João Clímaco and Marta Pascoal

A new method to determine unsupported non-dominated solutions in multicriteria integer linear programming - a reference point approach

No. 3 2012

ISSN: 1645-2631
A new method to determine unsupported non-dominated solutions in multicriteria integer linear programming - a reference point approach

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
E-mail: marta@mat.uc.pt

February 18, 2012

Abstract: In this paper we introduce a method for finding both supported and unsupported non-dominated solutions of a multicriteria integer linear program (MCILP). This consists of two phases, in the first a weighted-sum method is used that finds only the supported solutions of the problem, in the second the Chebyshev distance to a reference point is minimized in order to sweep the region between any pair of adjacent supported solutions and thus find the unsupported solutions. Computational experiments are discussed.

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

1 Introduction

Let us consider the multicriteria integer linear program (MCILP) with $k$ objective functions, $n$ variables and $m$ constraints:

$$\begin{align*}
\text{min} & \quad f_1(x) = c_1^T x \\
\text{min} & \quad f_2(x) = c_2^T x \\
& \quad \vdots \\
\text{min} & \quad f_k(x) = c_k^T x \\
s. t. & \quad Ax \leq b \\
& \quad x \geq 0 \\
& \quad x \in \mathbb{Z}^n
\end{align*}$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $c_r \in \mathbb{R}^n$, for any $r = 1, \ldots, k$. We assume that $S = \{ x \in \mathbb{Z}^n : Ax \leq b, \ x \geq 0 \}$ is nonempty and bounded. For easiness of presentation we consider that all variables are integer, however the method presented in the following still applies to multicriteria mixed integer linear problems.

A solution $x \in S$ of problem (1) is efficient if and only if there is no other $x' \in S$ 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 focus on the determination of non-dominated solutions, which means that if there are alternative efficient solutions with the same image then only one of them is calculated. Despite these two standard definitions, in the remaining of the paper we might use the terms efficient solution and non-dominated solution interchangeably. All the non-dominated solutions of a MCILP form the problem’s Pareto frontier. We distinguish two types of non-dominated solutions,
supported non-dominated solutions, which are non-dominated solutions that are optimal solutions of a (single-criterion) weighted sum problem (WSP)

\[
\min_{x \in S} \left\{ \sum_{r=1}^{k} w_r f^r(x) : w_r \geq 0, \ r = 1, 2, \ldots, k \right\}; \tag{2}
\]

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

While on the one hand supported non-dominated solutions can be found by means of solving a single-criterion WSP (if there exists an efficient algorithm for the problem), on the other finding unsupported solutions is usually a more difficult task and this type of solutions remain to characterize. Because of these differences the calculation of the two types of solutions is often done in two phases, first the supported solutions are computed, afterwards the search for unsupported solutions continues by looking for solutions the images of which lie within the duality gaps formed by two adjacent supported solutions. For further details see [1, 5, 7]. There has been some work on methods for the first phase, however, to our knowledge, the problem of finding unsupported solutions has merited little attention in the literature.

The search of solutions within duality gaps has been investigated since Current et al. [4]. Two main approaches have been proposed, namely using side constraints to obtain a sequence of “easier” problems, and using an algorithm able to rank the \( K \) best solutions of the problem to sweep that area. Whenever ranking solutions can be made fast the latter approach has revealed to be quite efficient. The bicriteria shortest paths problem or the bicriteria spanning trees problem are two of those cases, in the first case the search can be done quite easily, [3], the second case is harder but still manageable for problems that are not too big, [6]. However, in the general case it can even be more difficult to rank solutions, and therefore more difficult to perform the search. The method presented in the following has a broader application, and the empirical experiments show it can be used in practice.

In the next section the second phase method for computing the Pareto frontier of a MCILP is described. A weighted-sum method (WSM) for finding the supported solutions is outlined, and then an algorithm for computing all the unsupported solutions within two adjacent supported solutions is proposed. At the end of the section an example of the application of the algorithm is given, whereas Section 3 is devoted to computational results.

2 Algorithmic approach

Figure 1 illustrates the different types of solutions of a MCILP. Among the non-dominated solutions we distinguish the supported and the unsupported. In the present section we describe a method for computing the Pareto frontier of a MCILP. The first part of the method proposed in the following is responsible for the supported solutions calculation, while the remaining non-dominated solutions,
i.e., the unsupported, are calculated during the second part. In order to simplify the presentation we only consider the bicriteria case, with \( k = 2 \), however the results can be easily extended to a general \( k \).

The WSM calculates the non-dominated solutions by solving a sequence of WSPs and updating their parameters according to the solutions found. Descriptions of the WSM can be found in [3] or [7]. 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, calculating a new one by considering a single criteria WSP. The goal is to determine whether there exist further supported non-dominated solutions within the triangle \( f(s_1)c^*f(s_2) \), where \( c^* = (c_1^*, c_2^*) \) and \( c_i^* = f_i(s_i), i = 1, 2 \) – Figure 2.(a). The new solution, \( s_3 \) depicted in Figure 2.(b), is the best regarding the minimization of the objective function

\[
w_1f_1(s) + w_2f_2(s),
\]

with \( w_1 = f_2(s_1) - f_2(s_2), w_2 = f_1(s_2) - f_1(s_1) \), and because the solutions are restricted to the mentioned triangle the problem can be formulated as

\[
\begin{align*}
\min & \quad w_1f_1(s) + w_2f_2(s) \\
\text{s. t.} & \quad s \in S
\end{align*}
\]

(3)

It should be added that the weights \( w_1, w_2 \) can also be normalized. This approach is inspired in the noninferior set estimation (NISE) method [2]. The process is illustrated in Figure 2. A summary of the whole phase 1 method is included in Algorithm 1.

**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 calculated pairs of solutions

\[
\begin{align*}
s_1 & \leftarrow \text{lexicographic best solution with respect to } (f_1, f_2) \\
s_2 & \leftarrow \text{lexicographic best solution with respect to } (f_2, f_1) \\
L & \leftarrow \{(s_1, s_2)\} \\
X & \leftarrow \{(s_1, s_2)\} \\
\text{While } & \ X \neq \emptyset \text{ Do} \\
(s_1, s_2) & \leftarrow \text{element in } X; X \leftarrow X - \{(s_1, s_2)\} \\
w_1 & \leftarrow f_2(s_1) - f_2(s_2); w_2 \leftarrow f_1(s_2) - f_1(s_1)
\end{align*}
\]
Figure 2: Phase 1, weighted-sum method: (a) initial supported solutions; (b) supported solution obtained from the previous pair of solutions

\[ s \leftarrow \text{best solution to (3)} \]
\[ \text{If } s \text{ is defined Then} \]
\[ L \leftarrow L \cup \{s\} \]
\[ \text{If } f_1(s) \neq f_1(s_2) \text{ Then } X \leftarrow X \cup \{(s_1, s)\} \]
\[ \text{If } f_2(s) \neq f_2(s_1) \text{ Then } X \leftarrow X \cup \{(s, s_2)\} \]
\[ \text{EndIf} \]
\[ \text{EndWhile} \]

Notice that Algorithm 1 calculates at most one new supported solution given a pair of consecutive supported solutions. This process can also be adapted in order to privilege or limit the search to a certain region, for instance in an interactive method.

Figure 3: Phase 2, computing unsupported solutions: (a) initial search area (shaded); (b) areas to scan (shaded) after the unsupported solution \( s_3 \) is calculated in search for possible non-dominated solutions

The goal of the second part of the method is to search for other solutions lying within the duality gaps formed by pairs of adjacent supported solutions obtained by Algorithm 1. Non-supported solutions are not the optimal solution of any WSP, and thus 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 every criteria, with respect to a certain weighted Chebyshef distance [8].

**Theorem 1.** Let \( s_i \) be the optimal solution with respect to \( f_i \), and \( c_i^* = f_i(s_i) \) represent the optimal value for each criteria, \( i = 1, 2 \). If \( s \) is a non-dominated solution of (1) then it is the optimal
A consequence of this result is that any unsupported solution can be found by using a Chebyshev metric with adequate parameters. Taking now \( s_1 \) and \( s_2 \) as two adjacent supported solutions, we want to apply Theorem 1 in order to search for unsupported solutions within the duality gap given by the shaded area in Figure 3.(a). This is equivalent to look for non-dominated solutions within the rectangle defined by the points \( c^* \), \( f(s_1) \), \( (f_1(s_2), f_2(s_1)) \), \( f(s_2) \), where \( c^* = (c_1^*, c_2^*) \) and \( c_i^* = f_i(s_i) \), \( i = 1, 2 \).

If the weights are \( w_1 = f_2(s_1) - f_2(s_2) \), \( w_2 = f_1(s_2) - f_1(s_1) \), and the reference point is \( c^* \), given above, then solving the problem

\[
\min \ s.t. \quad \begin{align*}
& w_1 |f_1(s) - c_1^*|, w_2 |f_2(s) - c_2^*| \\
& s \in S
\end{align*}
\]

with the weighted Chebyshev metric, finds the best solution with respect to an objective function the level lines of which are proportional to the latter rectangle taking \( c^* \) as the reference point, as illustrated in Figure 3.(a) and shown in Proposition 1.

**Proposition 1.** Given \( s_1, s_2 \) a pair of adjacent supported non-dominated solutions of (3), the optimal solution of (4) is the best solution with respect to lines proportional to the rectangle with vertices \( c^* \), \( f(s_1) \), \( (f_1(s_2), f_2(s_1)) \) and \( f(s_2) \).

**Proof.** Let \( s_1, s_2 \) be a pair of solutions of (3), and \( s^* \) be the optimal solution in \( S \) regarding \( T \). Assume there is another solution \( s \in S \) such that

\[
\begin{align*}
& c_1^* \leq f_1(s) < f_1(s^*) \\
& c_2^* \leq f_2(s) < f_2(s^*)
\end{align*}
\]

Then,

\[
\begin{align*}
& 0 \leq f_1(s) - c_1^* < f_1(s^*) - c_1^* \\
& 0 \leq f_2(s) - c_2^* < f_2(s^*) - c_2^* \Rightarrow \begin{align*}
& w_1 |f_1(s) - c_1^*| < w_1 |f_1(s^*) - c_1^*| \\
& w_2 |f_2(s) - c_2^*| < w_2 |f_2(s^*) - c_2^*|
\end{align*}
\]

and therefore

\[
\max\{w_1 |f_1(s) - c_1^*|, w_2 |f_2(s) - c_2^*|\} < \max\{w_1 |f_1(s^*) - c_1^*|, w_2 |f_2(s^*) - c_2^*|\},
\]

which implies that \( s^* \) is not optimal with respect to \( T \), as assumed earlier. \( \square \)

When \( s_3 \) (\( s^* \) in Proposition 1) is determined, it partitions the rectangle into four new regions. One the one hand \( s_3 \) is optimal therefore the region bounded by \( c^* Af(s_3)B \) contains no other non-dominated solutions; and on the other all solutions in \( f(s_3)C(f_1(s_2), f_2(s_1))D \) are dominated by \( f(s_3) \), thus the only regions that may contain other non-dominated solutions are the rectangles \( Af(s_1)Cf(s_3) \) and \( Bf(s_3)Df(s_2) \) – Figure 3.(b) – which are the rectangles associated with the two
pairs of solutions \((s_1, s_2)\) and \((s_2, s_3)\), respectively. This procedure can be applied to each of those regions, and be repeated while there still are adjacent 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 just proposed is included in Algorithm 2.

**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 calculated pairs of solutions

\[
\begin{align*}
s_1 &\leftarrow S_1 \\
s_2 &\leftarrow S_2 \\
L &\leftarrow \{s_1, s_2\} \\
X &\leftarrow \{(s_1, s_2)\}
\end{align*}
\]

While \(X \neq \emptyset\) Do

\[
\begin{align*}
(s_1, s_2) &\leftarrow \text{element in } X; \\
X &\leftarrow X - \{(s_1, s_2)\} \\
w_1 &\leftarrow f_2(s_1) - c_2^*; \\
w_2 &\leftarrow f_1(s_2) - c_1^* \\
s &\leftarrow \text{best solution to (4)}
\end{align*}
\]

If \(s\) is defined Then

\[
\begin{align*}
L &\leftarrow L \cup \{s\} \\
\text{If } f_1(s) &\neq f_1(s_2) \text{ Then } X \leftarrow X \cup \{(s, s_2)\} \\
\text{If } f_2(s) &\neq f_2(s_1) \text{ Then } X \leftarrow X \cup \{(s_1, s)\}
\end{align*}
\]

EndIf

EndWhile

Finally it should be remarked that problem (4) can be formulated as the equivalent single criteria mixed integer linear program

\[
\begin{align*}
\min \quad & v \\
\text{s. t.} \quad & w_1 f_1(s) - v \leq w_1 c_1^* \\
& w_2 f_2(s) - v \leq w_2 c_2^* \\
& s \in S
\end{align*}
\]

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

As a final remark we mention that in practice it can be difficult to compute lexicographic best solutions at the beginning of the algorithm. As a consequence the initial search area can be wider than what is actually necessary and when inserting new solutions it should be verified whether they dominate some of the previous.

The proposed method is suitable for an interactive approach with the decision maker, as it allows him (her) to choose which pairs of solutions to analyze, and thus which regions to further explore seeking for non-dominated solutions.

As an example of the method’s application we consider the following bicriteria knapsack problem
with ten items,

\[
\begin{align*}
\min \quad & 30x_1 + 13x_2 - 89x_3 + 46x_4 + 6x_5 + 29x_6 + 34x_7 + 71x_8 - 23x_9 - 54x_{10} \\
\min \quad & -86x_1 - 53x_2 - 33x_3 - 4x_4 + 24x_5 - 43x_6 + 60x_7 - 72x_8 - 84x_9 - 37x_{10} \\
\text{s. t.} \quad & 12x_1 + 51x_2 + 39x_3 + 12x_4 + 3x_5 + 48x_6 + 96x_7 + 88x_8 + 74x_9 + 81x_{10} \leq 103 \\
\end{align*}
\]

The images of its supported (in red) and unsupported (in green) non-dominated solutions are depicted in Figure 4.

![Figure 4: Efficient solutions with Algorithms 1 and 2 applied to problem (6)](image)

3 Computational experiments

The two phases of the algorithm were implemented in C language and used CPLEX 12.2 to solve the intermediate mixed integer programs. The tests ran on a Dual Core AMD Opteron at 2 GHz, with 4 Gb of RAM, for two sets of randomly generated instances:

- The first set of instances consists of integer programs with two objective functions with \( m = 30, 40, 50, 100 \) constraints and \( n = 2m, 4m \) variables. 20% of the cost coefficients \( c_{ij} \) are generated in \([-100, -1]\), whereas the remaining are obtained uniformly in \([0, 100]\). The matrix coefficients \( a_{ij} \) are integers distributed as 10% in \([-100, -1]\), 10% equal to 0, and 80% in \([1, 100]\). The right-hand side coefficients \( b_i \) are integers calculated in \([100, \sum_{j=1}^{n} a_{ij}]\).

- The second set of instances consists of knapsack problems again with two objective functions, of type

\[
\begin{align*}
\min \quad & f_1(x) = c_1^T x \\
\min \quad & f_2(x) = c_2^T x \\
\text{s. t.} \quad & ax \leq W \\
\end{align*}
\]

\( x \in \{0, 1\}^n \)

with \( c_{ij} \) and \( a_j \) uniformly generated in \([-100, 100]\), and \( W \) a random integer in \([100, \sum_{j=1}^{n} a_j]\).
For each problem dimension thirty instances were generated, and the previous algorithms were applied. The average results in terms of the running times and of the number of computed non-dominated solutions are summarized in Table 1 for the general integer problems, and in Table 2 for the knapsack problems.

Table 1: Results for random integer problems

<table>
<thead>
<tr>
<th>CPU times (in seconds)</th>
<th>$m$</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>100</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td></td>
<td>2$m$</td>
<td>2$m$</td>
<td>2$m$</td>
<td>2$m$</td>
<td>4$m$</td>
<td>4$m$</td>
<td>4$m$</td>
<td>4$m$</td>
</tr>
<tr>
<td>Sorting</td>
<td></td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Phase 1</td>
<td></td>
<td>0.152</td>
<td>0.264</td>
<td>0.278</td>
<td>0.932</td>
<td>0.159</td>
<td>0.268</td>
<td>0.249</td>
<td>0.596</td>
</tr>
<tr>
<td>Phase 2</td>
<td></td>
<td>0.126</td>
<td>0.181</td>
<td>0.212</td>
<td>0.751</td>
<td>0.104</td>
<td>0.188</td>
<td>0.336</td>
<td>0.389</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>0.278</td>
<td>0.445</td>
<td>0.490</td>
<td>1.683</td>
<td>0.263</td>
<td>0.455</td>
<td>0.584</td>
<td>0.985</td>
</tr>
</tbody>
</table>

Number of solutions

<table>
<thead>
<tr>
<th>$n$</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>100</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>2$m$</td>
<td>2$m$</td>
<td>2$m$</td>
<td>2$m$</td>
<td>4$m$</td>
<td>4$m$</td>
<td>4$m$</td>
<td>4$m$</td>
<td></td>
</tr>
<tr>
<td>Phase 1</td>
<td>2.367</td>
<td>2.600</td>
<td>2.167</td>
<td>2.567</td>
<td>2.100</td>
<td>0.233</td>
<td>2.267</td>
<td>2.167</td>
</tr>
<tr>
<td>Phase 2</td>
<td>0.200</td>
<td>0.200</td>
<td>0.100</td>
<td>0.233</td>
<td>0.033</td>
<td>0.100</td>
<td>0.100</td>
<td>0.000</td>
</tr>
<tr>
<td>Total</td>
<td>2.567</td>
<td>2.800</td>
<td>2.267</td>
<td>2.800</td>
<td>2.133</td>
<td>2.333</td>
<td>2.367</td>
<td>2.167</td>
</tr>
</tbody>
</table>

The sorting time on Tables 1 and 2 refers to the time used to sort all the supported solutions found along phase 1, in order to pair adjacent solutions. For the considered instances the CPU time of this task was negligible. Similarly the CPU times of phases 1 and 2 are small, under 1 second, on general integer instances, although the number of non-dominated solutions is also rather small in these cases, specially unsupported solutions.

Table 2: Results for random knapsack problems

<table>
<thead>
<tr>
<th>CPU times (in seconds)</th>
<th>$n$</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>50</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sorting</td>
<td></td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Phase 1</td>
<td></td>
<td>0.064</td>
<td>0.238</td>
<td>0.348</td>
<td>0.544</td>
<td>1.381</td>
</tr>
<tr>
<td>Phase 2</td>
<td></td>
<td>0.106</td>
<td>0.826</td>
<td>2.263</td>
<td>5.773</td>
<td>32.822</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>0.170</td>
<td>1.064</td>
<td>2.611</td>
<td>6.317</td>
<td>34.203</td>
</tr>
</tbody>
</table>

Number of solutions

<table>
<thead>
<tr>
<th>$n$</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>50</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phase 1</td>
<td>5.233</td>
<td>9.633</td>
<td>12.367</td>
<td>19.233</td>
<td>37.533</td>
</tr>
<tr>
<td>Phase 2</td>
<td>5.067</td>
<td>20.633</td>
<td>38.733</td>
<td>87.067</td>
<td>282.467</td>
</tr>
<tr>
<td>Total</td>
<td>10.300</td>
<td>30.267</td>
<td>51.100</td>
<td>106.300</td>
<td>320.000</td>
</tr>
</tbody>
</table>

Knapsack instances show a number of non-dominated solutions greater than the general integer instances. Consequently the CPU times are greater as well. Still, for the biggest problems, comprising 100 items, 320 solutions were computed in an average time of 34.203 seconds, 1.381 used to find 37.533 supported solutions and 32.822 seconds to find 282.467 unsupported solutions.
It should be noticed 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 this methods would be smaller, depending on the regions to be searched.

Acknowledgments This work has been partially supported by Fundação para a Ciência e a Tecnologia (FCT) under projects PEst-C/EEI/UI0308/2011 and PTDC/EEA-TEL/101884/2008.

References


