[1] I. Araya, G. Trombettoni, and B. Neveu. Exploiting monotonicity in interval constraint propagation. In *AAAI*, 2010.

[2] V. Barichard and J. Hao. A population and interval constraint propagation algorithm. In *LNCS*, pages 88–101. Springer, 2003.

[3] F. Benhamou, F. Goualard, L. Granvilliers, and J-F. Puget. Revising hull and box consistency. In *International Conference on Logic Programming*, pages 230–244. MIT press, 1999.

[4] F. Benhamou and L. Granvilliers. Chapter 16 - continuous and interval constraints. In Peter van Beek Francesca Rossi and Toby Walsh, editors, *Handbook of Constraint Programming*, volume 2, pages 571 – 603. Elsevier, 2006.

[5] F. Benhamou, D. McAllister, and P. Van Hentenryck. CLP(Intervals) Revisited. In *International Symposium on Logic Programming*, pages 124–138, 1994.

[6] C. A. C. Coello, G. B. Lamont, and D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Springer-Verlag, 2006.

[7] C. A. C. Coello, G. B. Lamont, and D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[8] H. Collavizza, F. Delobel, and M. Rueher. Comparing partial consistencies. *Reliable Computing*, 5(3):213–228, 1999.

[9] I. Das and J. Dennis. Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems. *SIAM Journal on Optimization*, 8(3):631–657, 1998.

[10] K. Deb, A. Pratap, and T. Meyarivan. Constrained test problems for multi-objective evolutionary optimization. In E. Zitzler, L. Thiele, K. Deb, C. A. Coello Coello, and D. Corne, editors, *Evolutionary Multi-Criterion Optimization*, volume 1993 of *Lecture Notes in Computer Science*, pages 284–298. Springer Berlin Heidelberg, 2001.

[11] M. Ehrgott. *Multicriteria Optimization (2. ed.)*. Springer, 2005.

Table 3: Results for non-scalable problems.

	Methods	CPU		Δ_H		$\lvert\mathcal{S}_{out}\rvert$		# Nodes		$\lvert\mathcal{Y}_U\rvert$	# Store	
Binh	FT	1.35	*1.22*	1.32E-4	*1.07*	9 016	*1.06*	37 820	*1.05*	27 576	575	*1.07*
	FT+FO	2.05	*1.85*	1.32E-4	*1.07*	9 016	*1.06*	37 816	*1.05*	27 576	575	*1.07*
	basic	**1.11**	*1.00*	1.26E-4	*1.02*	8 573	*1.01*	36 102	*1.01*	31 976	542	*1.01*
	basic+FT	1.38	*1.25*	1.26E-4	*1.02*	8 573	*1.01*	36 102	*1.01*	31 976	542	*1.01*
	basic+peel	1.25	*1.13*	1.26E-4	*1.02*	8 573	*1.01*	36 098	*1.01*	31 976	540	*1.00*
	basic+div	2.02	*1.82*	**1.24E-4**	*1.00*	**8 516**	*1.00*	**35 866**	*1.00*	33 197	**539**	*1.00*
	basic+full	1.57	*1.41*	1.24E-4	*1.00*	8 648	*1.02*	36 420	*1.02*	32 426	**539**	*1.00*
KIM	FT	38.31	*1.28*	2.12E-4	*1.01*	64 478	*1.08*	304 197	*1.02*	80 623	**4 390**	*1.00*
	FT+FO	42.67	*1.43*	2.12E-4	*1.01*	64 478	*1.08*	304 197	*1.02*	80 623	**4 390**	*1.00*
	basic	**29.91**	*1.00*	**2.11E-4**	*1.00*	62 283	*1.04*	315 660	*1.05*	81 180	5 542	*1.26*
	basic+FT	43.43	*1.45*	2.12E-4	*1.01*	64 478	*1.08*	304 197	*1.02*	80 623	5 293	*1.21*
	basic+peel	40.18	*1.34*	2.12E-4	*1.01*	60 937	*1.02*	305 752	*1.02*	79 866	4 743	*1.08*
	basic+div	57.25	*1.91*	2.15E-4	*1.02*	**59 793**	*1.00*	299 382	*1.00*	77 470	5 251	*1.20*
	basic+full	72.29	*2.42*	2.15E-4	*1.02*	**59 793**	*1.00*	**299 286**	*1.00*	77 470	5 246	*1.19*
NBI	FT	NA										
	FT+FO	NA										
	basic	128.51	*1.61*	2.76E-7	*1.03*	82 027	*1.06*	701 000	*1.08*	367 267	2 810	*1.03*
	basic+FT	822.19	*10.33*	2.83E-7	*1.06*	81 082	*1.05*	693 527	*1.06*	364 203	2 754	*1.01*
	basic+peel	**79.60**	*1.00*	3.48E-7	*1.30*	81 518	*1.05*	699 310	*1.07*	369 013	2 870	*1.05*
	basic+div	122.97	*1.54*	3.82E-7	*1.43*	78 013	*1.01*	**651 614**	*1.00*	341 777	**2 725**	*1.00*
	basic+full	127.15	*1.60*	**2.68E-7**	*1.00*	**77 518**	*1.00*	669 806	*1.03*	354 412	2 730	*1.00*
OSY	FT	TO*										
	FT+FO	792.22	*5.41*	1.60E-4	*3.09*	781 001	*2.83*	6 522 128	*4.11*	31 888	3 795	*2.13*
	basic	966.13	*6.60*	1.46E-4	*2.82*	842 749	*3.05*	8 620 364	*5.43*	78 224	17 896	*10.05*
	basic+FT	990.93	*6.77*	5.83E-5	*1.13*	772 036	*2.80*	4 306 624	*2.71*	61 699	3 015	*1.69*
	basic+peel	**146.34**	*1.00*	**5.17E-5**	*1.00*	278 377	*1.01*	1 609 148	*1.01*	41 230	1 818	*1.02*
	basic+div	154.97	*1.06*	9.07E-5	*1.75*	**276 166**	*1.00*	**1 588 200**	*1.00*	41 696	**1 781**	*1.00*
	basic+full	159.91	*1.09*	5.90E-5	*1.14*	277 888	*1.01*	1 616 378	*1.02*	38 782	1 887	*1.06*
SR	FT	TO										
	FT+FO	602.11	*12.58*	1.21E-3	*3.28*	112 731	*2.96*	1 612 610	*4.22*	4 082	20 537	*29.46*
	basic	83.34	*1.74*	5.71E-4	*1.54*	55 117	*1.45*	611 986	*1.60*	3 215	5 013	*7.19*
	basic+FT	327.30	*6.84*	6.24E-4	*1.69*	49 839	*1.31*	608 430	*1.59*	2 982	5 834	*8.37*
	basic+peel	**47.86**	*1.00*	3.71E-4	*1.00*	38 117	*1.00*	391 902	*1.03*	3 754	895	*1.28*
	basic+div	52.87	*1.10*	**3.70E-4**	*1.00*	38 050	*1.00*	**381 920**	*1.00*	3 792	**697**	*1.00*
	basic+full	57.67	*1.20*	3.70E-4	*1.00*	**38 046**	*1.00*	384 048	*1.01*	3 785	762	*1.09*
TAN	FT	0.25	*1.05*	6.82E-4	*3.58*	1 954	*1.45*	8 280	*2.58*	1 004	279	*1.67*
	FT+FO	0.30	*1.28*	6.75E-4	*3.54*	1 934	*1.44*	8 150	*2.54*	1 004	277	*1.66*
	basic	0.29	*1.25*	3.51E-4	*1.84*	**1 345**	*1.00*	5 030	*1.57*	1 124	249	*1.49*
	basic+FT	0.30	*1.26*	3.30E-4	*1.73*	1 538	*1.14*	3 932	*1.22*	1 160	191	*1.14*
	basic+peel	0.24	*1.03*	**1.91E-4**	*1.00*	1 454	*1.08*	**3 212**	*1.00*	1 212	**167**	*1.00*
	basic+div	0.24	*1.01*	**1.91E-4**	*1.00*	1 454	*1.08*	**3 212**	*1.00*	1 212	**167**	*1.00*
	basic+full	**0.23**	*1.00*	**1.91E-4**	*1.00*	1 454	*1.08*	**3 212**	*1.00*	1 212	**167**	*1.00*
WB	FT	TO										
	FT+FO	1460.02	*1.65*	9.16E-5	*5.79*	940 836	*1.39*	14 760 371	*1.79*	68 222	6 090	*2.04*
	basic	902.31	*1.02*	2.21E-5	*1.39*	709 892	*1.05*	9 192 856	*1.12*	93 261	3 444	*1.15*
	basic+FT	1626.58	*1.83*	1.60E-5	*1.01*	718 056	*1.06*	9 581 054	*1.16*	111 815	3 007	*1.01*
	basic+peel	**887.41**	*1.00*	1.64E-5	*1.03*	680 957	*1.00*	8 425 136	*1.02*	116 333	3 078	*1.03*
	basic+div	1309.66	*1.48*	1.59E-5	*1.01*	**677 914**	*1.00*	8 384 008	*1.02*	121 062	3 047	*1.02*
	basic+full	1632.17	*1.84*	**1.58E-5**	*1.00*	687 219	*1.01*	**8 235 136**	*1.00*	114 090	**2 987**	*1.00*

Table 4: Results for scalable problems.

	Methods	CPU		Δ_H		$\lvert\mathcal{S}_{out}\rvert$		# Nodes		$\lvert\mathcal{Y}_U\rvert$	# Store	
CF3-2	FT	68.36	*2.75*	8.00E-3	*1.16*	115 464	*1.32*	293 597	*1.62*	1 308	4 798	*2.62*
	FT+FO	72.92	*2.93*	8.00E-3	*1.16*	115 464	*1.32*	293 597	*1.62*	1 308	4 798	*2.62*
	basic	32.92	*1.32*	8.84E-3	*1.29*	106 034	*1.22*	321 026	*1.78*	171	12 077	*6.61*
	basic+FT	85.61	*3.44*	7.97E-3	*1.16*	115 046	*1.32*	292 151	*1.62*	1 308	4 981	*2.72*
	basic+peel	**24.90**	*1.00*	7.60E-3	*1.11*	87 274	*1.00*	181 850	*1.01*	25 048	**1 828**	*1.00*
	basic+div	28.51	*1.14*	6.87E-3	*1.00*	87 180	*1.00*	180 844	*1.00*	26 800	1 974	*1.08*
	basic+full	30.59	*1.23*	**6.87E-3**	*1.00*	**87 178**	*1.00*	**180 840**	*1.00*	26 773	1 970	*1.08*
CF3-3	FT	2863.55	*5.02*	3.73E-4	*4.37*	2 902 757	*3.99*	9 712 252	*5.25*	502	473 394	*3.67*
	FT+FO	2998.70	*5.25*	3.73E-4	*4.37*	2 902 757	*3.99*	9 712 252	*5.25*	502	473 394	*3.67*
	basic	2139.42	*3.75*	5.43E-4	*6.38*	3 078 460	*4.23*	12 240 388	*6.62*	105	1836 605	*14.22*
	basic+FT	3330.27	*5.83*	3.73E-4	*4.37*	2 902 672	*3.99*	9 712 158	*5.25*	502	909 435	*7.04*
	basic+peel	612.51	*1.07*	8.65E-5	*1.02*	739 976	*1.02*	2 279 456	*1.23*	22 769	287 863	*2.23*
	basic+div	**570.98**	*1.00*	**8.52E-5**	*1.00*	727 842	*1.00*	**1 849 526**	*1.00*	19 291	**129 130**	*1.00*
	basic+full	597.37	*1.05*	8.54E-5	*1.00*	**727 743**	*1.00*	1 850 018	*1.00*	19 273	129 254	*1.00*
CTP1-3	FT	3.37	*1.32*	1.06E-3	*1.29*	16 568	*1.51*	46 382	*2.12*	1 075	83	*1.60*
	FT+FO	4.27	*1.67*	1.06E-3	*1.29*	16 565	*1.51*	46 380	*2.12*	1 075	83	*1.60*
	basic	4.40	*1.72*	8.27E-4	*1.01*	17 249	*1.58*	56 360	*2.57*	1 149	116	*2.23*
	basic+FT	4.44	*1.73*	8.30E-4	*1.01*	11 190	*1.02*	22 523	*1.03*	1 528	59	*1.13*
	basic+peel	**2.56**	*1.00*	**8.23E-4**	*1.00*	**10 943**	*1.00*	**21 900**	*1.00*	1 618	**52**	*1.00*
	basic+div	2.64	*1.03*	**8.23E-4**	*1.00*	**10 943**	*1.00*	**21 900**	*1.00*	1 618	**52**	*1.00*
	basic+full	2.72	*1.06*	**8.23E-4**	*1.00*	**10 943**	*1.00*	**21 900**	*1.00*	1 618	**52**	*1.00*
CTP1-5	FT	1258.75	*2.17*	1.00E-3	*1.06*	2 307 724	*1.28*	5 538 949	*1.27*	1 585	6 839	*1.06*
	FT+FO	1002.87	*1.73*	1.00E-3	*1.06*	2 307 724	*1.28*	5 538 949	*1.27*	1 585	6 839	*1.06*
	basic	**580.67**	*1.00*	**9.44E-4**	*1.00*	**1 801 418**	*1.00*	4 919 100	*1.13*	1 205	**6 474**	*1.00*
	basic+FT	1159.92	*2.00*	9.84E-4	*1.04*	2 071 267	*1.15*	4 391 666	*1.01*	1 590	6 664	*1.03*
	basic+peel	658.80	*1.13*	9.64E-4	*1.02*	2 172 140	*1.21*	**4 344 402**	*1.00*	1 339	8 966	*1.38*
	basic+div	683.51	*1.18*	9.64E-4	*1.02*	2 172 140	*1.21*	**4 344 402**	*1.00*	1 339	8 966	*1.38*
	basic+full	682.21	*1.17*	9.64E-4	*1.02*	2 172 140	*1.21*	**4 344 402**	*1.00*	1 339	8 966	*1.38*
CTP2-3	FT	4.05	*1.10*	1.37E-3	*1.51*	23 556	*1.85*	91 050	*2.83*	344	231	*1.76*
	FT+FO	4.97	*1.36*	1.37E-3	*1.51*	21 585	*1.70*	88 344	*2.75*	344	210	*1.60*
	basic	5.81	*1.59*	1.27E-3	*1.40*	19 372	*1.52*	73 620	*2.29*	344	176	*1.34*
	basic+FT	4.67	*1.27*	9.16E-4	*1.02*	14 759	*1.16*	36 140	*1.12*	453	138	*1.05*
	basic+peel	**3.67**	*1.00*	9.01E-4	*1.00*	**12 706**	*1.00*	**32 172**	*1.00*	453	**131**	*1.00*
	basic+div	3.81	*1.04*	**9.01E-4**	*1.00*	12 757	*1.00*	32 266	*1.00*	458	133	*1.02*
	basic+full	3.84	*1.05*	9.01E-4	*1.00*	12 744	*1.00*	32 232	*1.00*	458	133	*1.02*
CTP2-5	FT	**1475.13**	*1.00*	1.36E-3	*1.14*	5 820 443	*1.25*	17 583 569	*1.70*	449	20 403	*1.24*
	FT+FO	**1475.13**	*1.00*	1.36E-3	*1.14*	5 820 443	*1.25*	17 583 569	*1.70*	449	20 403	*1.24*
	basic	1989.88	*1.35*	1.51E-3	*1.27*	4 902 975	*1.05*	14 306 090	*1.38*	376	20 743	*1.26*
	basic+FT	1892.20	*1.28*	1.31E-3	*1.11*	5 007 180	*1.08*	11 665 627	*1.13*	473	**16 437**	*1.00*
	basic+peel	1790.75	*1.21*	1.19E-3	*1.00*	**4 650 838**	*1.00*	**10 339 316**	*1.00*	449	20 921	*1.27*
	basic+div	1717.77	*1.16*	**1.18E-3**	*1.00*	4 652 010	*1.00*	10 340 308	*1.00*	452	20 829	*1.27*
	basic+full	1762.81	*1.20*	**1.18E-3**	*1.00*	4 652 010	*1.00*	10 340 308	*1.00*	452	20 829	*1.27*

Table 5: Results for scalable problems.

	Methods	CPU		Δ_H		$\lvert\mathcal{S}_{out}\rvert$		# Nodes		$\lvert\mathcal{Y}_U\rvert$	# Store	
CTP6-3	FT	**4.58**	*1.00*	1.02E-1	*1.49*	34 100	*1.50*	100 174	*2.15*	415	738	*1.19*
	FT+FO	6.23	*1.36*	1.02E-1	*1.49*	34 100	*1.50*	100 174	*2.15*	415	738	*1.19*
	basic	7.46	*1.63*	8.82E-2	*1.30*	30 051	*1.32*	93 142	*2.00*	436	1 286	*2.07*
	basic+FT	6.58	*1.44*	7.97E-2	*1.17*	27 688	*1.22*	58 560	*1.25*	440	672	*1.08*
	basic+peel	5.42	*1.18*	**6.80E-2**	*1.00*	22 922	*1.01*	46 964	*1.01*	433	632	*1.02*
	basic+div	6.04	*1.32*	6.81E-2	*1.00*	22 772	*1.00*	46 678	*1.00*	432	632	*1.02*
	basic+full	6.50	*1.42*	6.81E-2	*1.00*	**22 767**	*1.00*	**46 674**	*1.00*	432	**621**	*1.00*
CTP6-4	FT	440.91	*1.27*	1.44E-1	*1.26*	1 425 155	*1.13*	3 847 452	*1.47*	397	**52 046**	*1.00*
	FT+FO	526.76	*1.52*	1.44E-1	*1.26*	1 425 155	*1.13*	3 847 452	*1.47*	397	**52 046**	*1.00*
	basic	454.02	*1.31*	1.49E-1	*1.31*	1 538 112	*1.22*	4 352 522	*1.66*	435	129 839	*2.49*
	basic+FT	386.89	*1.12*	1.17E-1	*1.03*	1 370 545	*1.09*	2 972 090	*1.14*	444	56 303	*1.08*
	basic+peel	**346.85**	*1.00*	**1.14E-1**	*1.00*	1 258 502	*1.00*	2 619 118	*1.00*	472	56 042	*1.08*
	basic+div	378.64	*1.09*	1.14E-1	*1.00*	**1 257 819**	*1.00*	**2 617 820**	*1.00*	470	56 117	*1.08*
	basic+full	395.30	*1.14*	1.14E-1	*1.00*	**1 257 819**	*1.00*	**2 617 820**	*1.00*	470	56 117	*1.08*
CTP7-4	FT	**8.66**	*1.00*	1.59E-2	*1.69*	41 387	*1.16*	147 548	*1.36*	269	11 214	*1.06*
	FT+FO	10.61	*1.23*	1.59E-2	*1.69*	41 374	*1.16*	147 430	*1.36*	269	11 214	*1.06*
	basic	229.05	*26.46*	3.04E-2	*3.23*	686 737	*19.26*	2 562 940	*23.67*	267	152 764	*14.45*
	basic+FT	10.25	*1.18*	1.15E-2	*1.22*	35 983	*1.01*	108 828	*1.01*	272	**10 574**	*1.00*
	basic+peel	9.94	*1.15*	9.43E-3	*1.00*	**35 649**	*1.00*	**108 274**	*1.00*	299	10 979	*1.04*
	basic+div	12.71	*1.47*	**9.42E-3**	*1.00*	**35 649**	*1.00*	**108 274**	*1.00*	302	10 979	*1.04*
	basic+full	15.05	*1.74*	**9.42E-3**	*1.00*	**35 649**	*1.00*	**108 274**	*1.00*	302	10 979	*1.04*
CTP7-5	FT	1340.76	*4.06*	2.13E-2	*1.80*	1 152 831	*1.15*	3 548 326	*1.28*	268	267 454	*1.00*
	FT+FO	1464.50	*4.43*	2.13E-2	*1.80*	1 152 814	*1.15*	3 548 080	*1.28*	268	267 454	*1.00*
	basic	TO										
	basic+FT	383.49	*1.16*	1.65E-2	*1.40*	1 014 276	*1.01*	2 795 006	*1.01*	282	**266 914**	*1.00*
	basic+peel	**330.53**	*1.00*	1.24E-2	*1.05*	1 004 614	*1.00*	2 774 522	*1.00*	304	271 668	*1.02*
	basic+div	417.43	*1.26*	**1.18E-2**	*1.00*	**1 002 647**	*1.00*	**2 770 202**	*1.00*	309	271 668	*1.02*
	basic+full	415.19	*1.26*	**1.18E-2**	*1.00*	**1 002 647**	*1.00*	**2 770 202**	*1.00*	309	271 668	*1.02*
MOP-7	FT	25.36	*1.76*	6.03E-7	*1.41*	13 498	*1.81*	219 840	*1.54*	19 175	2 432	*2.62*
	FT+FO	26.38	*1.83*	6.03E-7	*1.41*	13 498	*1.81*	219 840	*1.54*	19 175	2 432	*2.62*
	basic	15.10	*1.05*	6.03E-7	*1.41*	13 498	*1.81*	219 840	*1.54*	19 175	2 670	*2.87*
	basic+FT	25.80	*1.79*	6.03E-7	*1.41*	13 498	*1.81*	219 840	*1.54*	19 175	2 670	*2.87*
	basic+peel	23.31	*1.62*	4.53E-7	*1.06*	16 490	*2.21*	313 826	*2.20*	36 327	3 394	*3.65*
	basic+div	**14.39**	*1.00*	**4.28E-7**	*1.00*	**7 453**	*1.00*	**142 394**	*1.00*	48 304	**929**	*1.00*
	basic+full	15.33	*1.07*	**4.28E-7**	*1.00*	**7 453**	*1.00*	**142 394**	*1.00*	48 304	**929**	*1.00*
MOP-10	FT	736.43	*19.38*	1.05E-6	*1.43*	105 114	*11.79*	2 323 572	*9.02*	26 784	18 294	*13.82*
	FT+FO	731.71	*19.26*	1.05E-6	*1.43*	105 114	*11.79*	2 323 572	*9.02*	26 784	18 294	*13.82*
	basic	212.18	*5.58*	1.05E-6	*1.43*	105 114	*11.79*	2 323 572	*9.02*	26 784	20 842	*15.74*
	basic+FT	433.52	*11.41*	1.05E-6	*1.43*	105 114	*11.79*	2 323 572	*9.02*	26 784	20 842	*15.74*
	basic+peel	298.81	*7.86*	9.37E-7	*1.28*	118 310	*13.27*	2 834 216	*11.00*	39 957	21 820	*16.48*
	basic+div	**38.00**	*1.00*	**7.33E-7**	*1.00*	**8 915**	*1.00*	**257 740**	*1.00*	90 573	**1 324**	*1.00*
	basic+full	42.25	*1.11*	**7.33E-7**	*1.00*	**8 915**	*1.00*	**257 740**	*1.00*	90 573	**1 324**	*1.00*
MOP-13	FT	TO										
	FT+FO	TO										
	basic	3352.79	*50.70*	1.54E-6	*1.32*	838 042	*99.96*	23 526 888	*74.08*	33 884	174 020	*111.19*
	basic+FT	TO										
	basic+peel	TO										
	basic+div	**66.13**	*1.00*	**1.16E-6**	*1.00*	**8 384**	*1.00*	**317 600**	*1.00*	133 631	**1 565**	*1.00*
	basic+full	69.23	*1.05*	**1.16E-6**	*1.00*	**8 384**	*1.00*	**317 600**	*1.00*	133 631	**1 565**	*1.00*

[12] M. Ehrgott and X. Gandibleux. Bound sets for biobjective combinatorial optimization problems. *Computers & Operations Research*, 34(9):2674 – 2694, 2007.

[13] J. Fernández and B. Tóth. Obtaining an outer approximation of the efficient set of nonlinear biobjective problems. *Journal of Global Optimization*, 38(2):315–331, 2007.

[14] J. Fernández and B. Tóth. Obtaining the efficient set of nonlinear biobjective optimization problems via interval branch-and-bound methods. *Computational Optimization and Applications*, 42(3):393–419, 2009.

[15] M. Gavanelli. An algorithm for Multi-Criteria Optimization in CSPs. In *European Conference on Artificial Intelligence*. IOS Press, 2002.

[16] A. Goldsztejn, F. Domes, and B. Chevalier. First order rejection tests for multiple-objective optimization. *Journal of Global Optimization*, 58(4):653–672, 2014.

[17] F. Goualard. *GAOL 3.1.1: Not Just Another Interval Arithmetic Library*. LINA, 4.0 edition, 2006.

[18] L. Granvilliers and F. Benhamou. Algorithm 852: RealPaver: an interval solver using constraint satisfaction techniques. *ACM Transactions Mathematical Software*, 32(1):138–156, 2006.

[19] E. Hansen and G. W. Walster. *Global Optimization Using Interval Analysis - Revised And Expanded*. CRC Press, 2003.

[20] R. Hartert and P. Schaus. A Support-Based Algorithm for the Bi-Objective Pareto Constraint. In *AAAI Conference on Artificial Intelligence*, 2014.

[21] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis with Examples in Parameter and State Estimation, Robust Control and Robotics*. Springer-Verlag, 2001.

[22] R. B. Kearfott. Interval Computations: Introduction, Uses, and Resources. *Euromath*, Bulletin 2(1):95–112, 1996.

[23] R. Baker Kearfott. *Rigorous Global Search: Continuous Problems*. Kluwer Academic Publishers, 1996.

[24] I. Y. Kim and O. L. de Weck. Adaptive weighted-sum method for bi-objective optimization: Pareto front generation. *Structural and Multidisciplinary Optimization*, 29(2):149–158, February 2005.

[25] K. Klamroth, R. Lacour, and D. Vanderpooten. On the representation of the search region in multi-objective optimization. *European Journal of Operational Research*, 245(3):767 – 778, 2015.

[26] B. J. Kubica and A. Woźniak. Interval methods for computing the pareto-front of a multicriterial problem. In *International conference on Parallel processing and applied mathematics*, PPAM'07, pages 1382–1391, 2008.

[27] B. J. Kubica and A. Woźniak. Using the second-order information in pareto-set computations of a multi-criteria problem. In Kristján Jónasson, editor, *Applied Parallel and Scientific Computing*, volume 7134 of *LNCS*, pages 137–147. Springer, 2012.

[28] B. J. Kubica and A. Woźniak. Tuning the interval algorithm for seeking pareto sets of multi-criteria problems. In Pekka Manninen and Per ster, editors, *Applied Parallel and Scientific Computing*, volume 7782 of *LNCS*, pages 504–517. Springer, 2013.

[29] Y. Lebbah, C. Michel, and M. Rueher. An efficient and safe framework for solving optimization problems. *Journal of Computational and Applied Mathematics*, 199(2):372 – 377, 2007.

[30] O. Lhomme. Consistency Techniques for Numeric CSPs. In *International Joint Conference on Artificial Intelligence*, pages 232–238, 1993.

[31] O. Lhomme. Quick Shaving. In *AAAI Conference on Artificial Intelligence*, pages 411–415. AAAI Press, 2005.

[32] B. Martin. *Rigorous algorithms for nonlinear biobjective optimization*. Phd thesis, Université de Nantes, October 2014.

[33] B. Martin, A. Goldsztejn, L. Granvilliers, and C. Jermann. On continuation methods for non-linear bi-objective optimization: towards a certified interval-based approach. *Journal of Global Optimization*, pages 1–14, 2014. (Online first).

[34] K. Miettinen. *Nonlinear Multiobjective Optimization*, volume 12 of *Int. Series in Operations Research and Management Science*. Kluwer, 1999.

[35] R. Moore. *Interval Analysis*. Prentice-Hall, 1966.

[36] A. Neumaier. *Interval Methods for Systems of Equations*. Cambridge University Press, 1991.

[37] A. Neumaier. Second-order sufficient optimality conditions for local and global nonlinear programming. *Journal of Global Optimization*, 9(2):141–151, 1996.

[38] A. Neumaier. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, 13:271–369, 5 2004.

[39] B. Neveu, G. Trombettoni, and I. Araya. Adaptive constructive interval disjunction: algorithms and experiments. *Constraints*, 20(7):452–467, 2015.

[40] A. Osyczka and S. Kundu. A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. *Structural optimization*, 10(2):94–99, 1995.

[41] A. Przybylski, X. Gandibleux, and M. Ehrgott. A two phase method for multi-objective integer programming and its application to the assignment problem with three objectives. *Discrete Optimization*, 7(3):149 – 165, 2010.

[42] M. Reyes-Sierra and C. A. C. Coello. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International journal of computational intelligence research*, 2(3):287–308, 2006.

[43] G.R. Ruetsch. An interval algorithm for multi-objective optimization. *Structural and Multidisciplinary Optimization*, 30:27–37, 2005.

[44] S.M. Rump. A note on epsilon-inflation. *Reliable Computing*, 4:371–375, 1998.

[45] O. Schütze, A. Dell'Aere, and M. Dellnitz. On continuation methods for the numerical treatment of multi-objective optimization problems. In J. Branke et al., editor, *Practical Approaches to Multi-Objective Optimization*, number 04461 in Dagstuhl Seminar, 2005.

[46] B Tóth and J Fernández. Interval methods for single and bi-objective optimization problems-applied to competitive facility location problems. *Saarbrücken: Lambert Academic Publishing*, 2010.

[47] G. Trombettoni, I. Araya, B. Neveu, and G. Chabert. Inner regions and interval linearizations for global optimization. In *AAAI Conference on Artificial Intelligence*, 2011.

[48] G. Trombettoni and G. Chabert. Constructive interval disjunction. In *International conference on Principles and Practice of Constraint Programming*, volume 4741 of *LNCS*, pages 635–650. Springer, 2007.

[49] P. Van Hentenryck, L. Michel, and Y. Deville. *Numerica: A Modeling Language for Global Optimization*. MIT press, 1997.

[50] C. Villa, E. Lozinguez, and R. Labayrade. Multi-objective optimization under uncertain objectives: Application to engineering design problem. In R. C. Purshouse, P. J. Fleming, C. M. Fonseca, S. Greco, and J. Shaw, editors, *Evolutionary Multi-Criterion Optimization: 7th International Conference, EMO 2013, Sheffield, UK, March 19-22, 2013. Proceedings*, pages 796–810. Springer Berlin Heidelberg, 2013.

[51] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, and S. Tiwari. Multiobjective optimization test instances for the cec 2009 special session and competition. *University of Essex, Colchester, UK and Nanyang technological University, Singapore, special session on performance assessment of multi-objective optimization algorithms, technical report*, 264, 2008.

[52] Z. Zhang. Immune optimization algorithm for constrained nonlinear multiobjective optimization problems. *Applied Soft Computing*, 7(3):840 – 857, 2007.

[53] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, and V.G. da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.

Appendix A. Benchmarks

Appendix A.1. Kim and DeWeck (KIM) [24]

A problem with 2 objectives and 2 variables (only bound constraints).

$$\left[\begin{array}{lll} \min & f_1(x) := & -(3(1-x_1)^2\exp(-x_1^2-(x_2+1)^2) \\ & & -10(x_1/5.0-x_1^3-x_2^5)\exp(-x_1^2-x_2^2) \\ & & -3\exp(-(x_1+2)^2-x_2^2)+0.5(2x_1+x_2)) \\ \min & f_2(x) := & -(3(1+x_2)^2\exp(-x_2^2-(1-x_1)^2) \\ & & -10(-x_2/5.0+x_2^3+x_1^5)\exp(-x_1^2-x_2^2) \\ & & -3\exp(-(2-x_2)^2-x_1^2)) \end{array}\right], \tag{A.1}$$

with $-3 \leq \{x_1, x_2\} \leq 3$.

Appendix A.2. BINH [7]

A problem with 2 objectives, 2 variables and 2 inequality constraints. Objectives and constraints are quadratic.

$$\left[\begin{array}{lll} \min & f_1(x) := & 4x_1^2+4x_2^2 \\ \min & f_2(x) := & (x_1-5)^2+(x_2-5)^2 \\ \text{s.t} & g_1(x) := & (x_1-5)^2+x_2^2-25 \leq 0 \\ & g_2(x) := & -(x_1-8)^2-(x_2+3)^2+7.7 \leq 0 \end{array}\right], \tag{A.2}$$

with $0 \leq x_1 \leq 5$ and $0 \leq x_2 \leq 3$.

Appendix A.3. Tanaka (TAN) [7]

A problem with 2 objectives, 2 variables and 2 inequality constraints.

$$\left[\begin{array}{lll} \min & f_1(x) := & x_1 \\ \min & f_2(x) := & x_2 \\ \text{s.t} & g_1(x) := & -x_1^2-x_2^2+1+0.1\cos(16\arctan(x_1/x_2)) \leq 0 \\ & g_2(x) := & 2(x_1-0.5)^2+2(x_2-0.5)^2-1 \leq 0 \end{array}\right], \tag{A.3}$$

with $0 \leq x_1 \leq 5$ and $0 \leq x_2 \leq 3$.

Appendix A.4. Osyczka (OSY) [40]

A problem with 2 objectives, 6 inequality constraints and 6 variables.

$$\left[\begin{array}{lll} \min & f_1(x) := & -(25(x_1-2)^2+(x_2-2)^2+(x_3-1)^2 \\ & & +(x_4-4)^2+(x_5-1)^2) \\ \min & f_2(x) := & x_1^2+x_2^2+x_3^2+x_4^2+x_5^2+x_6^2 \\ \text{s.t} & g_1(x) := & -(x_1+x_2-2)\leq 0 \qquad g_4(x) := -(-x_1+3x_2+2)\leq 0 \\ & g_2(x) := & -(-x_1-x_2+6)\leq 0 \quad g_5(x) := -(-(x_3-3)^2-x_4+4)\leq 0 \\ & g_3(x) := & -(x_1-x_2+2)\leq 0 \qquad g_6(x) := -((x_5-3)^2+x_6-4)\leq 0 \end{array}\right], \tag{A.4}$$

with $0 \leq \{x_1, x_2, x_6\} \leq 10$, $1 \leq \{x_3, x_5\} \leq 5$ and $0 \leq x_4 \leq 6$.

Appendix A.5. NBI [9]

A problem with 2 objectives, 2 equality constraints, 1 inequality constraint and 5 variables.

$$\left[\begin{array}{lll} \min & f_1(x) := & x_1^2+x_2^2+x_3^2+x_4^2+x_5^2 \\ \min & f_2(x) := & 3x_1+2x_2-x_3/3+0.001(x_4-x_5)^3 \\ \text{s.t} & h_1(x) := & x_1+2x_2-x_3-0.5x_4+x_5-2=0 \\ & h_2(x) := & 4x_1-2x_2+0.8x_3+0.6x_4+0.5x_5^2=0 \\ & g_1(x) := & x_1^2+x_2^2+x_3^2+x_4^2+x_5^2-10\leq 0 \end{array}\right], \tag{A.5}$$

with $-50 \leq x_i \leq 50$.

Appendix A.6. SpeedReducer (SR) [52]

A problem modeling the design of a speed reducer, with 2 objectives, 11 inequality constraints and 7 variables.

$$\left[\begin{array}{lll} \min & f_1(x) := 0.7854x_1x_2^2(\frac{10x_3^2}{3}+14.933x_3-43.0934) & \\ & -1.508x_1(x_6^2+x_7^2)+7.477(x_6^3+x_7^3)+0.7854(x_4x_6^2+x_5x_7^2) & \\ & f_2(x) := \sqrt{(745x_4/x_2x_3)^2+1.69\times 10^7}/0.1x_6^3 & \\ \text{s.t} & g_1(x) := \frac{1}{x_1x_2^2x_3}-\frac{1}{27}\leq 0 & g_6(x) := \frac{x_1}{x_2}-12\leq 0 \\ & g_2(x) := \frac{1}{x_1x_2^2x_3^2}-\frac{1}{397.5}\leq 0 & g_7(x) := 5-\frac{x_1}{x_2}\leq 0 \\ & g_3(x) := \frac{x_4^3}{x_2x_3x_6^4}-\frac{1}{1.93}\leq 0 & g_8(x) := 1.9-x_4+1.5x_6\leq 0 \\ & g_4(x) := \frac{x_5^3}{x_2x_3x_7^4}-\frac{1}{1.93}\leq 0 & g_9(x) := 1.9-x_5+1.1x_7\leq 0 \\ & g_5(x) = x_2x_3-40\leq 0 & g_{10}(x) := f_1(x)-3300\leq 0 \\ & g_{11}(x) := \sqrt{(745x_5/x_2x_3)^2+1.575\times 10^8}/0.1x_7^3-1100\leq 0 & \end{array}\right], \tag{A.6}$$

with $2.6 \leq x_1 \leq 3.6$, $0.7 \leq x_2 \leq 0.8$, $17 \leq x_3 \leq 28$, $7.3 \leq \{x_4, x_5\} \leq 8.3$, $2.9 \leq x_6 \leq 3.9$ and $5 \leq x_7 \leq 5.5$.

Appendix A.7. Welded Beam (WB) [50]

A problem for the robust design of a Welded Beam, with 4 variables and 4 inequality constraints. This problem orignally contains a random parameter E, following a normal distribution, that we fixed to its mean value (see below).

$$\left[\begin{array}{lll} \min & f_1(x) := & 1.104x_1^2x_2 + 0.048x_3x_4(14+x_2) \\ \min & f_2(x) := & \frac{4FL^3}{Ex_3^3x_4} \\ \text{s.t} & g_1(x) := & \tau - 13600 \leq 0 \quad g_3(x) := 6000 - P_c \leq 0 \\ & g_2(x) := & \sigma - 30000 \leq 0 \quad g_4(x) := x_1 - x_4 \leq 0 \end{array}\right], \tag{A.7}$$

with $0.125 \leq \{x_1, x_4\} \leq 5$, $0.1 \leq \{x_2, x_3\} \leq 10$; and $F = 6000$, $L = 14$, $E = 30 \cdot 10^6$; and

$$\begin{aligned} \tau &:= \sqrt{(\tau')^2 + (\tau'')^2 + \frac{x_2\tau'\tau''}{\sqrt{0.25(x_2^2 + (x_1+x_3)^2)}}} \\ \tau' &:= \frac{6000}{\sqrt{2}x_1x_2}; \quad \tau'' := \frac{6000(14+0.5x_2)\sqrt{0.25(x_2^2+(x_1+x_3)^2)}}{2(0.707x_1x_2(x_2^2/12 + 0.25(x_1+x_3)^2))} \\ \sigma &:= \frac{504000}{x_3^2x_4}; \quad P_c := 64746.022(1 - 0.0282346x_3)x_3x_4^3 \end{aligned}$$

Appendix A.8. MOP [45]

A scalable problem with 2 objectives and n variables (only bound constraints).

$$\left[\begin{array}{lll} \min & f_1(x) := & \sum_{j\neq 1}(x_j - 1)^2 + (x_1 - 1)^4 \\ \min & f_2(x) := & \sum_{j\neq 2}(x_j + 1)^2 + (x_2 + 1)^4 \end{array}\right], \tag{A.8}$$

with $-5 \leq x_i \leq 5 \ \forall i \in 1..n$.

Appendix A.9. CTP1 [10]

Biobjective problems with n variables and 2 constraints based on the objectives.

$$\left[\begin{array}{lll} \min & f_1(x) := & x_1 \\ \min & f_2(x) := & \phi(x)\exp(-f_1(x)/\phi(x)) \\ \text{s.t} & g_j(x) := & -f_2(x) + a_j\exp(-b_jf_1(x)) \leq 0 \quad \forall j = 1,2 \end{array}\right], \tag{A.9}$$

with $0 \leq x_i \leq 1 \ \forall i \in 1, \ldots, n$; and $a_1 = 0.858$, $a_2 = 0.728$, $b_1 = 0.541$ and $b_2 = 0.295$. We selected $\phi(x) := 1 + 9\sum_{i=1}^{n} x_i$.

Appendix A.10. CTP2, CTP6 and CTP7 [10]

Biobjective problems with n variables and 1 constraint based on the objectives.

$$\left[\begin{array}{lll} \min & f_1(x) := & x_1 \\ \min & f_2(x) := & \phi(x)\exp(-f_1(x)/\phi(x)) \\ \text{s.t} & g_1(x) := & a|\sin(b\pi(\sin(\theta)(f_2(x)-e)+\cos(\theta)f_1(x))^c)|^d \\ & & -\cos(\theta)(f_2(x)-e)+\sin(\theta)f_1(x) \leq 0 \end{array}\right], \quad \text{(A.10)}$$

with $0 \leq x_i \leq 1\ \forall i \in 1,\ldots,n$, and $\phi(x) := 1 + 9\sum_{i=1}^{n} x_i$.

The parameters are for CTP2

$$\theta = -0.2\pi; \quad a = 0.2; \quad b = 10; \quad c = 1 \quad d = 6; \quad e = 1;$$

for CTP6

$$\theta = 0.1\pi; \quad a = 40; \quad b = 0.5; \quad c = 1 \quad d = 2; \quad e = -2;$$

and for CTP7

$$\theta = -0.05\pi; \quad a = 40; \quad b = 5; \quad c = 1 \quad d = 6; \quad e = 0.$$

Appendix A.11. CF3 [51]

Biobjective problems with n variables and 1 constraint based on the objectives.

$$\left[\begin{array}{lll} \min & f_1(x) := & x_1 + \frac{2}{|J_1|}\left(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right)+2\right) \\ \min & f_2(x) := & 1 - x_1^2 \\ & & +\frac{2}{|J_2|}\left(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right)+2\right) \\ \text{s.t} & g_1(x) := & -f_2(x) - f_1(x)^2 \\ & & +\sin(2\pi(f_1(x)^2 - f_2 + 1)) + 1 \leq 0 \end{array}\right], \quad \text{(A.11)}$$

with $0 \leq x_1 \leq 1$, $-2 \leq x_i \leq 2, \forall i = 2,\ldots,n$. In addition,

$$y_i = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right), \forall j = 2,\ldots,n$$

and $J_1 := \{j : j \text{ odd}, 2 \leq j \leq n\}$, $J_2 := \{j : j \text{ even}, 2 \leq j \leq n\}$,