

# Obtaining an outer approximation of the efficient set of nonlinear biobjective problems\*

José Fernández and Boglárka Tóth<sup>†</sup>

*Dpt. Statistics and Operations Research, University of Murcia, Spain*

November 14, 2006

**Abstract.** A new method for obtaining an outer approximation of the efficient set of nonlinear biobjective optimization problems is presented. It is based on the well known “constraint method”, and obtains a superset of the efficient set by computing the regions of  $\delta$ -optimality of a finite number of single objective constraint problems. An actual implementation, which makes use of interval tools, shows the applicability of the method and the computational studies on a set of competitive location problems demonstrate its efficiency.

**Keywords:** Nonlinear biobjective optimization, Efficient set, Outer approximation, Constraint method, Interval analysis, Competitive location

## 1. Introduction

Multiobjective optimization problems are ubiquitous. Many real-life problems require taking several conflicting points of view into account. In fact, although the origins of the multiobjective optimization literature are linked to utility theory, game theory, linear production theory and economics (see [21]), we now can find applications in many and diverse fields, such as portofolio optimization [11], jury selection [45], airline operations [12], radiation therapy [35], manpower planning [46] or reservoir management [1], among others. In [50], White mentions more than 500 applications between 1955 and 1986. Classical references on multiobjective optimization are the books [4, 5, 47, 51, 52]. Other more recent books are [9, 10, 19, 38].

In this paper we restrict ourselves to the biobjective case, that is, to the problem

$$\begin{aligned} \min \quad & \{f_1(x), f_2(x)\} \\ \text{s.t.} \quad & x \in S \subseteq \mathbb{R}^n \end{aligned} \tag{1}$$

---

\* This paper has been supported by the Ministry of Education and Science of Spain under the research project SEJ2005-06273/ECON, in part financed by the European Regional Development Fund (ERDF).

<sup>†</sup> On leave from the Research Group on Artificial Intelligence of the Hungarian Academy of Sciences and the University of Szeged, H-6720 Szeged, Aradi vértanúk tere 1., Hungary.



where  $f_1, f_2 : \mathbb{R}^n \rightarrow \mathbb{R}$  are two real-valued functions. Let us denote by  $f(x) = (f_1(x), f_2(x))$  the vector of objective functions, and by  $Z = f(S)$  the image of the feasible region.

When dealing with multiobjective problems we need to clarify what ‘solving’ a problem means. Some widely known definitions to explain the concept of solution of (1) follow.

*Definition 1.* A feasible vector  $x^* \in S$  is said to be *efficient* iff there does not exist another feasible vector  $x \in S$  such that  $f_i(x) \leq f_i(x^*)$  for all  $i = 1, 2$ , and  $f_j(x) < f_j(x^*)$  for at least one index  $j$ . The set  $S_E$  of all the efficient points is called the *efficient set*.

Efficiency is defined in the decision space. The corresponding definition in the criterion space is as follows.

*Definition 2.* An objective vector  $z^* = f(x^*) \in Z$  is said to be *nondominated* iff  $x^*$  is efficient. The set  $Z_N$  of all nondominated vectors is called the *nondominated set*.

In this paper we assume that both  $S_E$  and  $Z_N$  are bounded. Another related concept widely used is weak efficiency.

*Definition 3.* A feasible vector  $x^* \in S$  is said to be *weakly efficient* iff there does not exist another feasible vector  $x \in S$  such that  $f_i(x) < f_i(x^*)$  for  $i = 1, 2$ .

Notice that any efficient point is weakly efficient too, but the converse does not hold in general.

Ideally, solving (1) means obtaining the whole efficient set, that is, all the points which are efficient. That set might be described analytically as a closed formula, numerically as a set of points, or in mixed form as a parameterized set of points. Unfortunately, as pointed out in [43], for the majority of multiobjective optimization problems, it is not easy to obtain such a description, since the efficient set includes typically a very large number or infinite number of points. The methods proposed in the literature with that purpose are specialized either for particular problems (for instance, in [39] it is shown how to obtain the whole efficient set of some location problems) or for a particular class of multiobjective problems (for instance, the multiobjective simplex methods for the linear case [19]). To the extent of our knowledge, only one general method (see [18]) has been proposed in the literature with that purpose for the general nonlinear biobjective problem (1). The reason for this lack of methods is that even obtaining a single efficient point of a nonlinear biobjective problem can be a difficult task. That

is why some authors have proposed to present to the decision-maker a ‘representative set’ of efficient points which suitably represent the whole efficient set (either by modifying the definition of efficiency [3] or by selecting a finite set of efficient points with the criteria of coverage, uniformity and cardinality as quality measures [44]) or an ‘approximation’ of the efficient set by means of sets with a simpler structure (see [43] for a survey of methods with that aim).

The approach in this paper is similar to that in [18]: we offer a *superset* which tightly contains the complete efficient set. However, the method presented in this paper is completely different from that in [18], and what is more important, it is much faster (see Subsection 5.4). By drawing in the image space that superset the decision-maker can easily see the trade-off between the two objectives, i.e., how one objective improves as the other gets worse. Something similar can be done in the decision space, by drawing the superset in a color scale depending on the objective value of one of the objectives.

The paper is organized as follows. In the following section we present the tools used to derive our method. It is in Section 3 where we introduce our constraint-like method, which provides a superset of the efficient set of (1). In Section 4 we detail how the constraint-like method can be carried out in practice using interval tools. Some computational studies are reported in Section 5. The paper ends with conclusions and points for future research.

## 2. Preliminaries

### 2.1. THE CONSTRAINT METHOD

Although most of the literature on multiobjective optimization deals with either *discrete* or continuous multiobjective *linear* problems, we can also find many references dealing with *nonlinear* multiobjective optimization problems (see [13, 38] and the references therein), which is the target of this paper. In fact, there is a great and rich variety of methods for finding efficient points of nonlinear multiobjective optimization problems (see [13, 38] and the references therein), e.g., weighting method, lexicographic method, . . . , but among them, only a few (e.g., the constraint method, reference point methods, . . . ) are able to detect *all* nondominated points. Probably, the constraint method is the most famous among them. The rationale behind the constraint method is rather simple. One of the objective functions, say  $f_1$ , is selected to be minimized, whereas the other one,  $f_2$ , is converted into a

constraint by setting an upper bound  $f_2^{ub}$  to it <sup>1</sup>. The single objective problem to be solved, called *constraint problem*, is then

$$\begin{aligned} \min \quad & f_1(x) \\ \text{s.t.} \quad & f_2(x) \leq f_2^{ub} \\ & x \in S \end{aligned} \tag{2}$$

The goodness of the constraint method can be seen in the following three theorems (for a proof, see for instance [38]).

*Theorem 1.* The solution of the constraint problem (2) is weakly efficient.

*Theorem 2.* A feasible vector  $x^* \in S$  is efficient if and only if it is a solution of the constraint problems

$$\begin{aligned} \min \quad & f_1(x) \\ \text{s.t.} \quad & f_2(x) \leq f_2(x^*) \\ & x \in S \end{aligned} \quad \text{and} \quad \begin{aligned} \min \quad & f_2(x) \\ \text{s.t.} \quad & f_1(x) \leq f_1(x^*) \\ & x \in S \end{aligned} .$$

Instead of having to solve those two constraint problems in Theorem 2, efficiency can also be guaranteed with only one of them, provided that it has a unique solution, as the following theorem shows.

*Theorem 3.* A feasible vector  $x^* \in S$  is efficient if it is a unique solution of any of the constraint problems in Theorem 2.

From the previous theorems it follows that it is possible to find *all* the efficient solutions of *any* biobjective optimization problem by the constraint method. However, we need to solve one or two problems to find one nondominated point, which means that if the nondominated set is not a discrete set (as it is usually the case in continuous multiobjective optimization) then the method is not practical for finding the complete efficient set.

## 2.2. OBTAINING A REGION OF $\delta$ -OPTIMALITY

Consider a single-objective problem

$$\begin{aligned} \min \quad & h(x) \\ \text{s.t.} \quad & x \in Y \subseteq \mathbb{R}^n \end{aligned} \tag{3}$$

---

<sup>1</sup> We always minimize  $f_1$  subject to a constraint on  $f_2$ , but a similar process can be developed interchanging the functions.

where  $h : \mathbb{R}^n \rightarrow \mathbb{R}$  is a real-valued function and  $Y$  is a general set defined by any kind of constraints. If we denote by  $h^*$  the optimal value of (3), the region of  $\delta$ -optimality of (3) is the set

$$R_\delta = \{x \in Y : h(x) - h^* \leq \delta \cdot |h^*|\}.$$

The other tool that we need to derive the method of the next section is a procedure for obtaining the *region of  $\delta$ -optimality* of a given problem. In [40], a branch-and-bound method for obtaining the region of  $\delta$ -optimality of (3) is presented. It consists of two phases. The first one entails the determination of the optimal objective value of (3) up to a prespecified relative precision  $\epsilon$ . The second phase consists of the determination of  $R_\delta$ , up to a prespecified precision  $\eta$ .

The output of the algorithm is two lists of subsets,  $LIR_\delta$  and  $LOR_\delta$ . The union of the subsets in the first list,  $IR_\delta = \cup_{Q \in LIR_\delta} Q$ , intersected with  $Y$  gives an *inner approximation* of  $R_\delta$  (a subset of  $R_\delta$ ). The union of the subsets in both lists,  $OR_\delta = \cup_{Q \in LIR_\delta \cup LOR_\delta} Q$ , intersected with  $Y$  forms an *outer approximation* of  $R_\delta$  (a superset of  $R_\delta$ ), guaranteed to lie entirely within  $R_{\delta+\eta(1+\delta)}$  (see Figure 1), i.e.,

$$IR_\delta \cap Y \subseteq R_\delta \subseteq OR_\delta \cap Y \subseteq R_{\delta+\eta(1+\delta)}.$$

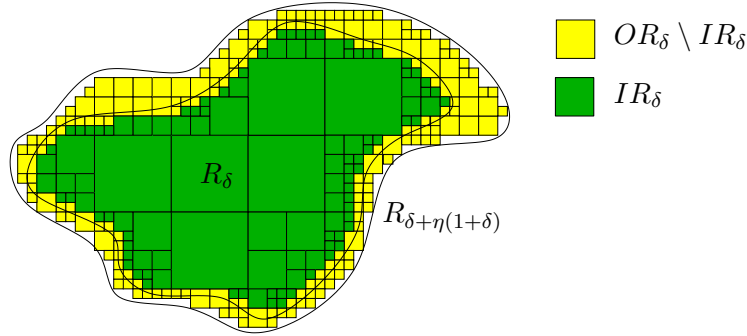


Figure 1. Inner and outer approximation of  $R_\delta$ .

There exist other procedures for obtaining subsets of  $R_\delta$  (see for instance [6, 32]) but for our constraint-like method we need the whole set  $R_\delta$ .

### 3. The constraint-like method

As explained above, with the classical constraint method we have to solve one (or two) constraint problems of the form (2) in order to be

able to obtain a single nondominated point,  $z^*$ . However, if in addition to solving (2) to optimality we compute its region of  $\delta$ -optimality, then  $R_\delta$  contains a ‘portion’ of the efficient set whose values in the criterion space are close to  $z^*$ . The idea of the constraint-like method to obtain a superset containing the whole efficient set is simply this: considering that  $Z_N$  is plotted in the criterion space, we sweep the nondominated set from (say) top to bottom by obtaining the regions of  $\delta$ -optimality of a finite sequence of constraint problems, whose type is a modification of (2), by choosing appropriate upper bounds  $f_2^{ub}$  for the  $f_2$  function. Next, we give the details.

### 3.1. THE METHOD

The basic type of constraint problems that we will use in our method is exactly the same as the one used in the classical constraint method. We will rewrite it as

$$(P_i) \quad \begin{array}{ll} \min & f_1(x) \\ \text{s.t.} & f_2(x) \leq f_2^{(i)} \\ & x \in S \end{array}$$

where  $f_2^{(i)}$  is a given constant defined below. Let  $\hat{x}^{(i)}$  denote an optimal solution of  $(P_i)$ ,  $i \geq 0$ , and let  $R_\delta^{(i)}$  be the region of  $\delta$ -optimality of  $(P_i)$ , that is,

$$R_\delta^{(i)} = \{x \in S : f_2(x) \leq f_2^{(i)}, f_1(x) - f_1(\hat{x}^{(i)}) \leq \delta \cdot |f_1(\hat{x}^{(i)})|\}.$$

The feasible set of  $(P_i)$  is  $Y^{(i)} = \{x \in S : f_2(x) \leq f_2^{(i)}\}$ .

In the first problem that we will consider,  $(P_0)$ , we set  $f_2^{(0)} = +\infty$ . Thus, problem  $(P_0)$  is in fact the single objective problem

$$\begin{array}{ll} \min & f_1(x) \\ \text{s.t.} & x \in S \end{array}$$

Once we have solved problem  $(P_i)$  and have obtained an outer approximation  $OR_\delta^{(i)}$  of  $R_\delta^{(i)}$  with the help of the procedure mentioned in Subsection 2.2, the constant  $f_2^{(i+1)}$  for the next problem  $(P_{i+1})$  is given by

$$f_2^{(i+1)} = \min\{\overline{F}_2(Q) : Q \in LIR_\delta^{(i)} \cup LOR_\delta^{(i)}, Q \cap R_\delta^{(i)} \neq \emptyset\} \quad (4)$$

where  $\overline{F}_2(Q)$  is an upper bound on all objective values of  $f_2$  found within the subset  $Q$  (see Figure 2). However, from a computational point of view, it can be better to set

$$f_2^{(i+1)} = \min\{\overline{F}_2(Q) : Q \in LIR_\delta^{(i)}, Q \cap Y^{(i)} \neq \emptyset\} \quad (5)$$

although this is a worst (higher) value than the one obtained with (4). Using (5) we only have to check whether a subset  $Q$  in  $LIR_\delta^{(i)}$  contains at least one feasible point. If so, we take that box into account for calculating the minimum in (5).

Let us denote by  $Q_N^{(i)}$  the subset at which the previous minimum (either (4) or (5)) is attained, i.e.,  $f_2^{(i+1)} = \overline{F}_2(Q_N^{(i)})$ .

*Lemma 1.*  $f_1(\hat{x}^{(i+1)}) \leq f_1(\hat{x}^{(i)}) + \delta|f_1(\hat{x}^{(i)})|$ .

*Proof.* Since  $f_2^{(i+1)} = \overline{F}_2(Q_N^{(i)})$ , we have in particular that all the points in  $Q_N^{(i)} \cap S$  are feasible for  $(P_{i+1})$ . On the other hand,  $Q_N^{(i)}$  contains at least one point  $\tilde{x}$  in  $R_\delta^{(i)}$ , and thus, its objective value  $f_1(\tilde{x})$  is less than or equal to  $f_1(\hat{x}^{(i)}) + \delta|f_1(\hat{x}^{(i)})|$ . Thus, the optimal value of  $(P_{i+1})$ ,  $f_1(\hat{x}^{(i+1)})$ , must be less than or equal to  $f_1(\hat{x}^{(i)}) + \delta|f_1(\hat{x}^{(i)})|$ .

Unfortunately, the use of this type of constraint problems is not enough to guarantee that the algorithm obtains an outer approximation of the whole efficient set. We also need a second type of constraint problems, namely,

$$\begin{aligned} & \min f_1(x) \\ (\check{P}_{i+j}) \quad & \text{s.t. } f_2(x) \leq f_2^{(i)} \\ & x \in S \\ & f_1(x) \geq f_1(\hat{x}^{(i)}) + j \cdot \delta|f_1(\hat{x}^{(i)})| \end{aligned}$$

Notice that whereas the constraint in the problems of type  $(P_i)$  forces to the image of the feasible set of the problems to go down in the criterion space as  $i$  increases, the additional constraint in the problems of type  $(\check{P}_{i+j})$  forces it to go to the right (see Figure 2) as  $j$  increases.

The method that we propose to obtain an outer approximation of the efficient set of (1) is the following (see Figure 2):

### Constraint-like method (CLM)

1.  $i \leftarrow 0$ ,  $f_2^{(0)} \leftarrow +\infty$ .
2. While  $(P_i)$  is feasible
  - a) Solve  $(P_i)$  and obtain an outer approximation  $OR_\delta^{(i)}$  of  $R_\delta^{(i)}$  using the procedure mentioned in Subsection 2.2.
  - b) Calculate  $f_2^{(i+1)}$  as given by (4) or (5).
  - c) If  $f_2^{(i+1)} \not\leq f_2^{(i)}$  then

- i)  $j \leftarrow 0$ .
  - ii) Repeat
    - A)  $j \leftarrow j + 1$ .
    - B) Solve the problem  $(\tilde{P}_{i+j})$  and obtain an outer approximation  $OR_\delta^{(i+j)}$  of its region of  $\delta$ -optimality  $R_\delta^{(i+j)}$  using the procedure mentioned in Subsection 2.2.
    - C) Calculate  $f_2^{(i+j+1)}$  as given by (4) or (5).
 until  $f_2^{(i+j+1)} < f_2^{(i)}$  or the problem  $(\tilde{P}_{i+j})$  is infeasible.
  - iii) If  $f_2^{(i+j+1)} < f_2^{(i)}$  then set  $i \leftarrow i + j + 1$ .
  - iv) If the problem  $(\tilde{P}_{i+j})$  is infeasible then set  $i \leftarrow i + j$  and go to step 3.
- else  $i \leftarrow i + 1$ .

3.  $\bigcup_{j=0}^{i-1} OR_\delta^{(j)}$  contains the efficient set of (1).

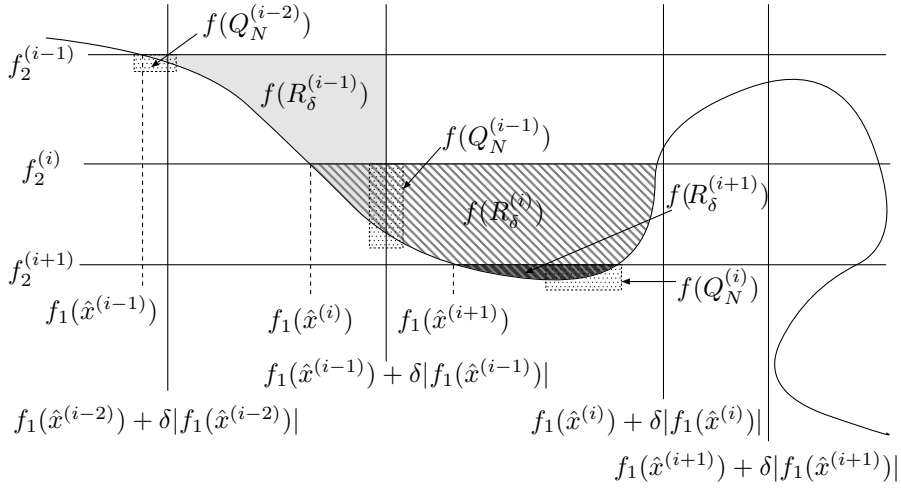


Figure 2. Image space of a biobjective problem using CLM. For the shake of clarity, for each problem  $(P_i)$  we have only drawn the image of one of the boxes forming  $OR_\delta^{(i)}$ , namely,  $f(Q_N^{(i)})$ . The light grey region is  $f(R_\delta^{(i-1)})$ , the striped region is  $f(R_\delta^{(i)})$ , and the dark grey region (that is totally covered by the striped one) is  $f(R_\delta^{(i+1)})$ .

*Theorem 4.* If CLM stops after a finite number of iterations, then it obtains all the efficient solutions of (1).

*Proof.* From Theorem 2, a necessary condition for a feasible vector to be efficient is that it must be a solution of a problem of the form

$$(P_{aux}) \quad \begin{array}{ll} \min & f_1(x) \\ \text{s.t.} & f_2(x) \leq f_2^{(aux)} \\ & x \in S \end{array}$$

We shall proof that all the optimal solutions of problems of that type lie in a region of  $\delta$ -optimality of one of the problems of type  $(P_i)$  or  $(\check{P}_{i+j})$  solved by CLM. Let  $\hat{x}^{(aux)}$  be an optimal solution of  $(P_{aux})$ .

Case A: let us suppose that  $f_2^{(i+1)} < f_2^{(aux)} < f_2^{(i)}$  (if  $f_2^{(i+1)} = f_2^{(aux)}$  then  $(P_{aux})$  coincides with  $(P_{i+1})$  and if  $f_2^{(i)} = f_2^{(aux)}$  then  $(P_{aux})$  coincides with  $(P_i)$ ; in either case, the result follows trivially). We shall proof that  $\hat{x}^{(aux)} \in R_\delta^{(i)}$ .

- Notice that  $f_1(\hat{x}^{(aux)}) \geq f_1(\hat{x}^{(i)})$ , since the feasible set of  $(P_i)$  contains the feasible set of  $(P_{aux})$  (remind that we are supposing that  $f_2^{(aux)} < f_2^{(i)}$ ).
- Analogously,  $f_1(\hat{x}^{(aux)}) \leq f_1(\hat{x}^{(i+1)})$ .
- On the other hand, from Lemma 1,  $f_1(\hat{x}^{(i+1)}) \leq f_1(\hat{x}^{(i)}) + \delta|f_1(\hat{x}^{(i)})|$ .

Thus,  $f_1(\hat{x}^{(i)}) \leq f_1(\hat{x}^{(aux)}) \leq f_1(\hat{x}^{(i)}) + \delta|f_1(\hat{x}^{(i)})|$ , that is,  $\hat{x}^{(aux)} \in R_\delta^{(i)}$ . The result follows from the fact that  $R_\delta^{(i)} \subseteq OR_\delta^{(i)}$ .

Notice that  $f_2^{(aux)} > f_2^{(0)}$  cannot hold, since  $f_2^{(0)} = +\infty$ . And if  $(P_{last})$  is the last problem solved by the algorithm, i.e.,  $(P_{last})$  is infeasible, then  $f_2^{(aux)} < f_2^{(last)}$  cannot hold either, since then  $(P_{aux})$  will be infeasible too. So, no efficient point is lost in this case by the algorithm.

Case B: let us suppose that  $f_2^{(i+1)} \geq f_2^{(i)}$ .

Subcase B.1: Consider the case in which  $f_2^{(i+1)} \geq f_2^{(i)} > f_2^{(aux)}$  holds and the algorithm has to solve  $k$  problems of type  $(\check{P}_{i+j})$ ,  $k \geq 1$ , before the condition  $f_2^{(aux)} > f_2^{(i+k+1)}$  holds. In this case  $f_1(\hat{x}^{(i)}) \leq f_1(\hat{x}^{(aux)}) \leq f_1(\hat{x}^{(i+k+1)})$  (notice that both  $(P_i)$  and  $(P_{i+k+1})$  are problems of the same type). Let us suppose that  $f_1(\hat{x}^{(i+r)}) < f_1(\hat{x}^{(aux)}) < f_1(\hat{x}^{(i+r+1)})$ ,  $1 \leq r \leq k$  (notice that the optimal objective values of the problems of type  $(\check{P}_{i+j})$  are nondecreasing; also, as before, we can assume strict inequalities). We shall proof that  $\hat{x}^{(aux)} \in R_\delta^{(i+r)}$ .

First, notice that  $\hat{x}^{(aux)}$  is a feasible point of  $(\check{P}_{i+r})$ : it is a point in  $S$ ,  $f_2(\hat{x}^{(aux)}) \leq f_2^{(aux)} < f_2^{(i)}$ , and  $f_1(\hat{x}^{(aux)}) \geq f_1(\hat{x}^{(i+r)}) \geq f_1(x^{(i)}) + r \cdot \delta |f_1(x^{(i)})|$ . And second, it is in  $R_\delta^{(i+r)}$ , since  $f_1(\hat{x}^{(i+r)}) \leq f_1(\hat{x}^{(aux)})$  and  $f_1(\hat{x}^{(aux)}) \leq f_1(\hat{x}^{(i+r)}) + \delta |f_1(\hat{x}^{(i+r)})|$ : if  $f_1(\hat{x}^{(aux)}) > f_1(\hat{x}^{(i+r)}) + \delta |f_1(\hat{x}^{(i+r)})|$ , since  $f_1(\hat{x}^{(i+r)}) + \delta |f_1(\hat{x}^{(i+r)})| \geq f_1(\hat{x}^{(i)}) + r \cdot \delta |f_1(\hat{x}^{(i)})| + \delta \cdot |f_1(\hat{x}^{(i)})| = f_1(\hat{x}^{(i)}) + (r+1) \cdot \delta |f_1(\hat{x}^{(i)})|$ , the point  $\hat{x}^{(aux)}$  will be feasible for the problem  $(\check{P}_{i+r+1})$ , but this is not possible, since  $f_1(\hat{x}^{(aux)}) < f_1(\hat{x}^{(i+r+1)})$  and  $\hat{x}^{(i+r+1)}$  is an optimal solution of  $(\check{P}_{i+r+1})$ .

Subcase B.2: Suppose now that  $f_2^{(i+1)} > f_2^{(aux)} > f_2^{(i)}$  (as before, we can suppose that the inequalities are strict). In this case,  $f_1(\hat{x}^{(aux)}) \leq f_1(\hat{x}^{(i)})$ , since the feasible set of  $(P_{aux})$  contains the feasible set of  $(P_i)$  (now we are supposing that  $f_2^{(aux)} > f_2^{(i)}$ ). Then, either there exists an index  $j$ ,  $j \leq i-1$ , such that  $f_2^{(j+1)} < f_2^{(aux)} < f_2^{(j)}$  (and then, as in case A,  $\hat{x}^{(aux)} \in R_\delta^{(j)}$ ) or there exists an index  $j$ ,  $j < i-1$ , such that  $f_2^{(j+1)} \geq f_2^{(j)} > f_2^{(aux)}$  holds and the algorithm has to solve  $k'$  problems of type  $(\check{P}_{i+j})$ ,  $k' \geq 1$ , before the condition  $f_2^{(aux)} > f_2^{(j+k'+1)}$  holds (and then, subcase B.1 applies). In either case,  $x^{(aux)}$  will be in the region of  $\delta$ -optimality of one of the problems solved by the algorithm.

Notice that we need problems of type  $(\check{P}_{i+j})$  to avoid that the algorithm gets stuck when  $f_2^{(i+1)} \geq f_2^{(i)}$ . This may happen when the nondominated set is not connected and the ‘jump’ (along the abscissas axis) is greater than  $\delta |f_1(\hat{x}^{(i)})|$  (see Figure 2, without problems of type  $(\check{P}_{i+j})$  CLM will get stuck after solving problem  $(P_{i+1})$ ) or when there is a continuum of weakly efficient points with the same  $f_2$ -value (i.e., in the image space the weakly efficient points form a segment parallel to the abscissas axis, being the length of that segment greater than  $\delta |f_1(\hat{x}^{(i)})|$ ).

On the other hand, in some pathological cases it may happen, at least theoretically, that the algorithm does not stop. For instance, we may have a biobjective problem in which  $f_2^{(i)} = \frac{1}{i}$  but in which the optimal value of the single objective problem

$$\begin{aligned} \min \quad & f_2(x) \\ \text{s.t.} \quad & x \in S \end{aligned} \tag{6}$$

is less than or equal to 0. In that case, CLM always uses problems of type  $(P_i)$ , but the algorithm cannot sweep the nondominated set till the bottom: it reaches at most till  $\lim_{i \rightarrow \infty} f_2^{(i)}$ .

### 3.2. USING A SINGLE TYPE OF CONSTRAINT PROBLEMS

In CLM, in addition to constraint problems of the form  $(P_i)$  we also needed problems of the form  $(\check{P}_{i+j})$  to be able to guarantee the finiteness of the algorithm. In this section we present a modification of CLM which only uses one type of constraint problems, denoted by  $(\bar{P}_i)$ , which is a simplified version of type  $(\check{P}_{i+j})$ , namely,

$$(\bar{P}_i) \quad \begin{array}{ll} \min & f_1(x) \\ \text{s.t.} & f_2(x) \leq f_2^{(i)} \\ & f_1(x) \geq f_1(\hat{x}^{(i-1)}) + \delta|f_1(\hat{x}^{(i-1)})| \\ & x \in S \end{array}$$

Here, again,  $\hat{x}^{(i-1)}$  denotes an optimal solution of  $(\bar{P}_{i-1})$

Similarly to the previous algorithm, the first problem that we will consider,  $(\bar{P}_0)$ , is again the single objective problem

$$\begin{array}{ll} \min & f_1(x) \\ \text{s.t.} & x \in S \end{array}$$

Once we have solved problem  $(\bar{P}_i)$  and have obtained a minimizer point  $\hat{x}^{(i)}$  and an outer approximation  $OR_\delta^{(i)}$  of its region of  $\delta$ -optimality  $R_\delta^{(i)}$  with the procedure mentioned in Subsection 2.2, we compute

$$f_2^{(i+1)} = \min\{f_2^{(i)}, \min\{\bar{F}_2(Q) : Q \in LIR_\delta^{(i)} \cup LOR_\delta^{(i)}, Q \cap R_\delta^{(i)} \neq \emptyset\}\} \quad (7)$$

or

$$f_2^{(i+1)} = \min\{f_2^{(i)}, \min\{\bar{F}_2(Q) : Q \in LIR_\delta^{(i)}, Q \cap Y^{(i)} \neq \emptyset\}\} \quad (8)$$

With these ingredients, the modified method for obtaining an outer approximation of the efficient set of (1) is the following (see Figure 3):

#### Modified constraint-like method (MCLM)

1.  $i \leftarrow 0$ .
2. While  $(\bar{P}_i)$  is feasible
  - a) Solve  $(\bar{P}_i)$  and obtain an outer approximation  $OR_\delta^{(i)}$  of its region of  $\delta$ -optimality  $R_\delta^{(i)}$  using the procedure mentioned in Subsection 2.2.

- b) Calculate  $f_2^{(i+1)}$  as given by (7) or (8).  
c)  $i \leftarrow i + 1$ .
3.  $\bigcup_{j=0}^{i-1} OR_\delta^{(j)}$  contains the efficient set of (1).

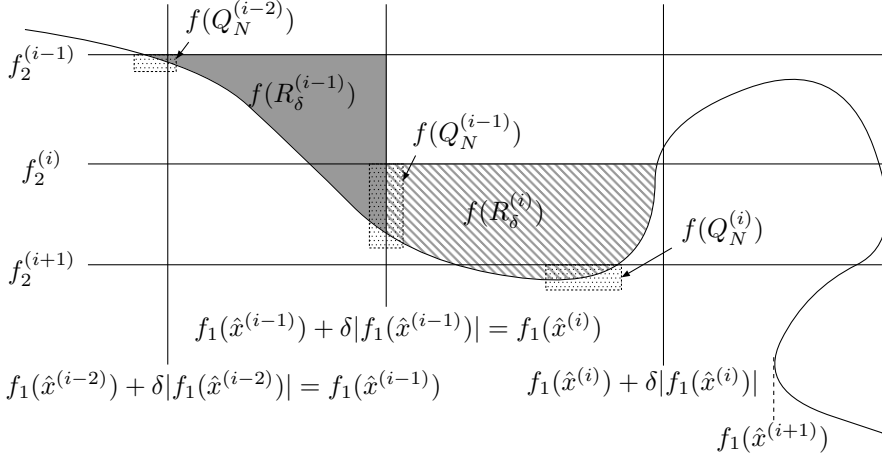


Figure 3. Image space of a biobjective problem using MCLM. The grey region is the image of  $R_\delta^{(i-1)}$ ,  $f(R_\delta^{(i-1)})$ , and the striped region is the image of  $R_\delta^{(i)}$ ,  $f(R_\delta^{(i)})$ .  $f(Q_N^{(i)})$  denotes the image of a subset  $Q_N^{(i)}$  at which the minimum (4) or (5) is attained, i.e., a subset such that  $f_2^{(i+1)} = \bar{F}_2(Q_N^{(i)})$ .

*Theorem 5.* Suppose that both the efficient set and the nondominated set of (1) are bounded. Suppose also that  $f_1(\hat{x}^{(0)}) > 0$ . Then MCLM obtains the complete efficient set of (1) in a finite number of steps.

*Proof.* From Theorem 2, a necessary condition for a feasible vector to be efficient is that it must be a solution of a problem of the form

$$(P_{aux}) \quad \begin{array}{ll} \min & f_1(x) \\ \text{s.t.} & f_2(x) \leq f_2^{(aux)} \\ & x \in S \end{array}$$

We shall prove that all the optimal solutions of problems of that type which are efficient lie in a region of  $\delta$ -optimality of one of the problems of type  $(\bar{P}_i)$  solved by MCLM. Let  $\hat{x}^{(aux)}$  be an optimal solution of  $(P_{aux})$ .

Notice that the sequence  $\{f_1(\hat{x}^{(i)})\}$  is nondecreasing. Thus, one of the following three cases must happen:

- A:  $f_1(\hat{x}^{(0)}) \leq f_1(\hat{x}^{(aux)}) \leq f_1(\hat{x}^{(0)}) + \delta|f_1(\hat{x}^{(0)})|$ . It means that  $\hat{x}^{(aux)}$  is in the region of  $\delta$ -optimality of  $(\bar{P}_0)$ .

B: There is an index  $i$ ,  $1 \leq i \leq last - 1$ , such that  $f_1(\hat{x}^{(i-1)}) + \delta|f_1(\hat{x}^{(i-1)})| \leq f_1(\hat{x}^{(aux)}) \leq f_1(\hat{x}^{(i)}) + \delta|f_1(\hat{x}^{(i)})|$  ( $last - 1$  is the index of the last feasible problem considered by the algorithm). Two subcases can happen:

B.1:  $f_2(\hat{x}^{(aux)}) \leq f_2^{(i)}$ . Then the point  $\hat{x}^{(aux)}$  is a feasible point of  $(\bar{P}_i)$ . In particular,  $f_1(\hat{x}^{(aux)}) \geq f_1(\hat{x}^{(i)})$ . Furthermore, by assumption,  $f_1(\hat{x}^{(aux)}) \leq f_1(\hat{x}^{(i)}) + \delta|f_1(\hat{x}^{(i)})|$ . Thus,  $\hat{x}^{(aux)}$  is in the region of  $\delta$ -optimality of  $(\bar{P}_i)$ .

B.2:  $f_2(\hat{x}^{(aux)}) > f_2^{(i)}$ . Then, there must exist a point  $\tilde{x} \in R_\delta^{(i-1)}$  such that  $f_2(\tilde{x}) \leq f_2^{(i)} < f_2(\hat{x}^{(aux)})$  and  $f_1(\tilde{x}) \leq f_1(\hat{x}^{(i-1)}) + \delta|f_1(\hat{x}^{(i-1)})| \leq f_1(\hat{x}^{(aux)})$ . That is,  $\tilde{x}$  dominates  $\hat{x}^{(aux)}$ , i.e.,  $\hat{x}^{(aux)}$  is not an efficient point for problem (1). Thus, in this case, we do not have to care about if  $\hat{x}^{(aux)}$  is in the region of  $\delta$ -optimality of one of the problems solved by the algorithm.

C:  $f_1(\hat{x}^{(aux)}) > f_1(\hat{x}^{(last-1)}) + \delta|f_1(\hat{x}^{(last-1)})|$ . Two cases can happen:

C.1: If  $f_2(\hat{x}^{(aux)}) > f_2^{(last)}$  then  $\hat{x}^{(aux)}$  will not be an efficient point (similarly to subcase B.2).

C.2: If  $f_2(\hat{x}^{(aux)}) \leq f_2^{(last)}$  then  $\hat{x}^{(aux)}$  will be a feasible point of  $(\bar{P}_{last})$ , but this is a contradiction, since  $(\bar{P}_{last})$  is infeasible (it was the problem provoking the termination of the algorithm).

Notice that  $f_1(\hat{x}^{(aux)}) < f_1(\hat{x}^{(0)})$  cannot happen, since the feasible set of  $(\bar{P}_0)$  contains the feasible set of  $(P_{aux})$ .

Thus, in any case, if  $\hat{x}^{(aux)}$  is an efficient point, then it lies in the region of  $\delta$ -optimality of one of the problems solved by the algorithm.

To prove that the algorithm stops after a finite number of steps, just notice that with problem  $(\bar{P}_i)$  we sweep  $\delta|f_1(\hat{x}^{(i)})|$  units length along the  $f_1$  axis on  $Z_N$  in the criterion space. Thus, if we denote by  $\hat{y}^{(0)}$  an optimal solution of problem (6) we need at most

$$\frac{f_1(\hat{y}^{(0)}) - f_1(\hat{x}^{(0)})}{\delta f_1(\hat{x}^{(0)})}$$

problems to sweep to whole nondominated set.

The condition  $f_1(\hat{x}^{(0)}) > 0$  in Theorem 5 is not restrictive. If a function  $f_1$  does not satisfy it, we can use instead, for instance, the function  $\hat{f}_1(x) = f_1(x) + |f_1(\hat{x}^{(0)})| + 1$ .

## 3.3. ADDITIONAL REMARKS

The constraint-like method obtains a superset of the efficient set  $S_E$  which maps into a superset of the nondominated set  $Z_N$ , which may be made as tight as required (up to the precision provided by the inclusion functions used) by reducing the value of  $\delta$  (and the tolerances used in the procedure obtaining the regions of  $\delta$ -optimality): the smaller the value of  $\delta$ , the better the quality of the approximation, but also the higher the number of subproblems to be solved.

Also, with the same  $\delta$ , the quality of the approximation can vary depending on the shape of the nondominated set. See for instance Figure 4. For the ease of follow, in that figure, it is supposed that the shape of the nondominated set is linear and that the regions of  $\delta$ -optimality (triangles in the example) do not overlap. Whereas in Figure 4(a) the overestimation provided by the algorithm has an area of 2 units, in Figure 4(b) the overestimation is only 1 units.

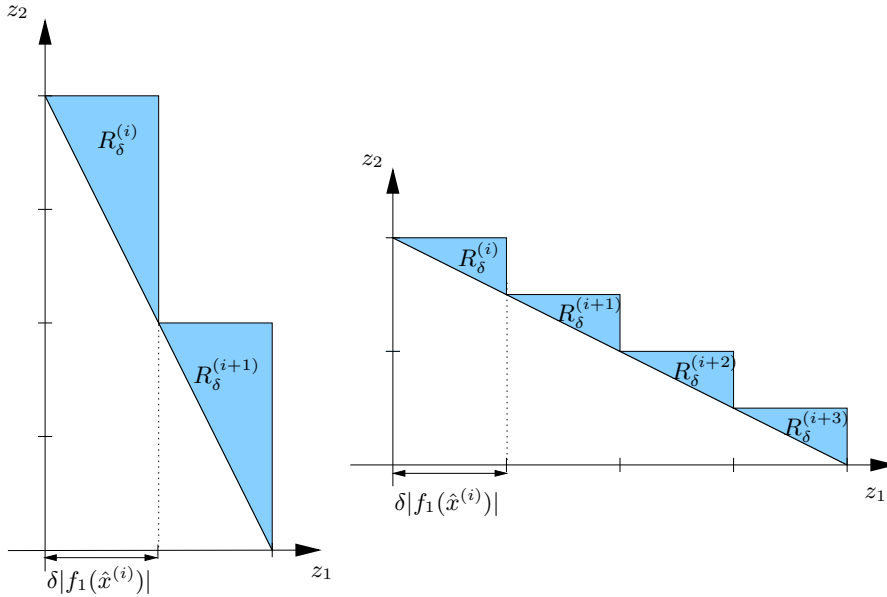


Figure 4. In (a) the overestimation is  $2 \cdot 1 = 2$  units and in (b) only  $4 \cdot 0.25 = 1$ .

But the quality can also vary depending on the objective function selected to be minimized in the constrained problems. Consider for instance again Figure 4, but supposing now that Figure 4(a) corresponds to the image space of a part of the nondominated set when the  $f_1$  values are plotted in the abscissas axis and the  $f_2$  values in the ordinate axis and Figure 4(b) when the  $f_1$  values are plotted in the ordinate axis and the  $f_2$  values in the abscissas axis. We can see clearly that in this

example we obtain a better approximation of the nondominated set by minimizing  $f_2$  in the constrained problems. As a rule, we could say that in the constraints problems it is better to select as objective function to be minimized the one whose range in the nondominated set is bigger.

Another way to improve the quality of the outer approximation of the efficient set is to remove from the regions of  $\delta$ -optimality  $OR_\delta^{(i)}$  all the subsets  $Q$  for which we can guarantee that do not contain any efficient point (see for instance the multi-objective cut-off test described in the next section).

Both CLM and MCLM generate the nondominated set from top-left to bottom-right. If we give to the decision-maker the regions of  $\delta$ -optimality as we compute them, then it may happen that he/she gets satisfied with one of the efficient points already found during the execution of the algorithm. In that case, the process can be stopped prematurely, and we can save the time of computing the rest of the efficient set. This can be seen as an interactive method.

*Remark 1.* The constraint-like method obtains a superset of the efficient set  $S_E$  which maps into a superset of the nondominated set  $Z_N$ , which may be made as tight as required by reducing the value of  $\delta$  (and the tolerances used in the procedure obtaining the regions of  $\delta$ -optimality): the smaller the value of  $\delta$ , the better the quality of the approximation, but also the higher the number of subproblems to be solved.

*Remark 2.* Problems  $(\tilde{P}_{i+j})$  and  $(\bar{P}_i)$  can be rewritten as

$$\begin{aligned} & \min f_1(x) \\ (\tilde{P}_i) \quad & \text{s.t. } f_2(x) \leq f_2^{(i)} \\ & x \in S \setminus \cup_{j=1}^{i-1} OR_\delta^{(j)} \end{aligned}$$

where instead of the constraint on  $f_1$ , we remove from  $S$  the outer approximations of the regions of  $\delta$ -optimality of the problems already solved. In fact, if we use this kind of problem instead of problems  $(P_i)$  as well, the two methods become equivalent.

*Remark 3.* Notice that although the method that we have used to obtain the whole set  $R_\delta$  when deriving (M)CLM is inspired by the method in [40], any other method which obtains the *complete* set  $R_\delta$  (or an outer approximation of it) could serve. If we denote by  $NOR_\delta$  the outer approximation (or exact representation) of  $R_\delta$  obtained by any other method, the only thing that has to be changed when using it in (M)CLM is the computation of the bounds  $f_2^{(i)}$ . If we denote by  $q$

any point in  $NOR_\delta$ , the bounds should be computed as

$$f_2^{(i+1)} = \min\{f_2^{(i)}, \min\{f_2(q) : q \in NOR_\delta^{(i)} \cap R_\delta^{(i)}\}\}.$$

In fact, (4) and (5) are surrogates for the previous formula.

#### 4. Carrying out the method: an interval implementation

Since the purpose of this paper is to deal with general nonlinear biobjective problems, the constraint problems that we have to solve may be global optimization ones (they may have many local optima). Thus, we need to use global optimization techniques to cope with them (the books [27, 28] are excellent introductions to the field of global optimization). Furthermore, instead of merely solving the constraint problems, we must also obtain their region of  $\delta$ -optimality. Among the global optimization techniques only a branch-and-bound scheme seems to be appropriate for our purposes, although the computation of bounds is a difficult task, too. In this paper we present such a method, which has some similarities with the two-phase method mentioned in Subsection 2.2 and described in [40], although modified for our purposes and in a more general framework: *Interval Analysis*, an analysis similar to the classical Real Analysis, but in which real numbers are replaced by compact intervals.

In this section, we first briefly summarize the fundamental concepts of interval analysis which are needed for this paper. For more details on the topic, the interested reader is referred to [25, 33, 42]. Then we describe the method proposed in more detail.

##### 4.1. INTERVAL ANALYSIS

In what follows, boldface will denote intervals, lower case will be used for scalar quantities or vectors (vectors are then distinguished from components by use of subscripts), and upper case for matrices. Brackets  $[\cdot]$  will delimit intervals, while parentheses  $(\cdot)$  indicate vectors and matrices. Underlines will denote lower bounds of intervals and overlines give upper bounds of intervals. For example, we may have the interval vector  $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^T$ , where  $\mathbf{x}_i = [\underline{x}_i, \overline{x}_i]$ . The *width* of an interval  $\mathbf{x}_i$  is denoted by  $w(\mathbf{x}_i) = \overline{x}_i - \underline{x}_i$  whereas the width of an interval vector  $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^T$  is to be understood as  $w(\mathbf{x}) = \max\{w(\mathbf{x}_i) : i = 1, \dots, n\}$ . The midpoint of  $\mathbf{x}$  will be denoted by  $\text{mid } \mathbf{x}$ . The set of intervals will be denoted by  $\mathbb{IR}$ , and the set of  $n$ -dimensional interval vectors, also called *boxes*, by  $\mathbb{IR}^n$ .

The *interval arithmetic operations* are defined by

$$\mathbf{x} * \mathbf{y} = \{x * y : x \in \mathbf{x}, y \in \mathbf{y}\} \text{ for } \mathbf{x}, \mathbf{y} \in \mathbb{IR}, \quad (9)$$

where the symbol  $*$  stands for  $+$ ,  $-$ ,  $\cdot$  and  $/$ , and where  $\mathbf{x}/\mathbf{y}$  is only defined if  $0 \notin \mathbf{y}$ . Definition (9) is equivalent to simple constructive rules (see [25, 33, 42]). The algebraic properties of (9) are different from those of real arithmetic operations (for instance, the subtraction and division in  $\mathbb{IR}$  are not the inverse operations of addition and multiplication, respectively), but the main properties from the operational point of view still hold, as for instance the so-called *inclusion isotonicity*,

$$\mathbf{x} \subseteq \mathbf{y}, \mathbf{z} \subseteq \mathbf{t} \implies \mathbf{x} * \mathbf{z} \subseteq \mathbf{y} * \mathbf{t} \text{ (if } \mathbf{y} * \mathbf{t} \text{ is defined) for } \mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{t} \in \mathbb{IR}.$$

which is implicitly used in the construction of *inclusion functions*, which are the main interval arithmetic tool applied to optimization methods.

*Definition 4.* A function  $\mathbf{h} : \mathbb{IR}^n \rightarrow \mathbb{IR}$  is said to be an *inclusion function* of  $h : \mathbb{R}^n \rightarrow \mathbb{R}$  provided

$$\{h(\mathbf{y}) : \mathbf{y} \in \mathbf{y}\} \subseteq \mathbf{h}(\mathbf{y})$$

for all boxes  $\mathbf{y} \subset \mathbb{IR}^n$  within the domain of  $h$ .

Observe that if  $\mathbf{h}$  is an inclusion function for  $h$  then we can directly obtain lower bounds and upper bounds of  $h$  over any box  $\mathbf{y}$  within the domain of  $h$  just by taking  $\underline{\mathbf{h}}(\mathbf{y})$  and  $\overline{\mathbf{h}}(\mathbf{y})$ , respectively. This means that with the use of inclusion functions, obtaining bounds is an automatic task.

For a function  $h$  predeclared in some programming language (like  $\sin$ ,  $\exp$ , etc.), it is not too difficult to obtain a *predeclared* inclusion function  $\mathbf{h}$ , since the monotonicity intervals of predeclared functions are well known and then we can take  $\mathbf{h}(\mathbf{y}) = \{h(\mathbf{y}) : \mathbf{y} \in \mathbf{y}\}$  for any  $\mathbf{y} \in \mathbb{IR}$  in the domain of  $h$ . For a general function  $h(\mathbf{y})$ ,  $\mathbf{y} \in \mathbb{R}^n$ , several methods can be employed to obtain inclusion functions although how to find an inclusion function as good as possible, that is, producing bounds as tight as possible, is still an open question (see [48, 49]). The easiest method to obtain an inclusion function is the *natural interval extension*, which is obtained by replacing each occurrence of the variable  $y$  with a box including it,  $\mathbf{y}$ , each occurrence of a predeclared function by its corresponding inclusion function, and the real arithmetic operators by the corresponding interval operators. Another method to evaluate inclusion functions when  $h(\mathbf{y})$  is differentiable is the *centered form*, given by  $\mathbf{h}(\mathbf{y}) = \mathbf{h}(c) + (\mathbf{y} - c)^T \nabla \mathbf{h}(\mathbf{y})$  where  $c$  is any point of  $\mathbf{y}$  (usually its midpoint) and  $\nabla \mathbf{h}(\mathbf{y})$  an inclusion function of the gradient

$\nabla h$  of  $h$  at  $\mathbf{y}$  (usually obtained as the natural interval extension of  $\nabla h$ ). The centered form usually gives over small boxes tighter bounds as compared to the natural interval extension. However, the natural interval extension is often very useful because of its computational simplicity.

#### 4.2. THE ALGORITHM

The implementation of CLM is described in pseudo-code form in Algorithm 1 (the code is available upon request from the authors). We have considered the constraint problems to be written in the form of  $(\tilde{P}_i)$ . In this way, we only have to deal with the constraint on  $f_2$ , and not with the one on  $f_1$  (the removal of the regions of  $\delta$ -optimality of the previous problems is done easily in our implementation, see Step 35 of Phase 2 in Algorithm 2).

In Algorithm 1,  $\mathbf{s}$  denotes a box containing the feasible set  $S$  and  $\max f_j$  is the maximum  $f_j$ -value that any efficient point can take,  $j = 1, 2$ , that is,  $(\max f_1, \max f_2)$  is the *nadir point* of (1). As we can see, to calculate  $\max f_1$  we first solve the problem (6) to optimality by calling Phase 1 of Algorithm 2 (see Step 3) with  $\delta = 0$  and using  $f_2$  instead of  $f_1$ , and then we compute  $\max f_1 = \max\{\overline{\mathbf{f}}_1(\mathbf{y}) : \mathbf{y} \in L_2\}$ , where  $L_2$  is the solution list of Phase 1. Something similar is done to calculate  $\max f_2$  (Step 5), although this time we do not modify the value of  $\delta$ , since  $(P_0)$  is the first problem for which we have to compute its region of  $\delta$ -optimality. After that, the algorithm (Steps 6 to 9) keeps calling Algorithm 2, the main procedure which obtains the regions of  $\delta$ -optimality of the constraint problems, until one of the problems is infeasible.

---

#### Algorithm 1 Constraint-Like Method

---

**Input:**  $\mathbf{f}_1, \mathbf{f}_2, \mathbf{h}_l (l = 1, \dots, r), \mathbf{s}, \delta, \epsilon, \eta, \mu$

**Output:**  $L_{sol}$

- 1  $\max f_1 = \max f_2 = \infty$ ;
  - 2 Eval  $\mathbf{f}_2(\mathbf{s})$ ;  $\tilde{\mathbf{f}} = \overline{\mathbf{f}}_2(\mathbf{s})$ ;  $\mathbf{s} \rightarrow L_1$ ;
  - 3 Call *Phase 1* using  $f_2$  as  $f_1$ , and  $\delta = 0$ ;  $\max f_1 = \max\{\overline{\mathbf{f}}_1(\mathbf{y}) : \mathbf{y} \in L_2\}$ ;
  - 4 Eval  $\mathbf{f}_1(\mathbf{s})$ ;  $\tilde{\mathbf{f}} = \overline{\mathbf{f}}_1(\mathbf{s})$ ;  $\mathbf{s} \rightarrow L_1$ ;
  - 5 Call *Phase 1*;  $\max f_2 = \max\{\overline{\mathbf{f}}_2(\mathbf{y}) : \mathbf{y} \in L_2\}$ ;
  - 6 **repeat**
  - 7     *Phase 2*
  - 8     *Phase 1*
  - 9 **until** *Phase 1* terminates with no solution.
- 

Algorithm 2 is the core of Algorithm 1. It allows to obtain the outer approximation of the region of  $\delta$ -optimality of the constraint problems.

The algorithm is inspired by the method in [40]. It consist of two phases. The aim of the first phase is to obtain the optimal value of the constraint problem (within a tolerance  $\epsilon$ ), whereas the aim of the second is to obtain its region of  $\delta$ -optimality  $R_\delta^{(i)}$  (within a tolerance  $\eta$ ).

However, Algorithm 2 differs from the method in [40] in several points. First, the bounds are computed here with the help of interval analysis. Second, in addition to the ‘feasibility test’ (with two implementations, Feasible and Infeasible, in the code) and the ‘ $\delta$ -cut-off test’ (CutOffTest in the code, a modification of the classical cut-off test [42], also used in [40]), we also use a variant of the ‘ $\delta$ -monotonicity test’ for strictly feasible and undetermined boxes (MonoTest) [17], a ‘pruning test’ [17] (Prune in the code) applied to  $f_2$  or to both  $f_1$  and  $f_2$ , and a multi-objective cut-off test (which discards dominated boxes) [18].

A brief description of the discarding tests used follows.

**Feasibility Test:** Let us suppose that  $S$  is given by  $S = \{y \in \mathbb{R}^n : h_l(y) \leq 0, l = 1, \dots, r\}$ . We say that a box  $\mathbf{y}$  *certainly satisfies* the constraint  $h_l(y) \leq 0$  if  $\bar{h}_l(\mathbf{y}) \leq 0$  and that  $\mathbf{y}$  *does certainly not satisfy* it if  $\underline{h}_l(\mathbf{y}) > 0$ . A box  $\mathbf{y} \subseteq \mathbf{s}$  is said *certainly feasible* if it certainly satisfies all the constraints, *certainly infeasible* if it does certainly not satisfy at least one of the constraints, and *undetermined* otherwise. The ‘Infeasible( $\mathbf{y}$ )’ test is true when the box  $\mathbf{y}$  is certainly infeasible, whereas the ‘Feasible( $\mathbf{y}$ )’ test is true when the box  $\mathbf{y}$  is certainly feasible.

**$\delta$ -Cut-off Test:** Every time a box  $\mathbf{y}$  is chosen from the list  $L_1$ , and provided that its midpoint  $c$  is certainly feasible, we use  $\bar{\mathbf{f}}_1(c)$  (previously computed to evaluate the centered form) to update (if possible, i.e. when  $\bar{\mathbf{f}}_1(c) < \tilde{f}$ ) the best upper bound  $\tilde{f}$  of the global minimum of the constraint problem (but see also the multi-objective cut-off test). If updated, then the ‘CutOffTest( $c, L_1 \cup L_2, L_3, \delta$ )’ sends to  $L_3$  all the boxes  $\mathbf{y}$  in  $L_1$  and  $L_2$  such that  $\underline{\mathbf{f}}_1(\mathbf{y}) > \tilde{f}$  (since they cannot contain the optimal value of the constraint problem), and then sends to  $L_4$  all the boxes  $\mathbf{y}$  in  $L_3$  such that  $\underline{\mathbf{f}}_1(\mathbf{y}) > \tilde{f}(1 + \delta)$  (since they cannot be in  $R_\delta^{(i)}$ ).

**Pruning test applied to  $f_2$ :** The pruning test was recently proposed in [37], and modified in [16]. It uses gradient information to determine regions in the actual box which cannot contain global optimizers. We briefly explain it using, not to complicate matters, a two-dimensional minimization problem, and we describe how to apply it along the  $y_1$ -direction. Consider a box  $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2)$ , and

**Algorithm 2** Main procedure*Phase 1*


---

**Input:**  $\mathbf{f}_1, \mathbf{f}_2, \mathbf{h}_l, L_1, LPES, \delta, \epsilon, \max f_1, \max f_2$   
**Output:**  $L_2, L_3, L_4, LPES, \tilde{f}$

```

1 while (  $L_1 \neq \emptyset$  ) do
2    $L_1 \rightarrow \mathbf{y}; c \rightarrow \text{mid}(\mathbf{y})$ 
3   Eval CenteredForm( $f_1, \mathbf{y}, c$ )
4   Eval CenteredForm( $f_2, \mathbf{y}, c$ )
5   if ( $\underline{f}_1(\mathbf{y}) > \max f_1$  ||  $\underline{f}_2(\mathbf{y}) > \max f_2$ )
6     continue
7   if Infeasible( $\mathbf{y}$ )
8     continue
9   if MultiCutOffTest( $\mathbf{y}$ )
10    continue
11  CutOffTest( $c, L_1 \cup L_2, L_3, \tilde{f}, \delta$ );
12  if (  $\underline{f}_1(\mathbf{y}) > \tilde{f}$  )
13    if (  $\underline{f}_1(\mathbf{y}) \leq \tilde{f} * (1 + \delta)$  )
14       $\mathbf{y} \rightarrow L_3$ ; continue
15    else
16       $\mathbf{y} \rightarrow L_4$ ; continue
17  if MonoTest( $\mathbf{y}, \mathbf{z}$ )
18     $\mathbf{y} \rightarrow L_3$ 
19     $\mathbf{z} \rightarrow \mathbf{y}$ 
20  if ( $\underline{f}_1(\mathbf{y}) + \epsilon |\underline{f}_1(\mathbf{y})| \geq \tilde{f}$  ||  $w(\mathbf{y}) < \epsilon$ )
21     $\mathbf{y} \rightarrow L_2$ 
22  else
23    Prune( $\mathbf{y}, \tilde{f}, \epsilon, \delta$ )  $\rightarrow \mathbf{y}_1, \mathbf{y}_2$ 
24  for  $i = 1, 2$  do
25     $\mathbf{f}_2(\mathbf{y}_i) \cap = \mathbf{f}_2(c) + (\mathbf{y}_i - c)^T \nabla \mathbf{f}_2(\mathbf{y})$ 
26    if Infeasible( $\mathbf{y}_i$ )
27      continue
28    Eval  $\mathbf{f}_1(\mathbf{y}_i)$ 
29     $\mathbf{f}_1(\mathbf{y}_i) \cap = \mathbf{f}_1(c) + (\mathbf{y}_i - c)^T \nabla \mathbf{f}_1(\mathbf{y})$ 
30    if MultiCutOffTest( $\mathbf{y}_i$ )
31      continue
32    if (  $\underline{f}_1(\mathbf{y}_i) > \tilde{f}$  )
33      if (  $\underline{f}_1(\mathbf{y}_i) \leq \tilde{f} * (1 + \delta)$  )
34         $\mathbf{y}_i \rightarrow L_3$ ; continue
35      else
36         $\mathbf{y}_i \rightarrow L_4$ ; continue
37    if ( $\underline{f}_1(\mathbf{y}_i) > \max f_1$  ||  $\underline{f}_2(\mathbf{y}_i) > \max f_2$ )
38      continue
39     $\mathbf{y}_i \rightarrow L_1$ 
40  endfor
41 endwhile
```

*Phase 2*


---

**Input:**  $\mathbf{f}_1, \mathbf{f}_2, \mathbf{h}_l, L_2, L_3, L_4, LPES, \tilde{f}, \delta, \epsilon, \eta, \mu, \max f_1, \max f_2$   
**Output:**  $L_1, L_{sol}, LPES$

```

1 while (  $L_3 \neq \emptyset$  ) do
2    $L_3 \rightarrow \mathbf{y}; c \rightarrow \text{mid}(\mathbf{y})$ 
3   Eval CenteredForm( $f_1, \mathbf{y}, c$ )
4   Eval CenteredForm( $f_2, \mathbf{y}, c$ )
5   if ( $\underline{f}_1(\mathbf{y}) > \max f_1$  ||  $\underline{f}_2(\mathbf{y}) > \max f_2$ )
6     continue
7   if Infeasible( $\mathbf{y}$ )
8     continue
9   if MultiCutOffTest( $\mathbf{y}$ )
10    continue
11  if (  $\underline{f}_1(\mathbf{y}) > \tilde{f} * (1 + \delta)$  )
12     $\mathbf{y} \rightarrow L_4$ ; continue
13  if (Feasible( $c$ ) &  $\overline{f}_2(c) < f_2^{(i)}$  &  $\overline{f}_1(c) < \tilde{f} * (1 + \delta)$ )
14     $f_2^{(i)} = \overline{f}_2(c)$ 
15  if ( $\overline{f}_1(\mathbf{y}) < \tilde{f} * (1 + \delta)(1 + \eta)$  & (Feasible( $\mathbf{y}$ ) ||  $w(\mathbf{f}_2(\mathbf{y})) < \mu$  ||  $w(\mathbf{y}) < \epsilon$  ) )
16     $\mathbf{y}_i \rightarrow L_2$ ; continue
17  Prune( $\mathbf{y}, \tilde{f}, \epsilon, \delta$ )  $\rightarrow \mathbf{y}_1, \mathbf{y}_2$ 
18  for  $i = 1, 2$  do
19     $\mathbf{f}_2(\mathbf{y}_i) \cap = \mathbf{f}_2(c) + (\mathbf{y}_i - c)^T \nabla \mathbf{f}_2(\mathbf{y})$ 
20    if Infeasible( $\mathbf{y}_i$ )
21      continue
22    Eval  $\mathbf{f}_1(\mathbf{y}_i)$ 
23     $\mathbf{f}_1(\mathbf{y}_i) \cap = \mathbf{f}_1(c) + (\mathbf{y}_i - c)^T \nabla \mathbf{f}_1(\mathbf{y})$ 
24    if MultiCutOffTest( $\mathbf{y}_i$ )
25      continue
26    if (  $\underline{f}_1(\mathbf{y}_i) \leq \tilde{f} * (1 + \delta)$  )
27       $\mathbf{y}_i \rightarrow L_3$ ; continue
28    else
29       $\mathbf{y}_i \rightarrow L_4$ ; continue
30    if ( $\underline{f}_1(\mathbf{y}_i) > \max f_1$  ||  $\underline{f}_2(\mathbf{y}_i) > \max f_2$ )
31      continue
32     $\mathbf{y}_i \rightarrow L_3$ 
33  endfor
34 endwhile
35  $L_2 \rightarrow L_{sol}; L_4 \rightarrow L_1$ 
36 for  $z^* \in LPES$  do
37   if  $z_2^* > f_2^{(i)}$ 
38     Remove  $z^*$  from  $LPES$ 
```

---

suppose that we know a lower bound for the value of the objective function  $h$  at  $(\text{mid } \mathbf{y}_1, \mathbf{y}_2)$ , and also bounds for the gradient  $\nabla h(\mathbf{y})$  (see Figure 5). Then a lower bounding function of  $h$  can

be constructed as the planes in Figure 5 (similarly to what is commonly done in Lipschitz optimization [26]). Then, using an upper bounding value  $\tilde{h}$ , the minimizer points in  $\mathbf{y}$  can lie only in  $\mathbf{x}$  and  $\mathbf{z}$ .

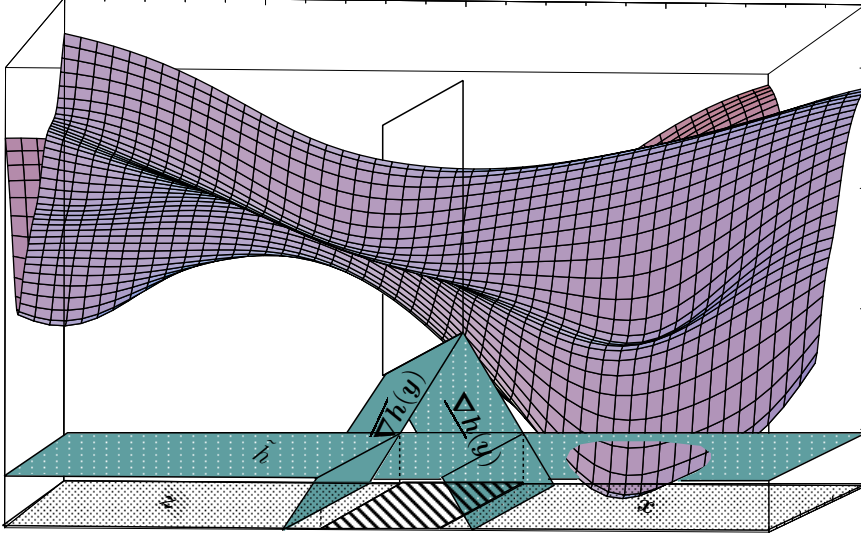


Figure 5. Pruning method using the gradient.

In particular, when applied to  $f_2$  using  $f_2^{(i)}$  as upper bounding value, it means that the feasible points in  $\mathbf{y}$  can only lie in  $\mathbf{x}$  or  $\mathbf{z}$ . Therefore, the other parts of  $\mathbf{y}$  are not of interest neither for the current constraint problem nor for the rest of constraint problems (since in the following constraint problems the upper bound on  $f_2$  will be smaller than or equal to  $f_2^{(i)}$ ) and can be deleted.

This pruning test can be done for any coordinate direction of the box, generating one or two new subboxes. In this sense, it can be seen as a bisection method along the chosen coordinate. However, a coordinate  $j$  is selected only if  $w(\mathbf{y}_j) > \epsilon$ . In particular, we choose the coordinate direction  $j$  such that the corresponding part removed from  $\mathbf{y}$  is the largest one.

**Pruning Test applied to  $f_1$  and  $f_2$ :** A similar process to the one described in the above pruning test can be done applying it to function  $f_1$  considering  $\tilde{f}(1+\delta)$  as upper bounding value. However, now, the parts of  $\mathbf{y}$  which are not of interest (because their  $f_1$  value

is greater than  $\tilde{f}(1 + \delta)$ ) have to be sent to  $L_4$ , because they may be of interest for the next constraint problems.

In order to apply the pruning technique to both  $f_1$  and  $f_2$  within the same algorithm without generating too many boxes, we have used the strategy which we explain with the following example (see Figure 6). Let us suppose that the dotted region A can be deleted

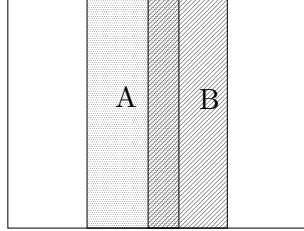


Figure 6. Pruning applied to  $f_1$  and  $f_2$ .

by pruning with  $f_2$ , while the striped area B can be cut by pruning with  $f_1$ . It is easy to see that cutting the region A is always a good decision, because we generate at most two new subboxes, and the deleted regions do not have to be taken into consideration any more (they are not of interest for any of the remaining problems). However the region  $B \setminus A$  has to be sent to  $L_4$ , if we decide to cut it. That is why we only cut it if its area is greater than 10% of the area of the original box. Furthermore, when  $B \setminus A$  is not connected, only its greater part is sent to  $L_4$  (provided that the previous condition holds). The selection of the coordinate direction to apply the test is done as in the previous test.

**$\delta$ -monotonicity test (for strictly feasible and undetermined boxes):** In [18] a monotonicity test (for strictly feasible and undetermined boxes) is described to decide whether the objective function  $f_1$  is strictly monotonous in a box, which allows either to discard the box or to reduce it to one of its facets. Since we are now computing  $R_\delta^{(i)}$  and not solving the constraint problem till optimality, we cannot discard monotonous boxes. Instead, when  $\text{MonoTest}(\mathbf{y}, \mathbf{z})$  is true, i.e., when the objective function is monotonous at  $\mathbf{y}$  ( $0 \notin \nabla f_1(\mathbf{y})$ ), the  $\delta$ -monotonicity test sends the box  $\mathbf{y}$  to  $L_3$  (since it cannot contain the optimal value of the constraint problem), and follows the process with the facet  $\mathbf{z}$  of  $\mathbf{y}$  containing the best points:  $\mathbf{z}$  cannot be discarded since it can lie on the border of the feasible set of the constraint problem). Note that this test is only useful in *Phase 1*.

**Multi-objective cut-off test:** This test was introduced in [18].

Every time  $\text{MultiCutOffTest}(\mathbf{y})$  is applied to a box  $\mathbf{y}$ , and provided that its midpoint  $c$  (as a point interval) is certainly feasible, we compute  $\bar{\mathbf{f}}(c) = (\bar{\mathbf{f}}_1(c), \bar{\mathbf{f}}_2(c))$  and update (if possible, i.e., if  $\bar{\mathbf{f}}(c)$  is not dominated by any point in  $L_{PES}$ ) the list  $L_{PES}$  of ‘provisional’ efficient solutions available so far. The test discards boxes whose points are not efficient, i.e., a box  $\mathbf{y}$  is removed (and then the test is true) if  $\underline{\mathbf{f}}(\mathbf{y}) > z^*$  for some  $z^* \in L_{PES}$ . When this test and the  $\delta$ -cut-off test are used together in the same algorithm, then  $\tilde{\mathbf{f}}$  is updated in the  $\delta$ -cut-off test only if  $\bar{\mathbf{f}}(c)$  is not dominated by any point in  $L_{PES}$ . This is to avoid the possibility that the list  $L_1$  of a constraint problem becomes empty, what may happen in some cases in which all its boxes are deleted by the multi-objective cut-off test or the  $\delta$ -cut-off test.

A few comments on the Algorithm 2 are in order. The box  $\mathbf{y}$  to be chosen from  $L_1$  (Step 2 of Phase 1) and  $L_3$  (Step 2 of Phase 2) is the one with the lowest  $\underline{\mathbf{f}}_1(\mathbf{y})$ . Notice that in some steps we compute the centered form as described in the previous subsection, whereas in the Steps 25 and 29 of Phase 1 and in Steps 19 and 23 of Phase 2 we use the inclusion of the gradient of the mother box (to save time). In steps 37 to 39 of Phase 2 we remove from  $L_{PES}$  the points  $z^* = (z_1^*, z_2^*)$  for which  $z_2^* > f_2^{(i)}$  holds, since those points will not discard any box in the next constraint problems.

In Step 20 of Phase 1 (see also Step 15 of Phase 2) we send a box to  $L_2$  if its width is less than a given tolerance. We have added this stopping rule because if the tolerances are chosen too small then, due to the overestimation of the inclusion functions, it may happen that the other stopping criteria are never fulfilled. On the other hand, in Step 15 of Phase 2, for sending a box  $\mathbf{y}$  to  $L_2$ , in addition to the condition  $\bar{\mathbf{f}}_1(\mathbf{y}) < \tilde{f}(1+\delta)(1+\eta)$  we also require the box  $\mathbf{y}$  either to be certainly feasible or to satisfy either  $w(\mathbf{f}_2(\mathbf{y})) < \mu$  ( $\mu$  is a given tolerance) or  $w(\mathbf{y}) < \epsilon$ . This is to avoid to send to  $L_2$  big undetermined boxes.

We also want to point out that if we remove the pruning tests (and we perform instead a simple bisection perpendicularly to the direction of maximum width), the  $\delta$ -monotonicity test and the multi-objective cut-off test we have an interval version of the GBSSS method [40].

The resulting solution list  $L_{sol}$  of Algorithm 1 is the desired list of boxes containing the complete efficient set of (1).

## 5. Computational experiments

### 5.1. COMPETITIVE LOCATION PROBLEMS

In our computational studies, we have used test problems which correspond to a location model. Location science deals with the location of one or more facilities in a way that optimizes a certain objective (minimization of transportation costs, minimization of social costs, maximization of the market share, etc.) See [7, 8, 20, 36] for an introduction to the topic. In fact, in many situations more than one objective has to be taken into account, thus leading to multiobjective location problems. See [39] for a survey of the most representative multicriteria location problems considered in the literature.

In particular, here we consider a biobjective competitive facility location problem introduced in [18]. Competitive location deals with the problem of locating facilities to provide a service (or goods) to the customers (or consumers) of a given geographical area where other competing facilities offering the same service are already present (or will enter to the market in the near future). Many competitive location models are available in the literature, see for instance the survey papers [14, 30, 41] and the references therein. However, the literature on multiobjective competitive location models is rather scarce. In fact, to our knowledge, [17, 18, 22] seem to be the only references in this field. This is in part due to the fact that single objective competitive location problems are difficult to solve, and considering more than one objective makes the problem near intractable.

We study the case of a franchise which wants to enlarge its presence in a given geographical region by opening one new facility. Both the franchisor (the owner of the franchise) and the franchisee (the actual owner of the new facility to be opened) have the same objective: maximize their own profit. However, the maximization of the profit obtained by the franchisor is in conflict with the maximization of the profit obtained by the franchisee. This suggests to use a biobjective model to obtain the efficient solutions for this problem, so that later on the franchisor and the franchisee can agree in both location and design for the new facility, taking the corresponding economical implications of their selection into account.

In the model the *demand* is supposed to be *inelastic* and concentrated at  $n$  demand points, whose locations  $p_i$  and *buying power*  $w_i$  are known. The location  $f_j$  and quality of the existing facilities is also known. In the spirit of Huff [29] we consider that the demand points split their buying power among all the facilities proportionally to the *attraction* they feel for them. The attraction (or utility) function of

a customer towards a given facility depends on the distance between the customer and the facility, as well as on other characteristics of the facility which determine its *quality*. The location and the quality of the new facility are the variables of the problem.

The following notation will be used throughout this paper:

$x$	location of the new facility, $x = (x_1, x_2)$ .
$\alpha$	quality of the new facility ( $\alpha > 0$ ).
$n$	number of demand points.
$p_i$	demand points, $p_i = (p_{i1}, p_{i2})$ ( $i = 1, \dots, n$ ).
$w_i$	demand (or buying power) at $p_i$ .
$m$	number of existing facilities.
$f_j$	existing facilities ( $j = 1, \dots, m$ ).
$k$	number of existing facilities that are part of one's own chain (the first $k$ of the $m$ facilities are assumed in this category, $0 < k < m$ ).
$d_{ij}$	distance between demand point $p_i$ and facility $f_j$ .
$d_{ix}$	distance between $p_i$ and the new facility $x$ .
$\alpha_{ij}$	quality of facility $f_j$ as perceived by demand point $p_i$ .
$g_i(\cdot)$	a non-negative non-decreasing function.
$\frac{\alpha_{ij}}{g_i(d_{ij})}$	attraction that demand point $p_i$ feels for facility $f_j$ .
$\gamma_i$	weight for the quality of $x$ as perceived by $p_i$ .
$\frac{\gamma_i \alpha}{g_i(d_{ix})}$	attraction that demand point $p_i$ feels for $x$ .

From the previous assumptions, the total market share attracted by the franchisor is

$$M(x, \alpha) = \sum_{i=1}^n w_i \frac{\frac{\gamma_i \alpha}{g_i(d_{ix})} + \sum_{j=1}^k \frac{\alpha_{ij}}{g_i(d_{ij})}}{\frac{\gamma_i \alpha}{g_i(d_{ix})} + \sum_{j=1}^m \frac{\alpha_{ij}}{g_i(d_{ij})}}.$$

We assume that the operating costs for the franchisor due to the new facility are fixed. In this way, the profit obtained by the franchisor is an increasing function of the market share that it captures. Thus, maximizing the profit obtained by the franchisor is equivalent to maximizing the market share that it captures. This will be the first objective of our problem.

The second objective of our problem is the maximization of the profit obtained by the franchisee, to be understood as the difference

between the revenues obtained from the market share captured by the new facility minus its operational costs. The market share captured by the new facility (franchisee) is given by

$$m(x, \alpha) = \sum_{i=1}^n w_i \frac{\frac{\gamma_i \alpha}{g_i(d_{ix})}}{\frac{\gamma_i \alpha}{g_i(d_{ix})} + \sum_{j=1}^m \frac{\alpha_{ij}}{g_i(d_{ij})}}$$

and the profit is given by the following expression,

$$\pi(x, \alpha) = F(m(x, \alpha)) - G(x, \alpha)$$

where  $F(\cdot)$  is a strictly increasing function which determines the expected sales (i.e., income generated) for a given market share  $m$  and  $G(x, \alpha)$  is a function which gives the operating cost of a facility located at  $x$  with quality  $\alpha$ .

In our computational studies we have considered  $F$  to be linear and  $G$  to be separable, of the form  $G(x, \alpha) = G_1(x) + G_2(\alpha)$ , where  $G_1(x) = \sum_{i=1}^n \Phi_i(d_{ix})$ , with  $\Phi_i(d_{ix}) = w_i / ((d_{ix})^{\varphi_{i0}} + \varphi_{i1})$ ,  $\varphi_{i0}, \varphi_{i1} > 0$  and  $G_2(\alpha) = e^{\frac{\alpha}{\alpha_0} + \alpha_1} - e^{\alpha_1}$ , with  $\alpha_0 > 0$  and  $\alpha_1$  given values (other possible expressions for  $G(x, \alpha)$  can be found in [16]).

The problem to be solved is then

$$\begin{cases} \max & M(x, \alpha) \\ \max & \pi(x, \alpha) \\ \text{s.t.} & d_{ix} \geq d_i^{\min} \quad \forall i \\ & \alpha \in [\alpha_{\min}, \alpha_{\max}] \\ & x \in R \subset \mathbb{R}^2 \end{cases} \quad (10)$$

where the parameters  $d_i^{\min} > 0$  and  $\alpha_{\min} > 0$  are given thresholds, which guarantee that the new facility is not located over a demand point and that it has a minimum level of quality, respectively. The parameter  $\alpha_{\max}$  is the maximum value that the quality of a facility may take in practice. By  $R$  we denote the region of the plane where the new facility can be located.

In order to have problem (10) written in the form of problem (1), in what follows we will use the following notation:  $y = (x, \alpha)$ ,  $f_1(y) = -M(x, \alpha)$ ,  $f_2(y) = -\pi(x, \alpha)$  and  $S$  will denote the feasible set of problem (10).

## 5.2. AN EXAMPLE

To clarify the biobjective nature of the problem and the way the constraint-like method works, consider Figure 7. In the picture on the left, light grey circles with numbers 1 to 5 denote forbidden regions around the existing demand points, supposed to be at the center of the forbidden regions and all with demand 1, the cross  $\times$  denotes the location of an existing facility owned by the chain and the dotted box  $\square$  the location of a competitor's facility. The franchisor would like the new facility to be located close to demand point 5 (he/she already captures most of the market of demand points 1 to 4, and in this way he/she can win a part of the market of demand point 5), whereas the franchisee would like the facility to be located close to the existing chain-owned facility (in this way, he can capture nearly half of the market of demand points 1 to 4, which is much more than he/she can get by locating close to demand point 5). In different colors we can see the part of the outer approximation of the efficient set (projected in the location space) obtained with the solution of the different constraint problems considered in the execution of CLM. In the picture on the right we can see the corresponding outer approximation of the nondominated set offered by the algorithm. Notice that the biobjective problem considered is rather difficult (each objective alone leads to a global optimization problem), and its efficient and nondominated sets may have a very general shape (they can be even non-connected, as in the example).

## 5.3. TEST PROBLEMS

To investigate the performance of Algorithm 1, as well as the efficiency of the different discarding tests, we have generated different problems, varying the number of demand points ( $n = 50, 100$ ), the number of existing facilities ( $m = 2, 5, 10$ ) and the number of those facilities belonging to the chain ( $k = 0, 1$  for  $m = 2$ ,  $k = 0, 1, 2$  for  $m = 5$  and  $k = 0, 2, 4$  for  $m = 10$ ). For every setting one instance was generated, by randomly choosing the parameters uniformly within the following intervals:

- $f_j, p_i \in [0, 10]^2$ ,
- $\omega_i \in [1, 10]$ ,
- $\gamma_i \in [0.75, 1.25]$ ,
- $\alpha_{ij} \in [0.5, 5]$ ,
- $\varphi_{i0} = \varphi_0 = 2, \varphi_{i1} \in [0.5, 2]$  (recall that we have used  $G_1(x) = \sum_{i=1}^n \Phi_i(d_{ix})$ , with  $\Phi_i(d_{ix}) = w_i \frac{1}{(d_{ix})^{\varphi_{i0} + \varphi_{i1}}}$ ),

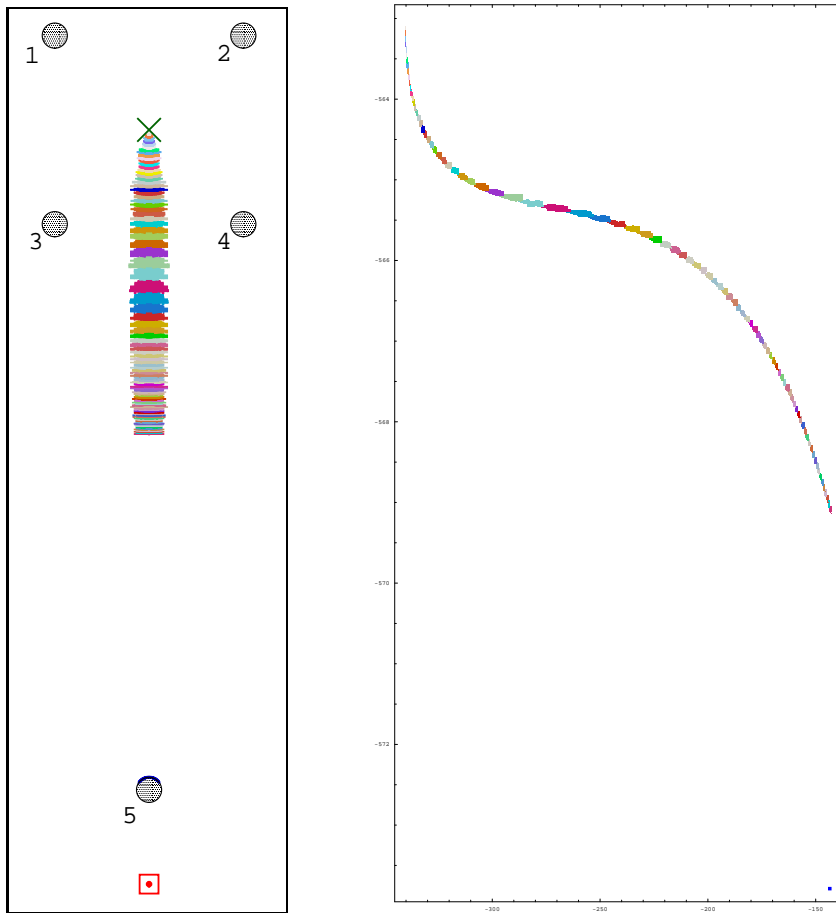


Figure 7. Conflicting objectives.

- $\alpha_0 \in [7, 9], \alpha_1 \in [4, 4.5]$ , the parameters for  $G_2(\alpha) = e^{\frac{\alpha}{\alpha_0} + \alpha_1} - e^{\alpha_1}$ ,
- $c \in [1, 2]$ , the parameter for  $F(M(x, \alpha)) = c \cdot M(x, \alpha)$ ,
- $b_1, b_2 \in [1, 2]$ , parameters for  $d_{ix} = \sqrt{b_1(x_1 - p_{i1})^2 + b_2(x_2 - p_{i2})^2}$   
(see [15])

The searching space for every problem was

$$x \in [0, 10]^2, \quad \alpha \in [0.5, 5]$$

We set  $\delta = 0.01$  and the tolerances used in the algorithm were  $\epsilon = 0.01$  and  $\eta = 0.005$ .

#### 5.4. RESULTS

All the computational results presented in this paper have been obtained on a PC with an Intel Pentium IV 2.33GHz processor and with 1 Gbyte RAM running under Linux operating system. For the implementation we have used the interval arithmetic modules provided in the PROFIL/BIAS library [31] and the automatic differentiation of the C++ toolbox library described in [24].

First, we have studied the usefulness of the different discarding tests. To this end, we have solved all the problems with the following algorithms:

**simple** : in this algorithm we only use the  $\delta$ -cut-off and the feasibility test. In the later, we use the corresponding natural interval extensions as the inclusion functions of the constraints to check the feasibility. The pruning test is substituted by a simple bisection of the box under consideration perpendicular to the direction of maximum width.

**basic** : we use the tests in ‘simple’, but now, in the feasibility test, for the constraint on  $f_2$  we use the centered form as inclusion function.

**basic + mono** : we use the tests in ‘basic’ and the  $\delta$ -monotonicity test.

**basic + mco** : we use the tests in ‘basic’ and the multi-objective cut-off test.

**basic + prun $f_2$**  : we use the tests in ‘basic’ and the pruning test applied to  $f_2$ .

**basic + prun $f_1f_2$**  : we use the test in ‘basic’ and the pruning test applied to  $f_1$  and  $f_2$ .

**basic + mco + prun $f_1f_2$**  : we use the tests in ‘basic + prun $f_1f_2$ ’ and the multi-objective cut-off test.

**basic + all** : in which we use all the discarding tests, that is, we use as main procedure the one given in Algorithm 2.

The results obtained are given in Table I. For the ‘basic’ algorithm we computed the CPU time in seconds (Time), the effort of the algorithm (to be understood as the number of function evaluations plus three times the number of gradient evaluations (recall that our problem is a 3-dimensional one), Effort), the maximum number of boxes stored in the lists (L1+L2+L3+L4) at any time during the execution of the

Table I. Comparison of the different discarding tests.

Problem			Basic					Simple				
$n$	$m$	$k$	Time	Effort	ML	FB	Vol	T%	Eff%	ML%	#FB%	Vol%
50	2	0	130.0	793758	12787	31297	8.8e-3	-	-	-	-	-
50	2	1	431.4	3003327	11500	129854	4.8e-2	-	-	-	-	-
50	5	0	411.6	2663334	11333	96011	1.4e-2	-	-	-	-	-
50	5	1	287.3	1812843	8078	72260	2.6e-2	-	-	-	-	-
50	5	2	625.7	4083624	16684	180296	8.2e-2	-	-	-	-	-
50	10	0	925.3	6211638	25372	241640	4.9e-2	-	-	-	-	-
50	10	2	523.2	3088389	21328	97610	3.6e-2	-	-	-	-	-
50	10	4	189.3	790578	11117	11139	4.5e-3	-	-	-	-	-
100	2	0	108.0	370505	12567	15749	5.2e-3	271	296	378	562	596
100	2	1	192.9	644146	5876	38062	7.3e-2	-	-	-	-	-
100	5	0	265.1	834287	18549	19802	2.0e-3	-	-	-	-	-
100	5	1	50.4	127800	4970	702	1.1e-4	419	594	703	8749	8454
100	5	2	115.4	207969	7744	2167	3.1e-3	691	1376	3607	12768	3871
100	10	0	140.7	431133	15809	6188	1.3e-3	-	-	-	-	-
100	10	2	304.0	570058	15349	6913	4.6e-4	-	-	-	-	-
100	10	4	686.0	1727009	21594	43013	3.1e-2	-	-	-	-	-
Average:			336.6	1710024	13791	62043	2.4e-2	-	-	-	-	-
Av. of %:								461	755	1563	7360	4307

algorithm (ML), the number of boxes in  $L_{sol}$  (FB) and the volume of the boxes in  $L_{sol}$  (Vol). For the rest of algorithms we give the relative values of each of those indices as compared to the values for ‘basic’, in percentage. The last two lines of the tables summarize the results, and give the corresponding values (in average) when we consider the sixteen problems altogether (Average), and the average of the relative values of each problem (Av. of %), respectively.

As we can see, whereas ‘basic’ is able to solve all the problems, ‘simple’ runs out of memory in thirteen of the sixteen problems. This clearly shows that the overestimation produced by the natural interval extension used in the feasibility test causes serious troubles to the algorithm, which may be overcome with the use of the centered form. ‘basic’ solves all the problems in a reasonable amount of time (less than 6 minutes) and the volume of the outer approximating set of the efficient set is, in average, 0.024, that is, 0.0053% of the volume of the searching region.

The  $\delta$ -monotonicity test does not seem to be useful for the type of problems under consideration, since it does not significantly reduce

Table I. (*Continued.*) Comparison of the different discarding tests.

Problem			basic + mono					basic + mco				
$n$	$m$	$k$	T%	Eff%	ML%	FB%	Vol%	T%	Eff%	ML%	FB%	Vol%
50	2	0	89.7	89.3	47.6	85.1	86.4	81.5	62.8	32.5	42.7	47.7
50	2	1	101.2	96.5	111.0	102.6	102.1	72.6	54.6	49.4	47.3	50.0
50	5	0	101.6	99.4	76.4	98.9	100.0	83.5	60.4	43.7	47.7	55.7
50	5	1	99.9	97.5	104.7	102.1	100.0	84.5	62.9	60.1	49.4	53.8
50	5	2	100.3	97.5	111.0	106.5	107.3	80.9	57.2	51.9	46.7	52.4
50	10	0	100.5	98.7	63.5	98.1	98.0	90.3	60.8	41.0	46.0	53.1
50	10	2	102.9	101.3	90.8	107.7	108.3	111.6	65.8	96.5	46.2	55.6
50	10	4	106.1	105.8	98.5	104.2	115.6	105.2	72.2	87.4	42.3	53.3
100	2	0	88.6	86.1	86.5	83.7	96.2	56.9	45.6	25.4	24.5	36.5
100	2	1	101.5	99.8	93.5	118.6	111.0	62.9	52.7	48.3	22.3	37.0
100	5	0	107.8	107.1	74.1	87.3	85.0	68.8	57.2	69.3	22.8	44.5
100	5	1	115.7	119.4	104.0	138.7	72.7	86.9	80.8	100.0	23.1	60.0
100	5	2	100.3	96.6	98.4	47.1	58.1	91.8	81.2	100.0	28.1	19.0
100	10	0	84.2	78.9	60.3	72.6	67.7	64.2	55.1	59.6	23.9	40.8
100	10	2	106.4	108.7	89.8	104.4	73.9	89.0	78.0	94.4	52.1	91.3
100	10	4	106.6	106.1	95.8	110.5	116.1	91.6	78.3	99.1	48.2	48.4
Average:			101.8	99.1	85.1	101.9	104.2	86.0	61.8	66.1	44.8	50.0
Av. of %:			100.8	99.3	87.9	98.0	93.6	82.6	64.1	66.2	38.3	49.9

any of the parameters under study. On the contrary, the multi-objective cut-off test is very effective: it reduces the CPU time more than 15%, the corresponding reductions in effort and maximum number of boxes stored are around 35%, and the number of boxes is  $L_{sol}$  and their volume is reduced by almost half.

The pruning test applied to  $f_2$  is also quite effective, specially in the reduction of storage (ML and FB). However, it is better to apply the pruning to both  $f_1$  and  $f_2$ , since all the parameters under study (except ML) improve with regard to ‘basic + prun $f_2$ ’.

When the multi-objective cut-off test and the pruning test applied to  $f_1$  and  $f_2$  are used together, the algorithm obtains the best results. The CPU time is reduced more than 25%, the effort and the volume of the solution boxes to nearly the half, the maximum number of boxes stored is reduced more than 55% and the number of final boxes more than 70%. Although the reductions obtained when used jointly are not the sum of the individual reductions obtained with each discarding test alone, the results clearly show that when they are used together the performance of the algorithm is much better. This is so because the

Table I. (*Continued.*) Comparison of the different discarding tests.

Problem			basic + prun $f_2$					basic + prun $f_1f_2$				
$n$	$m$	$k$	T%	Eff%	ML%	FB%	Vol%	T%	Eff%	ML%	FB%	Vol%
50	2	0	94.2	94.9	56.9	62.5	104.5	95.4	92.9	69.1	62.7	96.6
50	2	1	95.9	91.9	61.5	60.0	97.9	89.8	85.3	66.0	59.7	91.7
50	5	0	107.6	106.7	66.8	73.0	92.9	103.4	101.6	82.9	72.8	92.9
50	5	1	101.2	100.1	62.5	65.0	100.0	95.8	93.1	71.6	65.5	96.2
50	5	2	84.4	81.9	54.3	55.3	101.2	80.3	76.9	65.0	57.9	93.9
50	10	0	105.4	104.7	57.3	71.3	89.8	100.4	98.1	77.3	71.2	87.8
50	10	2	97.2	95.5	39.5	67.3	100.0	94.2	91.3	61.6	71.7	94.4
50	10	4	81.0	65.9	34.8	60.3	115.6	79.7	63.2	50.4	61.6	108.9
100	2	0	96.2	94.3	64.3	56.1	96.2	90.8	88.6	70.5	62.5	90.4
100	2	1	117.4	116.6	73.3	78.2	111.0	108.8	107.2	80.1	73.9	84.9
100	5	0	85.4	80.4	66.0	62.3	90.0	87.4	83.9	80.3	63.1	90.0
100	5	1	74.6	61.9	32.6	60.7	109.1	79.0	68.9	65.0	60.7	109.1
100	5	2	86.9	68.9	34.8	46.3	90.3	89.7	76.1	75.0	45.8	77.4
100	10	0	88.3	84.2	71.2	79.7	100.0	91.0	87.3	94.1	79.7	100.0
100	10	2	83.7	66.6	47.0	67.5	154.3	86.0	71.1	81.5	68.5	152.2
100	10	4	84.9	76.6	53.0	65.2	109.7	83.3	74.6	63.6	66.7	90.3
Average:			94.5	93.4	55.2	65.4	100.0	79.3	70.9	67.6	52.2	83.3
Av. of %:			92.8	86.9	54.7	64.4	103.9	90.9	85.0	72.1	65.3	97.3

type of information that they use is different, and thus, they discard different types of boxes.

If in addition to the previous tests we also use the  $\delta$ -monotonicity test (i.e., the algorithm ‘basic+all’) then all the parameters under study either remain in similar values or slightly worsen, confirming again that the  $\delta$ -monotonicity test is not useful for the type of problems under consideration.

Second, we have compared the best implementation of CLM (basic + mco + prun $f_1f_2$ ) with the interval branch-and-bound method presented in the companion paper [18], which is also aimed at the explicit construction of the full efficient set. The method in [18] deals with the biobjective problem directly, without reducing it to a sequence of single-objective problems, and uses as discarding tests the feasibility and multi-objective cut-off tests. The results obtained are summarized in Table II. For the biobjective interval B&B algorithm in [18] we give the values obtained when the tolerances used are  $\epsilon_1 = \epsilon_2 = \epsilon_3 = 0.01$ , whereas for the ‘basic+mco+prun $f_1f_2$ ’ we give the relative values of each of those indices as compare to the other algorithm, using  $\delta = 0.01$ ,

Table I. (*Continued.*) Comparison of the different discarding tests.

Problem			basic + mco + prun $f_1f_2$					basic + all				
$n$	$m$	$k$	T%	Eff%	ML%	FB%	Vol%	T%	Eff%	ML%	FB%	Vol%
50	2	0	73.4	57.8	28.2	31.2	62.5	77.6	63.1	25.1	33.1	63.6
50	2	1	62.3	49.3	37.7	32.4	58.3	65.3	49.8	47.1	33.3	60.4
50	5	0	83.7	62.9	40.1	39.0	65.7	84.4	65.9	48.6	38.4	67.1
50	5	1	78.5	61.3	49.3	38.7	65.4	83.1	64.1	59.5	40.9	69.2
50	5	2	70.0	51.1	42.6	30.9	57.3	70.5	51.7	52.4	32.7	59.8
50	10	0	85.5	60.1	35.3	37.3	61.2	94.4	65.7	36.9	38.7	63.3
50	10	2	91.4	59.4	47.9	40.8	63.9	102.1	67.9	44.5	42.9	66.7
50	10	4	75.0	44.0	44.8	29.8	64.4	80.5	53.0	45.4	28.0	71.1
100	2	0	49.0	38.7	23.2	16.8	38.5	38.7	33.9	19.7	6.9	26.9
100	2	1	65.2	54.0	44.6	16.7	37.0	55.2	48.5	39.8	10.0	41.1
100	5	0	60.3	46.0	50.8	15.6	50.0	65.4	57.4	44.3	11.1	34.5
100	5	1	68.1	52.4	55.1	16.1	83.6	75.2	65.2	69.9	15.8	38.2
100	5	2	84.7	64.1	63.6	20.6	23.9	90.6	77.3	71.7	12.0	31.3
100	10	0	52.2	40.1	33.2	15.1	45.4	58.1	48.7	43.6	11.1	32.3
100	10	2	76.1	50.1	57.9	35.7	158.7	86.4	69.1	53.9	51.4	91.3
100	10	4	75.9	56.8	54.4	36.1	48.4	80.3	65.0	47.4	35.6	45.2
Average:			75.7	55.7	43.6	34.0	54.2	80.4	60.4	44.9	34.7	58.3
Av. of %:			71.9	53.0	44.3	28.3	61.5	75.5	59.1	46.9	27.6	53.9

$\epsilon = 0.008$  and  $\eta = 0.0004$ . With those tolerances both methods produce outer approximations of similar quality (on average, the volumes of the boxes in the corresponding  $L_{sol}$  lists are very similar).

As we can see, the effort required by our constraint-like method is much greater than for the biobjective interval B&B algorithm (due to the use of gradient information). However, our implementation of CLM is considerably faster (on average it needs 60% less time) and needs much less space (in average, ML is reduced around 75%). As for the number of boxes in  $L_{sol}$ , it is greater for our constraint-like method, although the area covered by them is smaller. Thus, we can say that our constraint-like method clearly outperforms the biobjective interval B&B algorithm in [18].

## 6. Conclusions and future research

We have presented a general method for obtaining an outer approximation of the efficient set (and the nondominated set) of nonlinear

Table II. Comparison with the direct B&amp;B method

Problem			Biobjective Algorithm					basic+mco+prun $f_1f_2$				
$n$	$m$	$k$	Time	Effort	ML	FB	Vol	T %	Eff%	ML	FB%	Vol %
50	2	0	192.2	221297	5961	10968	4.6e-3	89.0	94.5	26.0	107.8	42.4
50	2	1	2161.1	647089	6814	53767	2.3e-2	102.7	61.6	13.2	184.0	24.0
50	5	0	750.5	479485	11224	25535	1.1e-2	59.1	137.5	18.2	120.2	110.3
50	5	1	958.5	513545	10280	30648	1.3e-2	94.7	170.9	71.4	65.3	82.4
50	5	2	711.6	1136529	17042	83636	3.5e-2	22.8	154.8	35.3	44.7	301.4
50	10	0	4671.7	1136641	15424	76011	3.2e-2	95.3	130.8	26.2	116.0	33.6
50	10	2	432.7	572121	38716	14151	5.9e-3	53.8	44.7	12.9	102.8	30.7
50	10	4	184.7	447873	40468	5227	2.2e-3	63.5	38.0	12.5	72.4	164.0
100	2	0	54.4	81497	3207	3959	1.7e-3	22.2	469.2	47.9	183.7	56.2
100	2	1	660.4	279157	5948	17463	7.3e-3	136.7	459.1	18.1	419.3	237.3
100	5	0	164.7	297785	22862	3343	1.4e-3	81.4	89.5	9.4	81.2	81.8
100	5	1	61.1	140285	14636	178	7.5e-5	67.1	302.4	56.3	142.3	63.0
100	5	2	147.4	319637	26935	595	2.5e-4	19.0	355.1	76.6	120.9	78.3
100	10	0	84.5	184801	16400	1401	5.9e-4	62.1	497.8	43.8	240.3	51.8
100	10	2	233.5	501065	45669	2412	1.0e-3	33.8	330.4	40.2	145.5	73.8
100	10	4	1067.2	953653	42691	20818	8.7e-3	92.3	295.8	46.0	101.4	88.6
Average:			783.5	494528	20267	21882	9.2e-3	41.5	274.4	23.9	148.7	92.4
Av. of %:								68.5	227.0	34.6	140.5	95.0

biobjective optimization problems. It is based on the *constraint method*, and transforms the problem to the computation of the regions of  $\delta$ -optimality of a finite number of single objective constraint problems.

A unified interval branch-and-bound implementation is developed, for which specific discarding tests are proposed. Although the  $\delta$ -cut-off test and the feasibility test (using the centered form as inclusion function for the constraint on  $f_2$ ) are enough to guarantee the convergence of the method, the use of other discarding tests, namely, the pruning test applied to  $f_1$  and  $f_2$  and the multi-objective cut-off test, makes the implementation much faster and reduces its storage requirements. In fact, when compared to another general method recently proposed in [18] with the same aim, our method has shown to be clearly superior.

The design of additional discarding tests, as well as other accelerating devices, will be the subject of future research. The extension of the method and the discarding tests to more than two objectives, and the study of its efficiency, should also be investigated.

## References

1. Agrell P.J., Lence B.J. and Stam A. (1998), An interactive multicriteria decision model for multipurpose reservoir management: the Shellmouth reservoir, *Journal of Multi-Criteria Decision Analysis* 7, 61–86.
2. Benson H.P., and Sayin S. (1997), Towards finding global representations of the efficient set in Multiple Objective Mathematical Programming, *Naval Research Logistics* 44, 47–67.
3. Carrizosa E., Conde E. and Romero-Morales D. (1997), Location of a semiobnoxious facility. A biobjective approach, in *Advances in multiple objective and goal programming 1996*, Lecture Notes in Economics and Mathematical Systems 455, Springer-Verlag, 338–346.
4. Chankong V. and Haimes Y.Y. (1983), *Multiobjective decision making theory and methodology*, Elsevier Science Publishing Co., Inc., New York.
5. Cohon J.L. (1978), *Multiobjective programming and planning*, Academic Press, Inc., New York.
6. Csendes T., Zabinsky Z.B. and Kristinsdottir B.P. (1995), Constructing large feasible suboptimal intervals for constrained nonlinear optimization, *Annals of Operations Research* 58, 279–293.
7. Drezner Z. (Ed.) (1995), *Facility Location: a Survey of Applications and Methods*, Springer, Berlin.
8. Drezner Z. and Hamacher H.W. (Eds.) (2002), *Facility location. Applications and theory*, Springer.
9. Ehrgott M. and Gandibleux X. (Eds.) (2002) *Multiple criteria optimization: state of the art annotated bibliographic surveys*, Kluwer.
10. Ehrgott M. (2005) *Multicriteria Optimization (2nd edition)*, Springer.
11. Ehrgott M., Klamroth K. and Schwehm S. (2004), An MCDM approach to portfolio optimization, *European Journal of Operational Research* 155, 752–770.
12. Ehrgott M. and Ryan D.M. (2002), Constructing robust crew schedules with bicriteria optimization, *Journal of Multi-Criteria Decision Analysis* 11, 139–150.
13. Ehrgott M. and Wiecek M.M. (2005), Multiobjective programming, in [19], 667–722.
14. Eiselt H.A., Laporte G. and Thisse J.-F. (1993), Competitive location models: a framework and bibliography, *Transportation Science* 27, 44–54.
15. Fernández J., Fernández P. and Pelegrín B. (2002), Estimating actual distances by norm functions: a comparison between the  $l_{k,p,\theta}$ -norm and the  $l_{b_1,b_2,\theta}$ -norm and a study about the selection of the data set, *Computers and Operations Research* 29, 609–623.
16. Fernández J., Pelegrín B., Plastria F. and Tóth B. (2005), Solving a Huff-like competitive location and design model for profit maximization in the plane, *European Journal of Operational Research*, to appear (doi:10.1016/j.ejor.2006.02.005).
17. Fernández J., Pelegrín B., Plastria F. and Tóth B. (2005), Planar location and design of a new facility with inner and outer competition: an interval lexicographical-like solution procedure, *Networks and Spatial Economics*, to appear.
18. Fernández J., Tóth B., Plastria F. and Pelegrín B. (2006), Reconciling franchisor and franchisee: a planar biobjective competitive location and design

- model, in *Recent Advances in Optimization*, Lectures Notes in Economics and Mathematical Systems 563, Springer-Verlag, 375-398.
19. Figueira J., Greco S. and Ehrgott M. (Eds.) (2004) , *Multiple criteria decision analysis: State of the art surveys*, Kluwer.
  20. Francis R.L., McGinnis L.F. and White J.A. (1992) *Facility layout location: an analytical approach*, Prentice Hall, Englewood Cliffs.
  21. Gal T. and Hanne T. (1997), On the development and future aspects of vector optimization and MCDM, in *Multicriteria analysis*, J. Clímaco (ed.), Springer-Verlag, Berlin, Heidelberg, 130–145.
  22. Ghosh A. and Craig C.S. (1991) FRANSYS: a franchise distribution system location model, *Journal of Retailing* 67, 466–495.
  23. Haimes Y.Y., Lasdon L.S. and Wismer D.A. (1971) On a bicriterion formulation of the problems of integrated system identification and system optimization, *IEEE Transactions on System, Man and Cybernetics* 1, 296-297.
  24. Hammer R., Hocks M., Kulisch U. and Ratz D. (1995), *C++ Toolbox for verified computing*, Springer-Verlag, Berlin.
  25. Hansen E. and Walster G.W. (2004), *Global Optimization Using Interval Analysis, Second Edition, Revised and Expanded*, Marcel Dekker.
  26. Hansen P. and Jaumard B. (1995), Lipschitz optimization, in *Handbook of Global Optimization*, Kluwer, Dordrecht, 407–494.
  27. Horst R. and Pardalos P.M. (Ed) (1995) *Handbook of Global Optimization, Nonconvex Optimization and its Applications Vol.2*, Kluwer.
  28. Horst R. and Tuy H. (1996) *Global Optimization: Deterministic Approaches (3rd edition)*, Springer.
  29. Huff D.L. (1964), Defining and estimating a trading area, *Journal of Marketing* 28, 34–38.
  30. Kilkenny M. and Thisse J.F. (1999) Economics of location: a selective survey, *Computers and Operations Research* 26, 1369–1394.
  31. O. Knüppel (1994), PROFIL/BIAS — a fast interval library, *Computing* 53, 277–287.
  32. Kristinsdottir B.P., Zabinsky Z.B., Csendes T. and Tuttle M.E. (1993), Methodologies for tolerance intervals, *Interval Computations* 3, 133–147.
  33. Kearfott R.B. (1996), *Rigorous Global Search: Continuous Problems*, Kluwer, Dordrecht.
  34. Kearfott R.B., Nakao M.T., Neumaier A., Rump S.M., Shary S.P., and van Hentenryck P. (2002), Standardized notation in interval analysis, submitted to *Reliable Computing*, available at <http://www.mat.univie.ac.at/~neum/software/int/>
  35. Küfer K.H., Scherrer A., Monz M., Alonso F., Trinkaus H., Bortfeld T. and Thieke C. (2003) Intensity-modulated radiotherapy - a large scale multi-criteria programming problem, *OR Spectrum* 25, 223–249.
  36. Love R.F., Morris J.G. and Wesolowsky G.O. (1988), *Facilities location: models and methods*, North-Holland, New York.
  37. Martínez J.A., Casado L.G., García I. and Tóth B., AMIGO: Advanced Multidimensional Interval analysis Global Optimization algorithm, in *Frontiers in Global Optimization, Nonconvex Optimization and Its Applications* vol. 74, Kluwer, 313–326.
  38. Miettinen K.S. (1998), *Nonlinear multiobjective optimization*, Kluwer, Boston.
  39. Nickel S. and Puerto J. (2005), *Location theory - a unified approach*, Springer, Berlin.

40. Plastria F. (1992), GBSSS: The Generalized Big Square Small Square Method for Planar Single-Facility Location, *European Journal of Operational Research* 62, 163–174.
41. Plastria F. (2001), Static competitive facility location: an overview of optimisation approaches, *European Journal of Operational Research* 129, 461–470.
42. Ratschek H. and Rokne J. (1988), *New Computer Methods for Global Optimization*, Ellis Horwood, Chichester.
43. Ruzika S. and Wiecek M.M. (2005), Approximation methods in multiobjective programming, *Journal of Optimization Theory and Applications* 126, 473–501.
44. Sayin S. (2000), Measuring the quality of discrete representations of efficient sets in multiple objective mathematical programming, *Mathematical Programming* 87, 543–560.
45. Schniederjans M. J. and Hollcroft E. (2005), A multi-criteria modeling approach to jury selection, *Socio-Economic Planning Sciences* 39, 81–102.
46. Silverman J., Steuer R.E. and Whisman A.W. (1988), A multi-period, multiple criteria optimization system for manpower planning, *European Journal of Operational Research* 34, 160–170.
47. Steuer R.E. (1986), *Multiple criteria optimization: theory, computation, and applications*, John Wiley & Sons, Inc., 1986.
48. Tóth B. and Csendes T. (2005), Empirical investigation of the convergence speed of inclusion functions, *Reliable Computing* 11, 253–273.
49. Tóth B., Fernández J. and Csendes T. (2006), Empirical convergence speed of inclusion functions for facility location problems, *Journal of Computational and Applied Mathematics*, to appear (doi:10.1016/j.cam.2005.07.037).
50. White D.J. (1990), A bibliography on the applications of Mathematical Programming multiple-objective methods, *Journal of the Operational Research Society* 41, 669–691.
51. Yu P.L. (1985), *Multiple-criteria decision making concepts, techniques and extensions*, Plenum Press, New York, 1985.
52. Zeleny M. (1982), *Multiple criteria decision making*, McGraw-Hill, Inc., 1982.

