An approach to determine unsupported non-dominated solutions in bicriteria integer linear programs

João C. N. Clímaco[(1)], Marta M. B. Pascoal[(1,2)*]

[(1)] Instituto de Engenharia de Sistemas e Computadores – Coimbra, Universidade de Coimbra, Rua Antero de Quental, 199, 3000-033 Coimbra, Portugal
E-mail: jclimaco@inescc.pt

[(2)] Departamento de Matemática da FCTUC, Apartado 3008, EC Santa Cruz, 3001-501 Coimbra, Portugal
Phone: +351-239791150 Fax: +351-239793069
E-mail: marta@mat.uc.pt

April 29, 2016

Abstract: In this paper we introduce a method for finding both supported and unsupported non-dominated solutions of a bicriteria integer linear program (BCILP). One-phase and two-phase implementations of the method are described, and their interactive versions are outlined. The one phase method and the second phase of the other are based on the minimisation of weighted Chebyshev distances to well chosen reference points. The dynamic change of reference point proposed here makes this method particularly suitable for interactive approaches. Computational experiments on random instances of three classes of BCILP are reported and discussed. The implementation of the proposed method as a method to approximate the set of non-dominated solutions is described and evaluated in computational terms.

Keywords: Bicriteria problems, Supported and unsupported non-dominated solutions, Two-phase method, Reference point, Chebyshev metrics.

1 Introduction

Let us consider the bicriteria integer linear program (BCILP) with two objective functions, n variables and m constraints:

$$\begin{array}{ll} \min & f(x) = (f_1(x), f_2(x)) \\ \text{subject to} & Ax \leq b \\ & x \geq 0 \\ & x \in \mathbb{Z}^n \end{array} \tag{1}$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $f_i(x) = c_i^T x$, with $c_i \in \mathbb{R}^n$ for $i = 1, 2$. We assume that the feasible region $S = \{x \in \mathbb{Z}^n : Ax \leq b,\ x \geq 0\}$ is nonempty and bounded. We assume, without loss of generality, the minimisation of the objective functions. When dealing with single criterion problems the goal is to find a feasible solution, which satisfies all the constraints, with optimal objective function value (regardless of its uniqueness). In this case the interesting solutions are compromise solutions, those for which it is impossible to find another feasible solution improving one objective function, without worsening at least another one. These form the set of efficient solutions and "solving" a bicriteria problem is intended as computing this set of solutions. Typically the decision maker (DM) will only choose and implement a single solution and the choice is guided by preferences

*Corresponding author

regarding some features of the solutions. Thus, bicriteria approaches follow three trends: *a priori* articulation of preferences (the objective functions are replaced by a single one that aggregates all the preferences), *a posteriori* (with no articulation of preferences and listing the whole set of efficient solutions), and interactive (there is a progressive articulation of preferences by the DM and an interactive selection by means of a dialogue phase).

The main purpose of the present work is to introduce an algorithm for calculating the efficient solutions of a general bicriteria integer linear program. The method is based on a reference point approach, using Chebyshev metrics and a particular choice of the reference points, and it enables the determination of all types of efficient solutions. The use of Chebyshev objective functions and reference points has been proposed in earlier works. The main characteristic of the method presented here is the proposed choice of reference points and its dynamical update, which make clear the areas left to search in the objective function space. For this reason the introduced approach is suitably designed for interactive applications. The running time of the method is not crucial for these type of applications, given that it implies feedback from the DM every time a solution is output. Nevertheless, the computational performance of the algorithm is evaluated on random instances and it is compared with methods in previous literature.

The remainder of the text is organised into five other sections. The next section is devoted to the introduction of concepts and notation, and to a short literature review on algorithms for bicriteria integer linear programs. Afterwards the method for determining the set of efficient solutions is introduced. Different variants of this method are discussed and an example illustrates its application. The section finishes with the presentation of computational experiment results. In Sections 4 and 5 it is discussed how the variants of the method can be applied to approximate the set of efficient solutions and how they can be used within an interactive approach, respectively. More computational tests and an example are presented. Conclusions are drawn in Section 6.

2 Literature review

A solution $x \in S$ of problem (1) is **efficient** if and only if there is no other solution $x' \in S$ such that $f(x') \leq f(x)$ and $f(x') \neq f(x)$. A criterion vector $f(x)$ is **non-dominated** if and only if x is efficient. We address the determination of non-dominated solutions. This means that only one solution is calculated representing an efficient point in the objective functions space. In spite of these two standard definitions, in the remainder of the paper we might use the terms efficient solution and non-dominated solution interchangeably. All the non-dominated solutions of a BCILP form the problem's **Pareto frontier**.

When two criteria have different priorities we talk about lexicographic solutions. A solution $s^* \in S$ is **lexicographic best with respect to** (f_1, f_2) if

1. $f_1(s^*) \leq f_1(s)$, for any $s \in S$, and
2. $f_2(s^*) \leq f_2(s)$, for any $s \in S$ such that $f_1(s^*) = f_1(s)$.

A solution of (1) which is lexicographic best with respect to (f_1, f_2), or with respect to (f_2, f_1), is an efficient solution of (1). These solutions can be found by solving two single criterion integer linear programs. Assuming that the aimed order is (f_1, f_2), first

$$\begin{array}{ll} \min & f_1(x) \\ \text{subject to} & x \in S \end{array} \tag{2}$$

is solved. Then, denoting its optimal value by c^*, the problem

$$\begin{array}{ll} \min & f_2(x) \\ \text{subject to} & f_1(x) = c^* \\ & x \in S \end{array} \tag{3}$$

is considered.

A solution $\bar{s}$ is **weakly efficient** when no other is strictly better in all criteria, that is, if and only if there is no $s \in S$ such that $f_1(s) < f_1(\bar{s})$ and $f_2(s) < f_2(\bar{s})$. For instance, given two solutions, s_1 and s_2, such that $f(s_1) = (10, 5)$ and $f(s_2) = (10, 10)$, then s_1 dominates s_2, but this latter solution is weakly efficient.

Two types of non-dominated solutions can be distinguished as illustrated in Figure 1,

- **Supported non-dominated solutions**, which are non-dominated solutions that are optimal solutions of a (single-criterion) weighted sum problem (WSP)

$$\min_{x \in S} \left\{ \sum_{i=1}^{2} w_i f_i(x) \right\} \tag{4}$$

  with $w_i > 0$, $i = 1, 2$. Note that if at least one of the weights is null the WSP may have alternative optima which are dominated solutions of (1).

- The remaining non-dominated solutions are called **unsupported solutions**, and they cannot be obtained as solutions of a WSP.

The former solutions can be obtained by solving a sequence of WSP's and varying the weights in a suitable manner, as will be described later. The determination of unsupported solutions tends to be more difficult than this, given that not all methods are able to find such solutions. Nevertheless, unsupported solutions can still be found using, for instance, the ε-constraint method, the WSM with additional constraints or single-phase methods with Chebyshev objective functions. In this work we introduce a reference point algorithm for calculating all non-dominated solutions of a BCILP, and show it can be used in one or two phases, as well as in an interactive framework.

Because not all methods are able to find unsupported non-dominated solutions, the calculation of the two types of solutions is often done in two phases. First the supported solutions are computed. Considering the non-dominated solutions in a given set sorted by increasing order of one of the two criteria, for instance f_1, two of them, s_1 and s_2, are said to be **adjacent non-dominated solutions** if there is no other non-dominated solution in that set, s, such that $f_1(s_1) < f_1(s) < f_1(s_2)$. It should be stressed that while the algorithms run the set of computed non-dominated solutions is

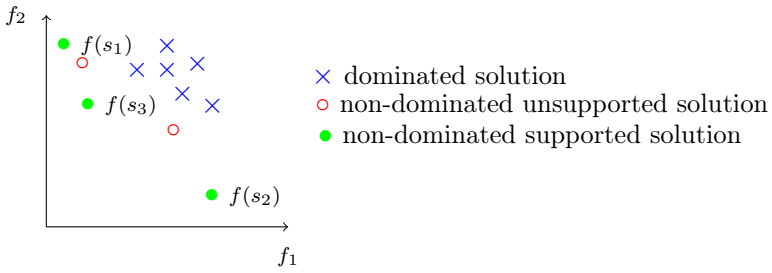


Figure 1: BCILP solutions

updated, and thus the adjacent pairs may change. In the second phase the search continues by looking for unsupported solutions the images of which lie within the duality gaps formed by two adjacent supported solutions. For further details see [2, 5, 12, 15, 17].

The search for solutions within duality gaps has been investigated since the work of Current, ReVelle and Cohon [6]. The proposed approaches include using side constraints to obtain a sequence of "easier" problems, ranking the K best solutions of the problem to sweep that area, using dynamic programming and using enumerative branch and bound procedures. The application of side constraints to find unsupported solutions was introduced in [6] and is related with the ε-constraint method, used by Mavrotas [11] for finding all non-dominated solutions. Using ranking algorithms to perform the search has revealed to be quite efficient whenever these methods can run fast. The bicriteria shortest path problem or the bicriteria spanning tree problem are two of those cases. In the first the search can be done quite easily, as presented by Coutinho-Rodrigues, Clímaco and Current [4], whereas the second is harder but still manageable for small problems, see Steiner and Radzik [14]. For general problems, however, it can be difficult to rank solutions, and therefore also difficult to perform the search. Alternative approaches using dynamic programming have been proposed concerning the bicriteria shortest path problem, see Raith and Ehrgott [12], as well as using branch and bound methods for the bicriteria knapsack problem, see Visée, Teghem, Pirlot and Ulungu [17].

As alternative ways to determine all the non-dominated solutions, rather than treating supported and unsupported solutions separately, we point out the parametric algorithm for bicriteria integer programs in [13], and the ε-constraint algorithm in [11]. Ralphs, Saltzman and Wiecek [13] propose a method based on the weighted Chebyshev scalarisation, where the weights depend on the images of pairs of consecutive solutions and the ideal point of the original problem. More recently Dächert, Gorski and Klamroth [7] presented an adapted augmented weighted Chebyshev method with an adaptive choice of parameters for discrete bicriteria optimisation problems. Ferreira [8] also used a weighted Chebyshev approach to obtain non-dominated solution for a bicriteria location and distribution problem. More recently, Boland, Charkhgard and Savelsbergh [1] proposed the balanced box method, which computes both supported and unsupported solutions. This is an extension of the method introduced by Hamacher, Pedersen and Ruzika [10], which finds solutions by splitting the search area in two, horizontally, and computing one lexicographic best solution in each of the

two new regions thus obtained.

3 Algorithmic approach

In the current section we describe a method to compute the non-dominated solutions of problem (1) in two variants. One of these variants is a two-phase approach with a first phase that solves weighted sum problems and a reference point based second phase, whereas the other consists of using only the second phase of the former one. This section begins by recalling the weighted sum method, proceeds with the introduction of the second phase method and then shows that this can be used in one-phase only. The presentation of both approaches is followed by an application example and by computational test results.

As mentioned before, our goal is to determine the non-dominated solutions of a BCILP, among which we distinguish the supported and the unsupported solutions. Figure 1 illustrates these different types of solutions. The first approach proposed here is comprised of two phases, where the first phase is responsible for the supported solutions calculation, whereas the remaining non-dominated solutions, i.e., the unsupported ones, are calculated during the second part. In the following the first phase of the method is described.

One of the most well known techniques to calculate supported non-dominated solutions is the WSM. This approach is introduced for instance by Cohon [3], Coutinho-Rodrigues et al. [4] or Steuer [15]. We now summarise this method, for completeness.

The WSM we use is inspired by the type of choice of weights used in the non inferior set estimation (NISE) method – [3]. It calculates non-dominated solutions by solving a sequence of WSPs and updating their parameters according to the found solutions. The method works by starting with the lexicographic best solution with respect to each criteria, and then taking pairs of consecutive solutions, s_1, s_2, and, for each, trying to calculate a new one by solving a single criterion WSP. If there exist further supported non-dominated solutions within the triangle $\Delta(f(s_1)c^*f(s_2))$, where $c^* = (c_1^*, c_2^*)$ and $c_i^* = f_i(s_i)$, $i = 1, 2$ – Figure 2a – one of them is calculated. The new solution s_3, the image of which is depicted in Figure 2b, is the best regarding the minimisation of the objective function

$$w_1 f_1(s) + w_2 f_2(s),$$

with $w_1 = f_2(s_1) - f_2(s_2)$, $w_2 = f_1(s_2) - f_1(s_1)$, and the problem can be formulated as

$$\begin{array}{ll} \min & w_1 f_1(s) + w_2 f_2(s) \\ \text{subject to} & s \in S \end{array} \tag{5}$$

It should be added that the weights w_1, w_2 can also be normalised. The process is illustrated in Figure 2 and a summary of the whole phase 1 method is included in Algorithm 1.

Several authors have shown that the solution of a WSP is a non-dominated solution. Additionally, when a NISE-like method is applied, either a new supported solution is found within a given pair of solutions, or else concludes that no other supported solutions exist, which proves the method's correctness. More details can be found in [3].

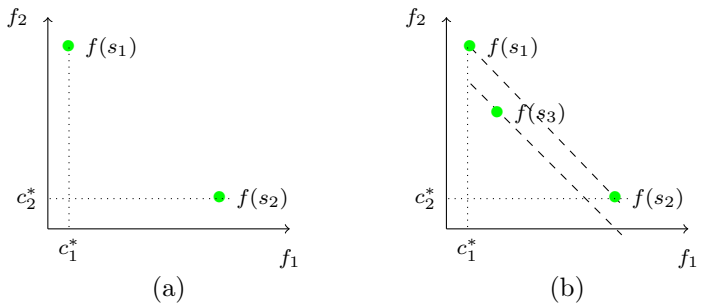


Figure 2: Phase 1, weighted-sum method: (a) initial supported solutions; (b) new supported solution obtained from the optimisation of the weighted sum objective function derived from the previous pair of solutions

Algorithm 1: Supported non-dominated solutions determination

```
   /* (s_1, s_2) identifies a pair of consecutive solutions                    */
   /* L is a set that stores the calculated non-dominated solutions            */
   /* X is an auxiliary set that stores the unscanned pairs of non-dominated
      solutions                                                                */
 1 s_1 ← lexicographic best solution with respect to (f_1, f_2)
 2 s_2 ← lexicographic best solution with respect to (f_2, f_1)
 3 L ← {s_1, s_2}
 4 X ← {(s_1, s_2)}
 5 while X ≠ ∅ do
 6     (s_1, s_2) ← element in X; X ← X − {(s_1, s_2)}
 7     w_1 ← f_2(s_1) − f_2(s_2); w_2 ← f_1(s_2) − f_1(s_1)
 8     s ← best solution to (5)
 9     if s is defined then
10         L ← L ∪ {s}
11         if f_1(s) ≠ f_1(s_2) then X ← X ∪ {(s_1, s)}
12         if f_2(s) ≠ f_2(s_1) then X ← X ∪ {(s, s_2)}
```

As mentioned in Section 2, the optimal lexicographic solutions can be obtained by solving two ILPs each. In practice this calculation can be replaced by optimising f_1 and f_2 separately to find two solutions. The obtained solutions may be weakly efficient (in fact dominated), which makes the initial search area wider than what is necessary. However, Algorithm 1 solves WSPs until no further supported non-dominated solutions are found. The correctness of this method follows from the fact that for each pair of adjacent solutions, s_1, s_2, an ILP is solved, which

- either provides a new supported non-dominated solution in $\Delta(f(s_1)c^*f(s_2))$, with $c_i^* = f_i(s_i)$, $i = 1, 2$, in case it exists,
- or outputs one of the initial solutions, in case no further solutions exist in $\Delta(f(s_1)c^*f(s_2))$.

Given that the lexicographic best solutions correspond to supported non-dominated points and lie within the original search region, these will be computed at a certain stage of the method.

In the following the second part of the two-phase approach for obtaining all the non-dominated solutions of a BCILP is introduced. The goal of this second part is to search for other solutions lying within the duality gaps formed by pairs of the adjacent supported solutions obtained by Algorithm 1, after they are sorted by increasing order f_1. As shown earlier, unsupported solutions are not the optimum solution of any WSP, and thus in general Algorithm 1 is not able to find the whole Pareto frontier. However, Theorem 1 states that unsupported non-dominated solutions are solutions closest to the **ideal point**, the point defined by the best value of both criteria, with respect to a certain weighted Chebyshev distance – see Steuer and Choo [16].

Theorem 1 ([16]). *Let s_i be the optimum solution with respect to f_i, and $c_i^* = f_i(s_i)$ represent the optimum value for each criteria, $i = 1, 2$. If $\bar{s}$ is a non-dominated solution of (1) then it is the optimum solution of*

$$\begin{array}{ll} \min & \max\{w_1|f_1(s) - c_1^*|, w_2|f_2(s) - c_2^*|\} \\ \textit{subject to} & s \in S \end{array}$$

for some $w_1, w_2 > 0$ with $w_1 + w_2 = 1$.

A consequence of this result is that any unsupported solution can be found by using a Chebyshev metric with adequate parameters. Let us now assume that all supported solutions are known. Our next goal is to split the search for the remaining non-dominated solutions, the unsupported, into several regions, the duality gaps of every adjacent supported solutions. That is, taking s_1 and s_2 as two adjacent supported solutions, we want to apply Theorem 1 to search for solutions within the duality gap defined by s_1 and s_2. Because s_1 and s_2 are supported solutions, this duality gap is given by the triangle $\Delta(f(s_1)\hat{c}f(s_2))$, with $\hat{c} = (f_1(s_2), f_2(s_1))$ – see Figure 3a. The objective function in Theorem 1 is a Chebyshev metric, which defines rectangles whose sides have lengths proportional to w_1 and w_2, and with c^* as a vertex. Therefore, the search in the duality gap can replicate this one, by using the same metric with adequate weights, w_1, w_2, and a different reference point, c^*, within the rectangle $\square(c^* f(s_1)\hat{c}f(s_2))$ – see Figure 3a.

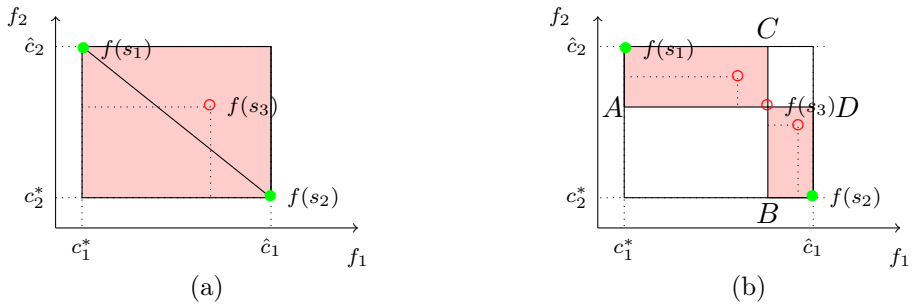


Figure 3: Phase 2, computing unsupported solutions: (a) initial search area; (b) areas to scan after the unsupported solution s_3 is calculated

In this case we consider $w_1 = f_2(s_1) - f_2(s_2)$, $w_2 = f_1(s_2) - f_1(s_1)$, and the reference point c^*, and solve the problem

$$\begin{array}{ll} \min & T(s) = \max\{w_1|f_1(s) - c_1^*|, w_2|f_2(s) - c_2^*|\} \\ \text{subject to} & s \in S \end{array} \tag{6}$$

The image of an optimal solution of (6) is the closest to c^* with respect to the Chebyshev metric T, in the region $\square(c^* f(s_1) \hat{c} f(s_2))$, for instance, like solution s_3 in Figure 3a. Like before, the function weights can be normalised without changing the optimal solution. The previous statement is shown in Proposition 1.

Unless otherwise stated, for simplicity, and with no los of generality, we will assume that (6) has a unique optimum solution.

Proposition 1. *Let s_1, s_2 be a pair of adjacent non-dominated solutions of (1) and s^* be an optimal solution of (6). Then, s^* minimises T in the region bounded by $\square(c^*(c_1^*, f_2(s^*))f(s^*)(f_1(s^*), c_2^*))$.*

Proof. Assume that s^* is the optimum solution of (6), but that there exists another solution $\bar{s} \in S$ in the region bounded by $\square(c^*(c_1^*, f_2(s^*))f(s^*)(f_1(s^*), c_2^*))$ such that $c_i^* \leq f_i(\bar{s}) < f_i(s^*)$, for $i = 1, 2$. Thus,

$$0 \leq f_i(\bar{s}) - c_i^* < f_i(s^*) - c_i^*, \quad i = 1, 2,$$

and because $w_i > 0$, $i = 1, 2$,

$$w_i |f_i(\bar{s}) - c_i^*| < w_i |f_i(s^*) - c_i^*|, \quad i = 1, 2.$$

Therefore,

$$T(\bar{s}) = \max_{i=1,2} \{w_i |f_i(\bar{s}) - c_i^*|\} < \max_{i=1,2} \{w_i |f_i(s^*) - c_i^*|\} = T(s^*),$$

which contradicts the assumption that s^* is optimal. □

Corollary 1.1. *If s^* is the optimum solution of (6), then s^* is a non-dominated solution of (1).*

Proof. Assume $s^* \in S$ is dominated by $\bar{s} \in S$, i.e., $f_i(\bar{s}) \leq f_i(s^*)$, for $i = 1, 2$, and $f(\bar{s}) \neq f(s^*)$. Then,

$$0 \leq f_i(\bar{s}) - c_i^* \leq f_i(s^*) - c_i^*, \quad i = 1, 2,$$

and, similarly to the proof of Proposition 1, we can derive

$$T(\bar{s}) = \max_{i=1,2} \{w_i |f_i(\bar{s}) - c_i^*|\} \leq \max_{i=1,2} \{w_i |f_i(s^*) - c_i^*|\} = T(s^*).$$

Finally, two alternatives are possible,

- $T(\bar{s}) = T(s^*)$, which contradicts s^* uniqueness, or
- $T(\bar{s}) < T(s^*)$, which contradicts s^* optimality,

therefore s^* is a non-dominated solution. □

When s_3 (in Figure 3, and corresponding to s^* in Proposition 1) is determined, it partitions the rectangle $\square(c^* f(s_1) \hat{c} f(s_2))$ into four new regions. Then, the search for non-dominated solutions in the original region can be restricted to searching for non-dominated solutions in just two of them, those bounded by the rectangles $\square(Af(s_1)Cf(s_3))$ and $\square(Bf(s_3)Df(s_2))$ – Figure 3b – which are the rectangles associated with the pairs of solutions (s_1, s_3) and (s_3, s_2).

Proposition 2. *Let s_1, s_2 be adjacent non-dominated solutions of (1) and s^* be the optimum solution of (6). Then, all non-dominated solutions in the region bounded by $\square(c^* f(s_1) \hat{c} f(s_2))$, with $c_i^* = f_i(s_i)$, $i = 1, 2$, $\hat{c}_1 = f_1(s_2)$, and $\hat{c}_2 = f_2(s_1)$, belong either*

1. *to the region bounded by $\square((c_1^*, f_2(s^*))f(s_1)(f_1(s^*), \hat{c}_2)f(s^*))$, or*
2. *to the region bounded by $\square((f_1(s^*), c_2^*)f(s^*)(\hat{c}_1, f_2(s^*))f(s_2))$.*

Proof. It suffices to show that there are no non-dominated solutions in the excluded areas, namely in $\square(c^*(c_1^*, f_2(s^*))f(s^*)(f_1(s^*), c_2^*)$ and $\square(f(s^*)(f_1(s^*), \hat{c}_2), \hat{c}, (\hat{c}_1, f_2(s^*))$.

1. Based on Proposition 1.1, there are no solutions in the first of these regions.
2. Let $\bar{s}$ be a solution within the region bounded by the second rectangle. Thus, $f_i(s^*) < f(\bar{s})$, for $i = 1, 2$, but, by definition, this implies that $\bar{s}$ is dominated by s^*.

□

The termination of the method is guaranteed because the swept area decreases in every iteration, reaching a point when no further non-dominated solutions have to be found. This is shown in Corollary 2.1.

Corollary 2.1. *Let s_1, s_2 be adjacent non-dominated solutions of (5) and s^* be the optimum solution of (6).*

1. *If $f(s^*) = f(s_1)$, then there are no other non-dominated solutions in the region bounded by $\square((c_1^*, f_2(s^*))f(s_1)(f_1(s^*), \hat{c}_2)f(s^*))$.*
2. *If $f(s^*) = f(s_2)$, then there are no other non-dominated solutions in the region bounded by $\square((f_1(s^*), c_2^*)f(s^*)(\hat{c}_1, f_2(s^*))f(s_2))$.*

Proof. It is immediate, given that in both cases the region reduces to a single point, $f(s^*)$. □

So far it was assumed that T does not have multiple optimal solutions. If that happens, there is more than one solution whose image lies along the line segments with endpoints $(c_1^*, f_2(s^*))$ and $f(s^*)$, as well as $f(s^*)$ and $(f_1(s^*), c_2^*)$, and some of them may be weakly dominated by other points over the same line segments. Computing all multiple optimal solutions is possible, but usually a heavy task for mathematical programming solvers. Nevertheless, Steuer and Choo [16] proposed adding a term to the objective function in (6), in order to overcome this drawback of the simple weighted Chebyshev metric. The augmented objective function is

$$T_\rho(s) = \max\{w_1|f_1(s) - c_1^*|, w_2|f_2(s) - c_2^*|\} + \rho\,(f_1(s) + f_2(s))$$

with $\rho > 0$ sufficiently small. Another alternative is to complement solving problem (6) for a given pair s_1, s_2 with solving two additional problems

$$\begin{array}{lll} \min & f_1(s) & \\ \text{subject to} & f_2(s) = f_2(s^*) & \\ & s \in S & \end{array} \quad \text{and} \quad \begin{array}{lll} \min & f_2(s) & \\ \text{subject to} & f_1(s) = f_1(s^*) & \\ & s \in S & \end{array}$$

thus ensuring that the (at most) two obtained solutions are non-dominated. Yet another possibility is to proceed similarly to what was suggested earlier for the WSM. This consists of storing the solutions that are weakly non-dominated (but still dominated). In this case, the two new rectangles still contain the solutions that dominate s^*, as shown in Proposition 3. Therefore, they will be found in a subsequent iteration.

Proposition 3. *Let s_1, s_2 be adjacent non-dominated solutions of (1) and s^* be an optimal solution of (6). Two cases may occur.*

1. *If s^* is dominated by $\bar{s}$ such that $f_2(s^*) = f_2(\bar{s})$, then $T(\bar{s}) < T(s^*)$ in the region bounded by $\square((c_1^*, f_2(s^*))f(s_1)(f_1(s^*), \hat{c}_2)f(s^*))$.*

2. *If s^* is dominated by $\bar{s}$ such that $f_1(s^*) = f_1(\bar{s})$, then $T(\bar{s}) < T(s^*)$ in region bounded by $\square((f_1(s^*), c_2^*)f(s^*)(\hat{c}_1, f_2(s^*))f(s_2))$.*

Proof. The two cases are similar, so we show only the first one. The new function T, referring to $\square((c_1^*, f_2(s^*))f(s_1)(f_1(s^*), \hat{c}_2)f(s^*))$, is defined by

$$T(s) = \max\{w_1|f_1(s) - f_1(s_1)|, w_2|f_2(s) - f_2(s^*)|\},$$

with $w_1 = f_2(s_1) - f_2(s^*)$, $w_2 = f_1(s^*) - f_1(s_1)$, for any $s \in S$. Then,

$$T(s^*) = \max\{w_1|f_1(s^*) - f_1(s_1)|, w_2|f_2(s^*) - f_2(s^*)|\} = w_1|f_1(s^*) - f_1(s_1)|.$$

Similarly, because $f_2(s^*) = f_2(\bar{s})$,

$$T(\bar{s}) = \max\{w_1|f_1(\bar{s}) - f_1(s_1)|, w_2|f_2(\bar{s}) - f_2(s^*)|\} = w_1|f_1(\bar{s}) - f_1(s_1)|.$$

Moreover, $\bar{s}$ dominates s^*, so $f_1(\bar{s}) < f_1(s^*)$, and thus $T(\bar{s}) < T(s^*)$, as claimed. □

The dominated solutions the method finds can be discarded efficiently as they are obtained or after all regions have been explored.

Solving problem (6) and reducing the search to the regions mentioned before can be repeated until there are no more pairs of solutions left to analyse. In the end all the regions that might contain unsupported solutions will have been swept. A summary of the phase 2 method here proposed is included in Algorithm 2.

Some notes to the implementation of this algorithm should be added. First, each pair of solutions that is stored in set X corresponds to, and identifies, an area (rectangle) to be swept. However, if solving the problem in two phases, when Algorithm 1 is finished its output, the solutions in set L, should be sorted by increasing order of f_1, so that the pairs of adjacent solutions are provided to the second phase and each one is an input to Algorithm 2. Second, problem (6) can be formulated as the equivalent, also single criterion, mixed integer linear program

$$\begin{array}{lllll} \min & v & & & \\ \text{subject to} & w_1 f_1(s) - v & \leq & w_1 c_1^* & \\ & w_2 f_2(s) - v & \leq & w_2 c_2^* & \\ & s \in S & & & \end{array} \tag{7}$$

Algorithm 2: Unsupported non-dominated solutions determination within the duality gap defined by (S_1, S_2)

```
   /* (S_1, S_2) is a given pair of adjacent supported solutions                */
   /* L is a set that stores the calculated non-dominated solutions             */
   /* X is an auxiliary set that stores the unscanned pairs of non-dominated
      solutions                                                                 */
 1 s_1 ← S_1
 2 s_2 ← S_2
 3 L ← {s_1, s_2}
 4 X ← {(s_1, s_2)}
 5 while X ≠ ∅ do
 6     (s_1, s_2) ← element in X; X ← X − {(s_1, s_2)}
 7     w_1 ← f_2(s_1) − c*_2; w_2 ← f_1(s_2) − c*_1
 8     s ← best solution to (6)
 9     if s is defined then
10         L ← L ∪ {s}
11         if f(s) ≠ f(s_2) then  X ← X ∪ {(s_1, s)}
12         if f(s) ≠ f(s_1) then  X ← X ∪ {(s, s_2)}
```

and thus it can be solved by means of a mixed integer programming solver.

According to Theorem 1 any non-dominated solution can be found by means of solving the BCILP for a suitable Chebyshev metric. Furthermore, according to Propositions 1 and 2, and their corollaries, solving problem (6) as in Algorithm 2 sweeps a rectangle delimited by the images of solutions s_1 and s_2. Because the set of non-dominated solutions lies within the rectangle given by the lexicographic best solutions with respect to f_1 and to f_2, all the solutions in the Pareto frontier can be computed using simply the procedure presented for the second phase. A summary of this method is still provided by Algorithm 2, but taking the lexicographic best solutions for f_1 and f_2 as the initial solutions S_1 and S_2, respectively.

With this single-phase approach there is no distinction between the type of the output solutions, unlike what happens with the method previously presented. Furthermore, the order of calculation of the solutions might be different when using the two approaches, as the example in the next section shows.

The proposed methods are suitable for being adapted in order to privilege or limit the search to a certain region, for instance in an interactive approach with the decision maker, as it allows him/her to choose which pairs of solutions to analyse, and thus which regions to further explore seeking for non-dominated solutions. More details will be given in Section 5.

3.1 Example

As an example of the application of the previous methods, let us consider a particular instance of the bicriteria $\{0,1\}$ knapsack problem. The instance has 10 items and can be formulated as

$$\begin{array}{ll} \min & -46x_1 + 36x_2 - 46x_3 - 74x_4 - x_5 - 40x_6 - 68x_7 + 44x_8 - 97x_9 - 81x_{10} \\ \min & 56x_1 + 78x_2 + 6x_3 + 77x_4 - 41x_5 - 17x_6 - 95x_7 - 87x_8 - 29x_9 - 2x_{10} \\ \text{subject to} & 62x_1 + 81x_2 + 77x_3 + 67x_4 + 12x_5 + 63x_6 + 4x_7 + 18x_8 + 10x_9 + 73x_{10} \leq 217 \\ & x \in \{0,1\}^{10} \end{array} \tag{8}$$

The images of the supported and unsupported non-dominated solutions of (8) are depicted in Figure 4. The images of the solutions represented in the plot are listed in Table 1, their order is given in the first column (of Algorithms 1 and 2).

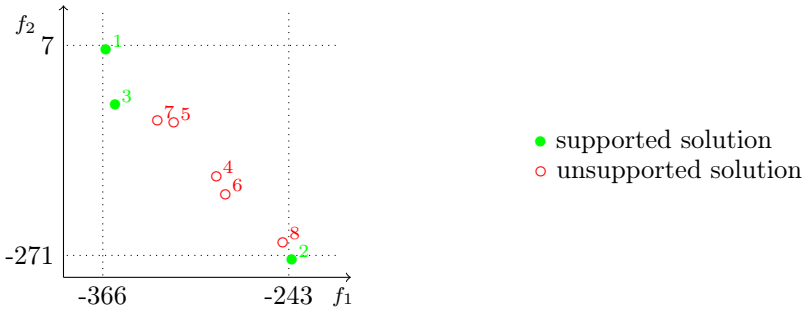


Figure 4: Efficient solutions to problem (8)

Table 1 shows the efficient solutions of this problem. The first column indicates the order of calculation of those solutions when Algorithm 1 and Algorithm 2 are applied, whereas the second column shows the same but when using only one phase, consisting of Algorithm 2. The first solutions obtained with both methods correspond to the best for each criterion. After knowing these initial solutions, the two-phase approach first computes the supported, and only afterward the unsupported. However, when using only one-phase, that is Algorithm 2, the solutions besides the best for each criterion are obtained in a different order.

Alg 1 + Alg 2	$f(s)$	Alg 2
1	$(-366, 7)$	1
2	$(-243, -271)$	2
3	$(-360, -66)$	6
4	$(-293, -161)$	3
5	$(-321, -90)$	7
6	$(-287, -184)$	5
7	$(-332, -87)$	4
8	$(-249, -248)$	8

Table 1: Efficient solutions output by Algorithms 1 and 2, and by Algorithm 2, when applied to problem (8)

3.2 Computational experiments

In the following, results for the computational evaluation of Algorithms 1 and 2 are presented. Besides comparing them with the methods in [1, 13], the main goal of these tests is to understand the sizes of problems each algorithm can solve and how the running times vary with the parameters. We denote the two-phase algorithm by `A1`, the one-phase algorithm by `A2`, the algorithm introduced by Boland et al. [1] by `BCS`, and the algorithm proposed by Ralphs et al. [13] by `RSW`. The algorithm proposed in [1] also searches for non-dominated solutions within rectangles (called boxes) that change as new solutions are obtained. For each pair of solutions defining a rectangle, two new non-dominated solutions are computed. The new solutions are the lexicographic best with respect to the two criteria, considering also particular additional constraints in order to restrict the search (and define the rectangles). The algorithm proposed in [13] differs from the one-phase approach here proposed essentially due to a different strategy of search, because in our case the reference point is changed from one iteration to the next one. All the codes were implemented in C language and using CPLEX 12.6 to solve the intermediate mixed integer programs. These codes were tested on a Dual Core AMD Opteron at 2 GHz, with 4 Gb of RAM, aiming at computing all the non-dominated solutions of bicriteria problems. Three sets of randomly generated instances were considered:

- The first set of instances consists of $\{0,1\}$ knapsack problems with two objective functions,

$$\begin{array}{ll} \min & f_1(x) = c_1 x \\ \min & f_2(x) = c_2 x \\ \text{subject to} & ax \leq W \\ & x \in \{0,1\}^n \end{array}$$

  with c_j^1, c_j^2 and a_j uniformly generated in $[-100, 100]$, W a random integer in $[100, \sum_{j=1}^n a_j]$, and $n = 10, 20, 30, 50, 100, 200$.

- The second set of instances consists of $\{0,1\}$ two-dimensional knapsack problems with two objective functions,

$$\begin{array}{ll} \min & f_1(x) = c_1 x \\ \min & f_2(x) = c_2 x \\ \text{subject to} & a_1 x \leq W_1 \\ & a_2 x \leq W_2 \\ & x \in \{0,1\}^n \end{array}$$

  with c_j^i and a_j^i uniformly generated in $[-100, 100]$, W_i a random integer in $[100, \sum_{j=1}^n a_j^i]$, $i = 1, 2$, and $n = 10, 20, 30, 50, 100, 200$.

- The third set of instances consists of complete assignment problems, again with two objective functions, with the form

$$\begin{array}{ll} \min & f_1(x) = \sum_{i=1}^n \sum_{j=1}^n c_{ij}^1 x_{ij} \\ \min & f_2(x) = \sum_{i=1}^n \sum_{j=1}^n c_{ij}^2 x_{ij} \\ \text{subject to} & \sum_{j=1}^n x_{ij} = 1, \ i = 1, \ldots, n \\ & \sum_{i=1}^n x_{ij} = 1, \ j = 1, \ldots, n \\ & x \in \{0,1\}^{n \times n} \end{array}$$

with c_{ij}^1, c_{ij}^2 uniformly generated in $[-100, 100]$, and $n = 10, 15, 20, 25, 30, 50$.

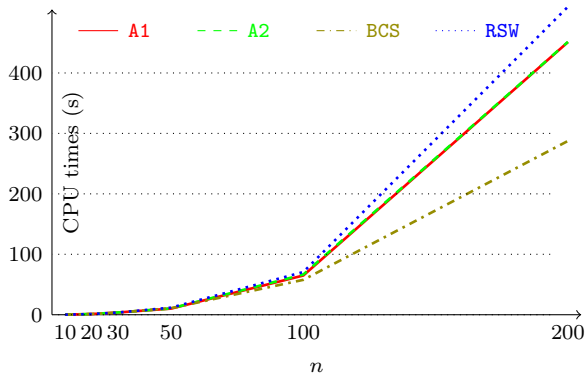
For each dimension thirty instances were generated, and the previous programs were applied.

A comparison of the average results for each code, `A1`, `A2`, `BCS` and `RSW`, in terms of running times is summarised in Table 2 for the knapsack problems, whereas Table 3 shows the number of non-dominated solutions in these instances. The CPU time for sorting the supported solutions found along phase 1 in `A1` was negligible, therefore it does not appear separately in the tables.

The CPU times obtained with all the four codes were not very different, in particular the times for the methods based on a reference point approach, `A1` and `A2`, are specially close. In fact, its correspondent lines in the picture of Table 2 (solid for `A1` and dashed for `A2`) seem to overlap. In spite of the fact that the times do not differ much, there seems to be a tendency for their faster growth with n for code `RSW` rather than with the other two approaches, and a slower growth for code `BCS`. The best times for each problem dimension are shown in bold in Table 2. The method with the best performance varies depending on the set of instances. However, `A1` outperformed `A2` in all cases. Additionally, `A1` seemed to perform better than the other methods for small problems and `BCS` for bigger size instances.

Table 2: Average CPU times (in seconds) for random knapsack problems

n	10	20	30	50	100	200
`A1`	**0.128**	1.133	3.730	**10.115**	64.928	451.078
`A2`	0.146	1.173	3.769	10.361	66.579	451.428
`BCS`	0.203	1.299	**3.599**	10.833	**57.753**	**287.522**
`RSW`	0.130	**1.110**	3.679	11.744	70.698	509.566



Besides presenting the average number of non-dominated solutions for each size, Table 3 also shows how many of those are supported (obtained in Phase 1) and how many are unsupported (obtained in Phase 2), which gives an idea of how the workload is distributed.

Table 3: Average number of solutions for random knapsack problems

n	10	20	30	50	100	200
Phase 1	5.167	9.567	12.133	18.867	37.200	69.267
Phase 2	5.100	20.633	38.467	86.700	282.767	1000.400
Total	10.267	30.200	50.600	105.567	319.967	1069.667

Table 4 shows the average CPU times for computing one non-dominated solution. In the case of code `A1` these times are split into the time for calculating one supported solution (solid line in the picture) and the time for calculating one unsupported solution (dashed line in the picture). As expected, according to these results solving phase 1 problems is easier than solving phase 2

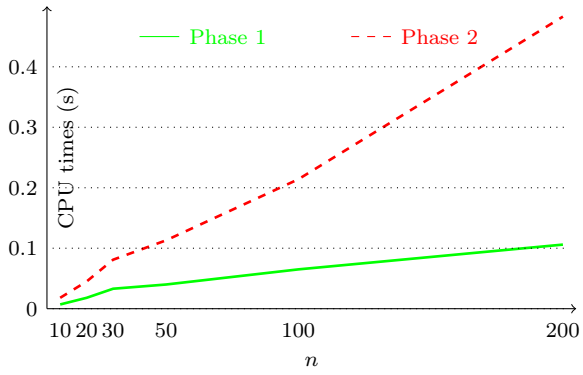
problems. Furthermore, the CPU times for computing unsupported solutions grow faster with n than the times for obtaining supported solutions. The Chebyshev problems solved by each code (in phase 2 for code `A1`) all have the same number of variables and of constraints, but are not exactly the same, given that different scalars and different reference points can be used. This fact results in slightly different running times which, together with the number of non-dominated solutions of each type, have an impact in the overall running times presented by each algorithm.

Table 4: Average CPU times (in seconds) per solution for random knapsack problems

(a) Results for all codes

n	10	20	30	50	100	200
`A1`	**0.012**	0.038	0.074	**0.096**	0.203	0.422
Phase 1	0.007	0.018	0.033	0.040	0.065	0.106
Phase 2	0.018	0.045	0.081	0.113	0.214	0.483
`A2`	0.014	0.039	0.074	0.098	0.208	0.422
`BCS`	0.020	0.043	**0.071**	0.103	**0.180**	**0.269**
`RSW`	0.013	**0.037**	0.073	0.111	0.221	0.476

(b) Results for Phases 1 and 2 of code `A1`

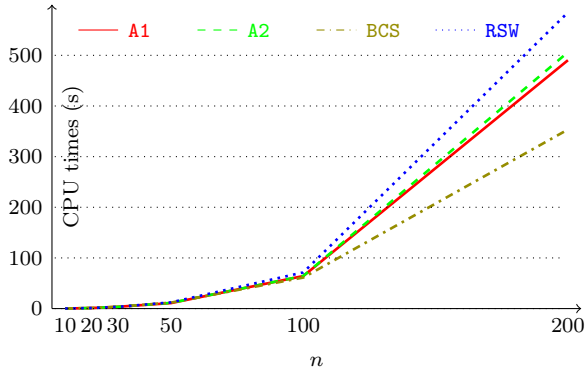


For the biggest knapsack problems, comprising 200 items, the new method with two phases, `A1`, computed 1069.667 solutions in an average time of 451.078 seconds, 69.267 of them supported solutions and 1000.400 of them unsupported. As for the method implemented only in one phase, `A2`, the same solutions were obtained in an average time of 451.428 seconds. As mentioned above, the quickest method for these instances was `BCS`, which required 287.522 seconds, in average, to solve each of them.

The running times for codes `A1`, `A2`, `BCS` and `RSW` on the two-dimensional knapsack instances are shown in Table 5, whereas Table 6 presents their number of non-dominated solutions.

Table 5: Average CPU times (in seconds) for random two-dimensional knapsack problems

n	10	20	30	50	100	200
`A1`	**0.128**	1.127	3.579	**10.844**	64.261	490.338
`A2`	0.151	1.111	3.501	11.223	64.418	506.166
`BCS`	0.206	1.209	3.512	11.787	**61.267**	**353.504**
`RSW`	0.132	**1.091**	**3.415**	12.262	71.471	584.677



Although it was only slightly slower to find each non-dominated solution for two-dimensional knapsack problems – Table 7 – than for knapsack problems with only one constraint – Table 4 –,

the number of non-dominated solutions was higher in the latter case – Table 3. Nevertheless, in general the two-dimensional problems were harder to solve than the previous knapsack instances – Table 5.

Table 6: Average number of solutions for random two-dimensional knapsack problems

n	10	20	30	50	100	200
Phase 1	4.800	8.533	10.333	18.500	29.433	45.355
Phase 2	4.800	17.900	30.167	86.300	226.100	684.387
Total	9.600	26.433	40.500	104.800	255.533	729.742

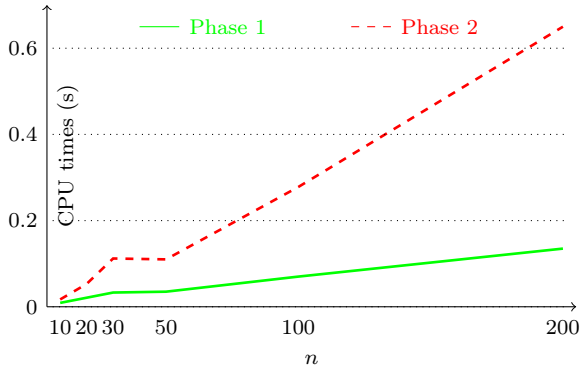
The relative performance of the methods was similar to the previous results. The code `A1` was almost always better than `A2`. `RSW` was the quickest method for problems with 20 or 30 items. However, it was still the most sensitive method to the increasing size of the problems. Like before, code `BCS` was advantageous for the biggest problems.

Table 7: Average CPU times (in seconds) per solution for random two-dimensional knapsack problems

(a) Results for all codes

n	10	20	30	50	100	200
`A1`	**0.012**	0.037	0.071	**0.103**	0.201	0.458
Phase 1	0.009	0.021	0.033	0.035	0.070	0.135
Phase 2	0.017	0.053	0.112	0.110	0.278	0.650
`A2`	0.015	0.037	0.069	0.106	0.201	0.473
`BCS`	0.020	0.040	0.069	0.112	**0.191**	**0.330**
`RSW`	0.013	**0.036**	**0.067**	0.116	0.223	0.547

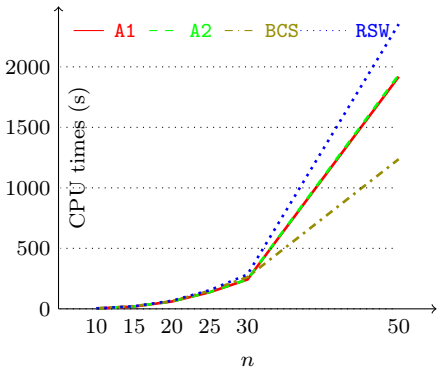
(b) Results for Phases 1 and 2 of code `A1`



The biggest instances of the two-dimensional knapsack problems, again with 200 items, were solved in less than six minutes by code `BCS`. The two- and the one- phase methods presented earlier, respectively `A1` and `A2`, found the same set of solutions in about eight minutes.

Table 8: Average CPU times (in seconds) for random assignment problems

n	10	15	20	25	30	50
`A1`	**3.222**	**19.130**	**60.816**	**137.420**	**242.352**	1918.357
`A2`	3.261	19.700	61.361	138.342	243.362	1932.813
`BCS`	4.276	22.548	63.110	149.849	262.254	**1237.120**
`RSW`	3.609	21.256	66.938	154.346	284.533	2353.239



Finally, Tables 8 to 10 show the total running times, the number of non-dominated solutions and the running times per solution of the same four codes applied to the assignment problem instances. In this case the CPU times are generally bigger for the same n and increase faster than for the previous problems. The performances of `A1` and `A2` are still very close, whereas code `RSW` is always slower than these two and the gap in the times became greater with n. Even though the number of non-dominated assignments – in Table 9 – is, in certain cases, smaller than the number of non-dominated knapsacks solutions, according to the running times the first problems are harder to solve than the latter.

Table 9: Average number of solutions for random assignment problems

n	10	15	20	25	30	50
Phase 1	7.367	12.567	17.533	22.367	26.233	48.567
Phase 2	14.933	44.667	76.267	121.800	154.700	388.033
Total	22.300	57.233	93.800	144.167	180.933	436.600

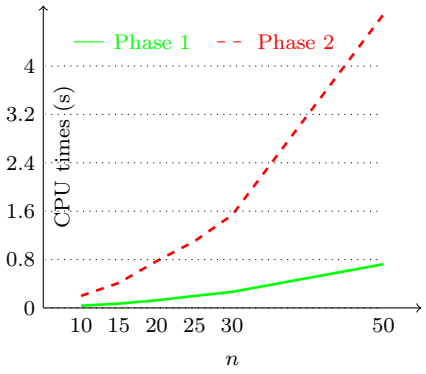
The time each code required to obtain one non-dominated solution, as well as the times to compute one supported solution or one unsupported solution with `A1`, are presented in Table 10. Although the total number of solutions in these instances is smaller than for knapsack problems and calculating unsupported solutions takes longer than calculating supported solutions, the fact that the assignment problems are more difficult to solve has a smaller impact in the total running times than in the first case. On this set of problems the subproblems generated by code `A1` were solved faster than the others. In terms of the overall running times the new methods, `A1` and `A2`, outperformed code `RSW` as well as code `BCS`, except for the bigger instances with $n = 50$.

Table 10: Average CPU times (in seconds) per solution for random assignment problems

(a) Results for all codes

n	10	15	20	25	30	50
`A1`	**0.145**	**0.338**	**0.653**	**0.961**	**1.351**	4.394
Phase 1	0.036	0.071	0.126	0.198	0.266	0.723
Phase 2	0.199	0.414	0.775	1.102	1.537	4.853
`A2`	0.147	0.348	0.659	0.967	1.357	4.427
`BCS`	0.192	0.398	0.678	1.048	1.462	**2.834**
`RSW`	0.162	0.376	0.719	1.079	1.586	5.390

(b) Results for Phases 1 and 2 of code `A1`



Finally, it should be recalled that the presented times refer to the problem of finding all the non-dominated solutions. When applied as an interactive approach the running times of these methods are smaller, depending mostly on the regions to be searched.

4 Approximating the Pareto frontier

When Phase 1 is over it outputs part of the Pareto frontier, formed only by those non-dominated solutions that are supported. This can be seen as an approximation to the Pareto frontier and in order to improve this approximation new non-dominated solutions can be added, until all have been determined or a measure of the quality of the approximation is met. Other approaches of the same type were introduced by Cohon [3], Current et al. [6], and by Hamacher and Ruhe [9] using different metrics.

In the following an error upper bound for the Pareto frontier approximations will be used. As seen in the previous section, each pair of consecutive solutions obtained along the application of Algorithm 2 provides an area to sweep, in search for new non-dominated solutions. As shown in the following, for a given problem this area is smaller the higher the number of solutions, therefore it can be used to halt the process, or compared with the area of the complete Pareto frontier (when known).

Let $\bar{P} = \{s_1, \ldots, s_\ell\}$ be a set of non-dominated solutions, sorted in a way that $f_1(s_1) < \ldots < f_1(s_\ell)$. Let us denote by $A(s_i, s_{i+1})$ the area of the region bounded by $f(s_i)$ and $f(s_{i+1})$, i.e., $A(s_i, s_{i+1}) = (f_1(s_{i+1}) - f_1(s_i))(f_2(s_i) - f_2(s_{i+1}))$, $i = 1, \ldots, \ell - 1$. Then, the area associated with $\bar{P}$ is defined by

$$A(\bar{P}) = \sum_{i=1}^{\ell-1} A(s_i, s_{i+1}).$$

Proposition 4. *Let $\bar{P} = \{s_1, \ldots, s_\ell\}$ be a set of non-dominated solutions, where $f_1(s_1) < \ldots < f_1(s_\ell)$, and s' be another non-dominated solution such that $f_1(s_1) < f_1(s') < f_1(s_\ell)$. Then, $A(\bar{P}) > A\left(\bar{P} \cup \{s'\}\right)$.*

Proof. Assume j is the greatest index such that $f_1(s_j) < f_1(s')$, and that $f_1(s') < f_1(s_{j+1})$. By definition,

$$\begin{array}{rcl} A\left(\bar{P} \cup \{s'\}\right) & = & \sum_{i=1}^{j-1} A(s_i, s_{i+1}) + A(s_j, s') + A(s', s_{j+1}) + \sum_{i=j+1}^{\ell-1} A(s_i, s_{i+1}) \\ & = & A(\bar{P}) + A(s_j, s') + A(s', s_{j+1}) - A(s_j, s_{j+1}) \end{array}$$

Next it is shown that $A(s_j, s') + A(s', s_{j+1}) < A(s_j, s_{j+1})$. By assumption and because s' is a non-dominated solution,

$$f_1(s_j) < f_1(s') < f_1(s_{j+1}) \text{ and } f_2(s_{j+1}) < f_2(s') < f_2(s_j).$$

Therefore,

$$\begin{array}{rcl} A(s_j, s_{j+1}) & = & (f_1(s_{j+1}) - f_1(s_j))(f_2(s_j) - f_2(s_{j+1})) \\ & = & A(s', s_{j+1}) + A(s_j, s') + (f_1(s_{j+1}) - f_1(s'))(f_2(s_j) - f_2(s')) + \\ & & (f_1(s') - f_1(s_j))(f_2(s') - f_2(s_{j+1})) \\ & > & A(s', s_{j+1}) + A(s_j, s') \end{array}$$

as we wanted to show. □

Let P be the Pareto frontier of a BCILP and $\bar{P} \subseteq P$ be a subset of P. Let also the error associated with $\bar{P}$ be defined by $e(\bar{P}) = A(\bar{P}) - A(P)$.

Corollary 4.1. *Under the conditions of Proposition 4,* $e(\bar{P}) > e\left(\bar{P} \cup \{s'\}\right)$.

Proposition 4 states that the area associated with subsets of non-dominated solutions decreases as more solutions are computed. Therefore, it can be used as a measure of the quality of the current subset of the Pareto frontier. Thus, the application of the method can halt if that area is smaller than a maximum allowed tolerance for the quality of the computed set of non-dominated solutions. By doing so the area of potential locations of new interesting non-dominated images is reduced. Moreover, a "well-distributed" set of such solutions can be computed. This is done by selecting the pair of consecutive images that corresponds to the widest area. This procedure is outlined in Algorithm 3.

Algorithm 3: Approximate non-dominated solutions determination within the duality gap defined by (S_1, S_2)

```
   /* (S1, S2) is a pair of adjacent solutions (f1(S1) < f1(S2))          */
   /* ε is the maximum allowed area of a set of non-dominated solutions  */
   /* X is an auxiliary set that stores the unscanned pairs of non-dominated
      solutions                                                          */
 1 s1 ← S1
 2 s2 ← S2
 3 L ← {s1, s2}
 4 X ← {(s1, s2)}
 5 while X ≠ ∅ and A(X) ≥ ε do
 6     (s1, s2) ← pair of solutions in X with the most distant images
 7     X ← X − {(s1, s2)}
 8     w1 ← f2(s1) − c2*; w2 ← f1(s2) − c1*
 9     s ← best solution to (6)
10     if s is defined then
11         L ← L ∪ {s}
12         if f1(s) ≠ f1(s2) then X ← X ∪ {(s1, s)}
13         if f2(s) ≠ f2(s1) then X ← X ∪ {(s, s2)}
```

4.1 Computational experiments

A new set of computational tests was performed, in order to evaluate the speed of convergence of the proposed method, and thus evaluating its ability to approximate the Pareto frontier of a problem. These tests comprised the same instances listed in Section 3.2 and codes `A1` and `BCS`. The stopping criterion was a given upper bound on the relative gap between the area associated with the exact Pareto frontier and the area associated with the set of non-dominated solutions output by the codes.

Two evaluating measures were recorded: the number of non-dominated solutions computed by the approximating method, and the running time required to obtain them. Such values are shown in Tables 11 to 13 for a maximum relative gap tolerance of 10% and 20%.

Table 11: Average results to approximate Pareto frontiers for random knapsack problems

	A1				BCS				
	Gap $< 10\%$		Gap $< 20\%$		Gap $< 10\%$		Gap $< 20\%$		
n	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)	(3)
10	0.122	10.133	0.124	10.133	0.212	10.267	0.196	10.267	10.267
20	1.190	30.200	1.143	30.200	1.294	30.200	1.310	30.200	30.200
30	3.818	50.600	3.813	50.600	3.609	50.600	3.568	50.600	50.600
50	10.678	105.567	10.219	105.567	10.938	105.567	10.805	105.567	105.567
100	64.734	319.967	62.430	319.967	58.637	319.723	58.242	319.723	319.967
200	239.393	867.767	198.621	787.933	216.144	941.459	193.047	878.552	1069.667

Legend: (1) CPU time (in seconds); (2) # of computed non-dominated solutions; (3) Total # of non-dominated solutions.

When compared to the corresponding results when codes are applied in an exact manner we observe that some CPU times for obtaining a set of non-dominated solutions with a gap smaller than 10% were smaller than the former. This was due to the additional operations that are performed for checking the stopping criterion, and is less relevant for more difficult problems.

In most of the smaller instances of the knapsack and the two-dimensional knapsack problems all the non-dominated solutions have been found for both a 10% and a 20% gaps. These results were different for the class of assignment problems, which were harder to solve. In this case only part of the Pareto frontier was obtained and there was a clearer decrease of the CPU times.

Table 12: Average results to approximate Pareto frontiers for random two-dimensional knapsack problems

	A1				BCS				
	Gap $< 10\%$		Gap $< 20\%$		Gap $< 10\%$		Gap $< 20\%$		
n	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)	(3)
10	0.147	9.600	0.142	9.600	0.197	9.600	0.191	9.600	9.600
20	1.237	26.433	1.180	26.433	1.224	26.433	1.254	26.433	26.433
30	3.916	40.500	3.711	40.500	3.530	40.500	3.569	40.500	40.500
50	11.601	104.800	11.341	104.800	12.004	104.800	12.649	104.800	104.800
100	66.914	255.533	65.552	255.533	63.128	255.533	62.467	255.533	255.533
200	276.116	615.133	230.724	561.633	261.884	635.499	229.560	590.551	729.742

Legend: (1) CPU time (in seconds); (2) # of computed non-dominated solutions; (3) Total # of non-dominated solutions.

In general the conclusions are similar to what was drawn in Section 3, when comparing the performances of codes `A1` and `BCS`. The first of them was faster (and computed more solutions) for small and medium size instances, whereas the second became the fastest of both codes for the bigger instances. This was more obvious for the assignment problems class.

For the class of assignment problems, the running times of code `A1` decreased from 39% to 52%, to compute between 90% and 85% of all the non-dominated solutions, for a 10% gap. For a 20% gap,

the decrease in the running times varied between 44% and 59%, for knowing from 84% to 79% of the Pareto frontier. The same values for code `BCS`, with a 10% gap, were of a time decrease between 37% to 15% for computing between 89% to 87% of the Pareto frontier. When the maximum gap is 20%, the times reduced between 45% to 21% to obtain about 83% of the non-dominated solutions. It is worth remarking that the times decrease of approximating the Pareto frontier was bigger for code `A1` rather than for `BCS`.

Table 13: Average results to approximate Pareto frontiers for random assignment problems

	`A1`				`BCS`				
	Gap $< 10\%$		Gap $< 20\%$		Gap $< 10\%$		Gap $< 20\%$		
n	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)	(3)
10	1.981	20.033	1.803	18.633	2.699	19.867	2.364	18.600	22.300
15	10.324	48.567	8.836	45.400	15.319	51.300	13.072	47.633	57.233
20	33.918	81.133	28.535	74.600	44.646	83.633	39.296	77.967	93.800
25	74.764	124.133	62.536	115.400	114.673	125.267	103.547	118.500	144.167
30	131.963	155.100	105.924	142.533	199.828	157.233	181.081	148.600	180.933
50	931.250	370.200	787.233	343.800	1042.708	380.267	975.303	362.200	436.600

Legend: (1) CPU time (in seconds); (2) # of computed non-dominated solutions; (3) Total # of non-dominated solutions.

5 Interactive approach

In this section we intend to make a progressive and selective search of non-dominated solutions, intending to avoid the effort in the calculation of "areas" identified by the DM as most interesting. This is done taking into account the possible feasible values of the objective functions in those "areas".

Algorithm 1 together with Algorithm 2, Algorithm 2 by itself, ad Boland et al.'s algorithm can be used as interactive approaches. In this section we will exemplify the different possibilities. We propose two approaches based on Algorithm 1 combined with Algorithm 2. In both, the first part of the interactive procedures here suggested follow the framework of the procedure proposed in [4]. Let us start by an approach using a NISE-like approach in an interactive way for searching in a progressive and selective way the supported non-dominated solutions.

In order to illustrate this process, let us consider the following $\{0,1\}$ knapsack problem with ten items and two objective functions to be maximised.

$$\begin{array}{ll} \max & 91x_1 + 9x_2 - 66x_3 + 32x_4 - 13x_5 + 99x_6 + 62x_7 - 98x_8 - 50x_9 + 26x_{10} \\ \max & 29x_1 - 14x_2 + 73x_3 + 87x_4 - 90x_5 + 69x_6 - 16x_7 + 56x_8 - 61x_9 + 71x_{10} \\ \text{subject to} & 62x_1 + 21x_2 + x_3 + 57x_4 + 93x_5 + 34x_6 + 89x_7 + 59x_8 + 39x_9 + 90x_{10} \leq 409 \\ & x \in \{0,1\}^{10} \end{array} \tag{9}$$

The set of all efficient solutions for this problem is shown in Figure 5.

A NISE-like approach starts by the calculation of solutions 1 and 2, depicted in Figure 6a. Maximising the weighted-sum of f_1 and f_2, such that the line passing through points 1 and 2 is a constant cost line of such a weighted-sum, the solution 3 is obtained – see Figure 6b. Now, a

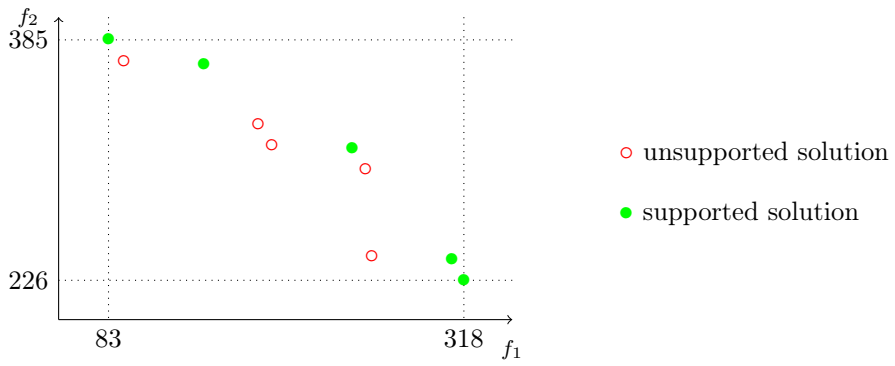


Figure 5: Efficient solutions to problem (9)

dialogue phase follows, asking the DM whether he is interested in the search for supported non-dominated solutions between 2 and 3, or between 1 and 3, or both. In the first case the point 5 is found, in the second it will be point 4, and both of them in the last case – Figure 6c.

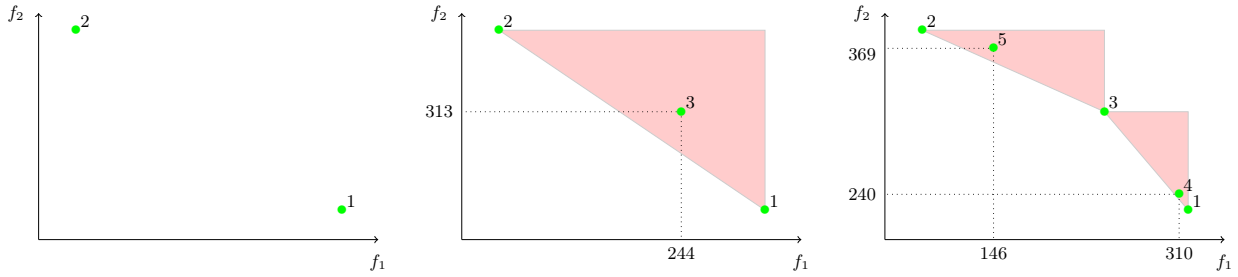


Figure 6: Interactive calculation of supported non-dominated solutions to problem (9) with the two-phase method

Note that the DM's decision is based on the knowledge of the feasible intervals of the objective function values. This type of dichotomic questions and the speed of calculation of the supported non-dominated solutions contributes to select, in an easy way, the "areas" where phase two will be used to search unsupported non-dominated solutions minimising a weighted Chebyshev distance. Note that the selection of the first phase is crucial because the second procedure is much more time consuming, as seen before.

The second way of using phase 1 of the two-phase algorithm is to start by calculating the whole set of supported solutions. Then, in a dialogue phase, the DM can identify visually the duality gaps where the second phase search should be done. The calculation in the duality gaps, between two current adjacent supported non-dominated solutions, in order to start the search of unsupported non-dominated solutions, is done by minimising a weighted Chebyshev distance to the "ideal solution" associated with the considered duality gap. As explained earlier, the weights are chosen such that the constant cost contours related to the weighted Chebyshev metric are proportional to the intervals associated with the two adjacent supported non-dominated solutions

defining the duality gap under consideration. Of course, here again the search can be complete or can halt according to some pre-defined bound, as explained in Section 4.

We now exemplify the application of the second version of the interactive procedure above referred to, based on the methods presented in Section 3, but driven by a DM to guide the search for non-dominated solutions of this problem.

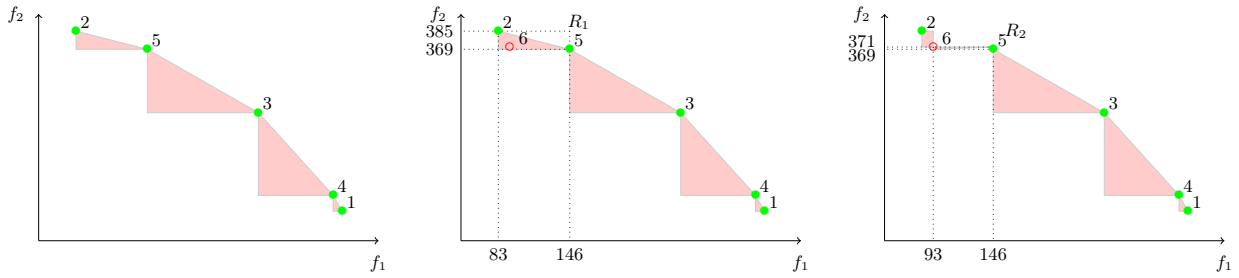


Figure 7: Interactive calculation of supported and unsupported non-dominated solutions to problem (9) with the two-phase method

First we consider the two-phase method, consisting of a first phase with Algorithm 1, followed by a second phase with Algorithm 2. Suppose that all supported solutions have been computed and that the interactivity is only applied with Algorithm 2. Then the list of images of the supported solutions, points 1 to 5 in Figure 7a (namely, $(318, 226)$, $(83, 385)$, $(244, 313)$, $(310, 240)$, $(146, 369)$), is presented to the DM. He/she is also informed that other non-dominated solutions may exist in the duality gaps between adjacent pairs of these solutions, namely the shaded areas in that plot. Assume that, when shown this information, the DM asks to find (if possible) more non-dominated solutions in the region defined by points 2 and 5. An augmented Chebyshev metric with $R_1 = (146, 385)$ as the reference point – in Figure 7b – is considered in problem (6). Then, the solution with image $(93, 371)$, given by point 6 in Figure 7b, is output. Two new regions are then defined, one associated with points 2 and 6, and another associated with points 6 and 5. Proceeding with the search, and now using $R_2 = (146, 371)$ as the reference point, leads to a previous solution (corresponding to point 5 or to point 6) – see Figure 7c – and, thus, the DM is told that there are no other non-dominated solutions with an image in the region between points 5 and 6.

If the DM wishes to continue searching, a new region has to be chosen, which can be the region limited by points 2 and 6, 5 and 3, 3 and 4, or 4 and 1. Otherwise, the process halts. Note, again, that the selection of the first phase is crucial because the second procedure is much more time consuming, as seen before.

It should be noted that the search can change from the first phase to the second phase between two current adjacent solutions obtained in the first phase (i.e., that may not be true adjacent solutions). For example, after obtaining the points 1, 2 and 3, in Figure 6b, the DM's next step may be to search between points 2 and 3 using the second phase of the algorithm.

We now exemplify the application of a similar interactive procedure based only on the one-phase method outlined in Algorithm 2, guided by a DM. The departure points are the non-dominated solutions that optimise individually each criterion. These are the first two solutions saved by the system and have objective function values: $(318, 226)$ and $(83, 385)$ – represented by 1 and 2 in Figure 8a. The ideal solution for this problem is $R_1 = (318, 385)$ and this is the trivial starting reference point. Then problem (6) is solved for these parameters and a new solution is computed, its objective functions vector being $(244, 313)$ – point 3 in Figure 8b.

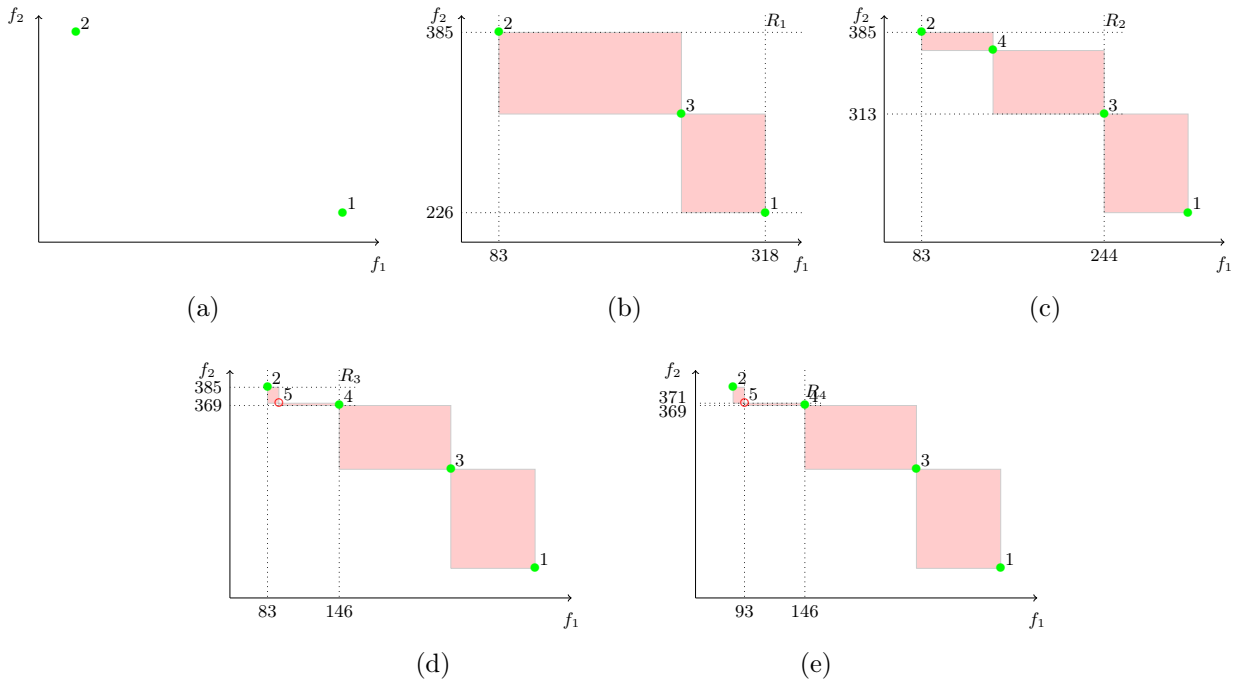


Figure 8: Interactive calculation of non-dominated solutions to problem (9)

The knowledge of point 3 allows to restrict the search region for non-dominated solution images to the two shaded areas in Figure 8b. Suppose the DM is then requested to choose one of those regions to continue the search and consider that this chosen region is bounded by points 2 and 3, that is, by $(83, 385)$ and $(244, 313)$. This consists of solving (6) again, considering $R_2 = (244, 385)$ as the new reference point, and the next non-dominated solution is obtained. Its objective function values are $(146, 369)$ – which is point 4 in Figure 8c and the shaded areas in that plot represent the regions that may contain images of non-dominated solutions.

Assuming that the DM continues the search within the region limited by points 2 and 4, the image of the next non-dominated solution is $(93, 371)$ – point 5 in Figure 8d. Once again the search can be restricted to the shaded regions in that plot. Continuing, using now $R_3 = (146, 371)$ as the reference point, leads to an already known solution (either point 4 or 5) – Figure 8e. The DM can halt the process as soon as he/she is satisfied. If the he/she wants to continue the search, he/she is then informed that no other non dominated solutions can be found in that region, so he/she can

either choose to halt the process or else to proceed with the decision process from a different starting reference point. Namely the regions delimited by points 1 and 3, points 3 and 4, and points 2 and 5 could still be further explored in order to find new non-dominated solutions.

The method introduced by Boland et. al [1] can also be implemented in interaction with a DM who points the region to be searched, according to his/her preferences. Such implementation is now illustrated, when solving problem (9). Again the first two solutions to be found and presented to the DM are the lexicographic best for each criterion. Then, two new constrained problems are solved, considering the maximisation is sought, and the solutions with images 3 and 4 on Figure 9b are found. The knowledge of these two new solutions allows to partition the search space into two rectangular regions/boxes, one bounded by points 2 and 3, and another bounded by points 4 and 1, both shaded on Figure 9b.

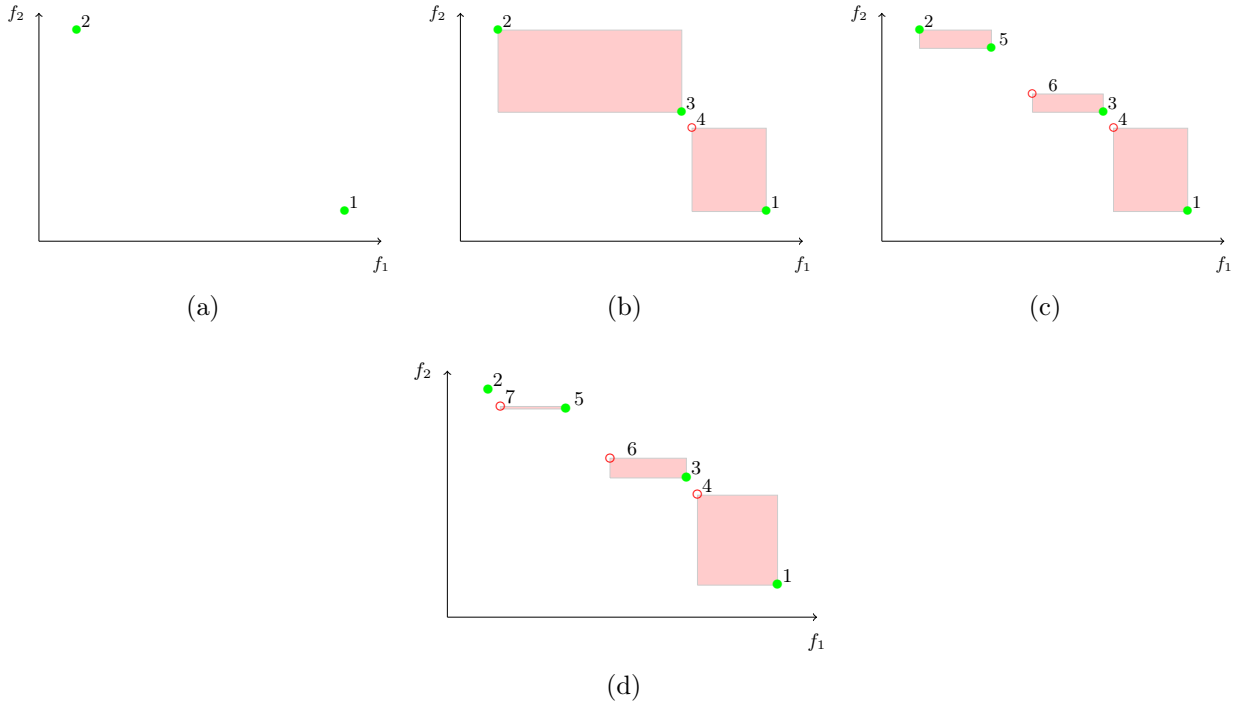


Figure 9: Interactive calculation of non-dominated solutions to problem (9) with Boland et al.'s method

Suppose that, when presented this information, the DM decides to search for more non-dominated solutions in the first of those two boxes. Then a new iteration of the method by Boland et al. is applied, thus computing the solutions with points 5 and 6 on Figure 9c as the images. Now the DM is informed that there still may exist other non-dominated points in the shaded box, bounded by points 2 and 5, 6 and 4, or 4 and 1. Assuming the DM wants to continue searching in the first of them, only one new solution is found, corresponding to point 7. Next we suppose the DM wishes to halt the calculation, even though he/she could continue searching for solutions with images within the boxes bounded by 7 and 5, 6 and 3 or 4 and 1.

First, we have to emphasise that at any moment of the search the DM can say that he is interested in the complete calculation of a subset of non-dominated solutions corresponding to a more or less wide area of the objective function space. In these circumstances, this can be done automatically (one shot) using the generating approach proposed above. Secondly, for such an area it is also possible to use the approximate algorithm proposed in the previous section to calculate an approximation of those solutions. Additionally, and as mentioned earlier, in such an interactive procedure phase 1 can be replaced by phase 2 at any time the DM finds convenient, as long as two current adjacent non-dominated solutions are given. Also, other searching processes, based on different search regions, could be tested. For instance, the combination of using lower bounds for the objective functions with the search for the best solutions with respect to a Chebyshev distance to a reference point, like the ideal point.

6 Conclusions

The non-dominated solutions of BCILP can be classified into supported and unsupported, the latter type of solutions being harder to compute. For instance, they cannot be obtained by one of the most common multicriteria approaches, WSMs. In this paper a method was developed based on a weighted Chebyshev metric with respect to a reference point, which is associated with the search for non-dominated solutions the images of which lie on a rectangle defined by this metric and a given pair of non-dominated solutions. This method can be used to find all the solutions or can be combined with a WSM in a two-phase approach. These two versions were adapted with the goals of calculating an approximation of the Pareto frontier up to a given tolerance or being used interactively.

When used to find all non-dominated solutions of bicriteria random knapsack, two-dimensional knapsack and assignment instances, computational experiments revealed that the two versions of the proposed method can compete with advantage with the algorithm in [13], whereas the presented methods were faster than the algorithm in [1] for the smallest instances and slower for the biggest. Furthermore, bicriteria knapsack problems with 200 items were solved in an average CPU time of 451.078 seconds by the two-phase method, in 451.428 seconds when using only one phase, whereas bicriteria two-dimensional problems also with 200 items were solved in 490.338 and in 506.166 seconds, and bicriteria 50×50 assignment problems were solved in 1918.357 and in 1932.813 seconds, both with the same codes, respectively.

A version of the introduced method for approximating the Pareto frontier was also proposed. In this case, and for random assignment problem instances of size up to 50×50, it was shown that a set of non-dominated solutions with an area that differs at most 10% from the exact could be found in less 39% to 52% of the running time required to find the whole Pareto frontier, and in less 44% to 59% when the relative gap is of 20%. Given the obtained results, a two-phase implementation of Boland et al.'s method [1] is expected to enhance the original for big problems.

The WSM together with the proposed Chebyshev approach (Algorithm 2), the former Chebyshev

approach alone and the Boland's algorithm can also be adapted as interactive procedures. In the last section of the paper we exploited them to create progressive and selective interactive procedures for searching non-dominated solutions. It was concluded that the methods combining the WSM with Algorithm 2 are the most efficient because the WSM is used to reduce the scope of the search and the Chebyshev approach is used just in a second phase for very focused search. Remember that, as we have seen in Section 3, the WSM calculation phase is much faster than the one based on a Chebyshev approach.

Acknowledgments This work was partially supported by Fundação para a Ciência e a Tecnologia (FCT) under projects PEst-C/EEI/UI0308/2011 and PTDC/EEA-TEL/101884/2008. The work of Marta Pascoal was also partially funded by the FCT under grant SFRH/BSAB/113683/2015.

References

[1] N. Boland, H. Charkhgard, and M. Savelsbergh. A criterion space search algorithm for biobjective integer programming: The balanced box method. *INFORMS Journal on Computing*, 2:558–584, 2015.

[2] J. Clímaco and M. Pascoal. Multicriteria path and tree problems: discussion on exact algorithms and applications. *International Transactions in Operational Research*, 19:63–98, 2012.

[3] J. Cohon. *Multiobjective programming and planning*. Academic Press, New York, 1978.

[4] J. Coutinho-Rodrigues, J. Clímaco, and J. Current. An interactive bi-objective shortest path approach: searching for unsupported non dominated solutions. *Computers & Operations Research*, 26:789–798, 1999.

[5] J. Current and M. Marsh. Multiobjective transportation network design and routing problems: Taxonomy and annotation. *European Journal of Operational Research*, 65:4–19, 1993.

[6] J. Current, C. ReVelle, and J. Cohon. An interactive approach to identify the best compromise solution for two objective shortest path problems. *Computers & Operations Research*, 17:187–198, 1990.

[7] K. Dächert, J. Gorski, and K. Klamroth. An augmented weighted Tchebycheff method with adaptively chosen parameters for discrete bicriteria optimization problems. *Computers & Operations Research*, 39:2929–2943, 2012.

[8] C. Ferreira. *Problemas de localização e distribuição multicritério – aproximações e estudo de alguns casos com implicações ambientais*. PhD thesis, Univ. of Aveiro, 1998.

[9] H. Hamacher and G. Ruhe. On spanning tree problems with multiple objectives. *Annals of Operations Research*, 52:209–230, 1994.

[10] H. W. Hamacher, C. R. Pedersen, and S. Ruzika. Finding representative systems for discrete bicriterion optimization problems. *Operations Research Letters*, 35:336–344, 2007.

[11] G. Mavrotas. Effective implementation of the ε-constraint method in multi-objective mathematical programming problems. *Applied Mathematics and Computation*, 213:455–465, 2009.

[12] A. Raith and M. Ehrgott. A comparison of solution strategies for biobjective shortest path problems. *Computers & Operations Research*, 36:1299–1331, 2009.

[13] T. K. Ralphs, M. J. Saltzman, and M. M. Wiecek. An improved algorithm for solving biobjective integer programs. *Annals of Operations Research*, 147:43–70, 2006.

[14] S. Steiner and T. Radzik. Computing all efficient solutions of the biobjective minimum spanning tree problem. *Computers & Operations Research*, 35:198–211, 2008.

[15] R. Steuer. *Multiple Criteria Optimization: Theory, Computation and Applicattion.* John Wiley & Sons, Inc., New York, 1986.

[16] R. Steuer and E.-U. Choo. An interactive weighted Tchebycheff procedure for multiple objective programming. *Mathematical Programming*, 26:326–344, 1983.

[17] M. Visée, J. Teghem, M. Pirlot, and E.L. Ulungu. Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem. *Journal of Global Optimization*, 12:139–155, 1998.