An ε-constraint based method for finding all the efficient solutions and all non-dominated vectors for bi-criteria network flow problems

Augusto Eusébio[*†] Jos Rui Figueira[†‡]

March 10, 2007

[*]Escola Superior de Tecnologia e Gestão, Instituto Politécnico de Leiria, Morro do Lena - Alto Vieiro, 2411-901, Leiria, Portugal, Phone: (+351) 244 820 300, Fax:(+351) 244 820 310, E-mail: aeusebio@estg.ipleiria.pt

[†]CEG-IST, Center for Management Studies, Instituto Superior Técnico, TagusPark, Av. Cavaco Silva, 2780-990, Porto Salvo, Portugal, Phone: (+351) 21 423 35 07, Fax: (+351) 21 423 35 68, E-mail: figueira@ist.utl.pt

[‡]Associate Researcher at LAMSADE, Universit Paris-Dauphine, Place du Marchal De Lattre de Tassigny, 75 775 Paris Cedex 16, France.

Abstract

This paper presents a method for identifying all the efficient solutions and non-dominated vectors for integer bi-criteria "minimum cost" network flow problems. The method combines a network simplex algorithm, the ε-constraint method and a branch-and-bound algorithm. The set of all non-dominated vectors in the criterion space is determined by solving an ε-constraint problem with branch-and-bound techniques. By exploring the branch-and-bound then all the efficient solutions can be defined. The main advantage of the proposed method concerns the identification of non-integer solutions exploiting only network structures. Computational results are also reported in this paper.

Keywords: *Multicriteria linear and integer programming, Bi-criteria network flows, Network simplex algorithm, Efficient/non-dominated solutions*

Introduction

Studies of the characterization of the efficient set and design of new approaches for multiple criteria combinatorial problems are scarce. There are, of course, many questions which remain open in this field (see [8] for a review). In this paper a method for identifying the non-dominated vectors set and the efficient solutions set for the bi-criteria "minimum cost" network flow problem is presented. The method is based on ε-constraint and branch-and-bound techniques. This method is also valid for more general linear integer bi-criteria problems, but in such cases LP-relaxation must be solved by linear programming algorithms that may not be very efficient. The present paper shows how the LP-relaxation can be easily computed exploiting the particular structure of networks and continues the work [6]. The method has been implemented and tested and the results are shown in Section 5. It is well known that, in general, the number of non-dominated vectors is significantly fewer than the number of efficient solutions (see [4]) and the proposed algorithm can be used to corroborate this fact as it can be seen in Section 4.

The paper starts with section 1 where definitions and notation, required in the remain sections, are introduced. Section 2 contains a short presentation of the network simplex method. The proposed method is presented in Section 3 and it is illustrated in section 4. Some computational results are reported in Section 5.

1 Concepts: Definitions and notation

Let $\mathcal{G} = (\mathcal{S}, \mathcal{A})$ be a *directed* and *connected graph*, where $\mathcal{S}$ is a finite set of *nodes* or *vertices* with cardinality $|\mathcal{S}| = m$, and $\mathcal{A}$ is a collection of ordered pairs of elements of $\mathcal{S}$ called *arcs*, with cardinality $|\mathcal{A}| = n$.

A graph $\mathcal{G}' = (\mathcal{S}', \mathcal{A}')$ is called a *subgraph* of $\mathcal{G} = (\mathcal{S}, \mathcal{A})$ if $\mathcal{S}' \subseteq \mathcal{S}$ and $\mathcal{A}' \subset \mathcal{A}$. It is a spanning subgraph of $\mathcal{G}$ if $\mathcal{S}' = \mathcal{S}$. A *path* $\mathcal{P}$ is a sequence of vertices and arcs, $i_1 - a_1 - i_2 - a_2 - \ldots - i_{s-1} - a_{s-1} - i_s$, without repetition of vertices and where for which $1 \leq k \leq s-1$ either $a_k = (i_k, i_{k+1}) \in \mathcal{A}$, or $a_k = (i_{k+1}, i_k) \in \mathcal{A}$. A *directed path* is a path without backwards arcs. A *cycle* $\mathcal{C}$ is a closed path where the only repeated vertex is the starting and the end point that coincide. A *directed cycle* is a closed directed path. When in a given graph $\mathcal{G}$ there is always a path linking any two different vertices of $\mathcal{G}$, the graph is called *connected*. A *tree* $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ is a subgraph without cycles where $\mathcal{V} \subseteq \mathcal{S}$ and $\mathcal{E} \subset \mathcal{A}$. A tree $\mathcal{T}$ is called a *spanning tree* when it spans the set of vertices $\mathcal{S}$ of $\mathcal{G}$, that is $\mathcal{V} = \mathcal{S}$. A spanning tree is denoted by $\mathcal{T} = (\mathcal{S}, \mathcal{E})$. Consider (k, l) a given arc belonging to the set $\mathcal{A}$ but not in $\mathcal{E}$. Then, there is a unique cycle $\mathcal{C}$ when the arc (k, l) is added to $\mathcal{E}$. The orientation of $\mathcal{C}$ is the same as (k, l). In a cycle $\mathcal{C}$ a partition of its vertices can be made by

separating the arcs having the same orientation as $\mathcal{C}$ from the arcs in the opposite direction. The collection of all possible cycles of this type is called *fundamental cycle basis* (for more details about network optimization, see [2, 1]). All these definitions are essential for a better understanding of the network simplex method, presented in Section 2.

A directed graph with numerical values assigned to its vertices and/or arcs is called *network*. Let $\mathcal{G} = (\mathcal{S}, \mathcal{A})$ be a network with two *"costs"* c^1_{ij} and c^2_{ij}, a *lower bound* l_{ij} and an *upper bound* or *capacity* u_{ij} associated with every arc $(i, j) \in \mathcal{A}$. The numerical values l_{ij} and u_{ij} respectively denote the minimum and the maximum amount that must flow on the arc (i, j). Finally, let x_{ij} be the *amount of flow* on the arc (i, j). A numerical value b_i is also associated with each vertex $i \in \mathcal{S}$ denoting its *supply* (if $b_i > 0$) or its *demand* (if $b_i < 0$). A vertex with $b_i = 0$ is called a *transshipment vertex*. The bi-criteria "minimum cost" network flow problem can be stated as follows:

$$\begin{aligned}
&\min f_1(x) = \sum_{(i,j)\in\mathcal{A}} c^1_{ij} x_{i,j} \\
&\min f_2(x) = \sum_{(i,j)\in\mathcal{A}} c^2_{ij} x_{ij} \\
&\text{subject to :} \\
&\sum_{j \mid (i,j)\in\mathcal{A}} x_{ij} - \sum_{k \mid (k,i)\in\mathcal{A}} x_{ki} = b_i, \quad \forall i \in \mathcal{S} \\
&\\
&l_{ij} \le x_{ij} \le u_{ij}, \quad \forall (i,j) \in \mathcal{A}
\end{aligned} \tag{1}$$

In what follows, the assumptions below must be taken into account: The graph is directed and connected; all the numerical values for the costs, lower and upper bounds on the arcs and supplies/demands on the vertices are integral and finite; the condition $\sum_{i\in\mathcal{S}} b_i = 0$ must be fulfilled; the integer bi-criteria "minimum cost" network flow problem has at least two feasible solutions and the minimum values for the individual objective functions are different.

Problem (1) can be presented in a more dense form as follows:

$$\begin{array}{lll}
\text{"min"} & F(x) & = (f_1(x), f_2(x)) \\
\text{subject} & \text{to :} & \\
 & x \in X & \leftarrow \{x \in \mathbb{R}^m | Ax = b, l \le x \le u\},
\end{array} \tag{2}$$

where, $x = (x_{i_1 j_1}, \ldots, x_{i_n j_n}) \in \mathbb{R}^n$, is the vector of decision variables; X is the set of feasible solutions of (1); A is the $m \times n$ *node-arc incidence matrix*; l is the vector of lower bounds; u is the vector of upper bounds; b is the vector of supplies/demands

on vertices; $F(x) = \big(f_1(x), f_2(x)\big)^T$ is the vector of objectives to be "minimized"; $Y = F(X)$ is the set of all feasible vectors y in R^2, where $y = (y_1, y_2)^T$ with $y_q = f_q(x)$ for $q = 1, 2$.

Some concepts of multi-criteria programming must also be reviewed for a better understanding of the next sections (see [13, 12] and [5]). *Dominance* is a key concept in multiple criteria decision analysis. Let us define this concept for the general multiple criteria case with r criteria.

Definition 1.1 (Dominance) *Consider y' and y'' two criterion vectors. Then, y' dominates y'' iff $y' \leq y''$ and $y' \neq y''$, that is, $y'_q \leq y''_q$ for all $q = 1, \ldots, r$ with at least one strict inequality.*

Definition 1.2 (Non-dominated vector) *A vector $y' \in Y$ is called non-dominated (ND) iff there does not exist another vector $y \in Y$ such that $y \leq y'$ and $y \neq y'$. Otherwise, y' is a dominated criterion vector.*

A distinction between efficient solutions in decision variable space and non-dominated vectors in criteria space can be made. Efficient solutions are crucial for the usefulness of multiple criteria methods. This concept was first introduced by [10]. Thus, these solutions are called *Pareto optimal*, and also *non-inferior* or *functional efficient* solutions.

Definition 1.3 (Efficient solution) *A solution $x' \in X$ is said to be efficient iff it is impossible to find another solution $x \in X$ with a better evaluation of a given criterion without deteriorating the evaluations of at least another criterion.*

In multiple criteria linear integer programming, two types of non-dominated vectors can be distinguished: supported and unsupported non-dominated vectors.

Let

$$Y^{\geqq} = Conv(ND(Y) + \mathbb{R}^p_{\geqq})$$

where, $\mathbb{R}^p_{\geqq} = \{y \in \mathbb{R}^p | y \geqq 0\}$ and $ND(Y) + \mathbb{R}^p_{\geqq} = \{y \in \mathbb{R}^p : y = y' + y'', y' \in ND(Y) \text{ and } y'' \in \mathbb{R}^p_{\geqq}\}$, $y \geqq 0$ if $y_q \geq 0$, $q = 1, 2 \cdots, p$ and $Conv$ stands for convex hull.

Definition 1.4 (Supported ND) *Let y denote a non-dominated criterion vector. Then, if y is on the boundary of $Y^{\geqq}$, y is a supported non-dominated criterion vector. Otherwise, y is an unsupported non-dominated criterion vector.*

Definition 1.5 (Supported-extreme ND) *Let y be a supported non-dominated criterion vector. Then, y is a supported-extreme vector if it is an extreme point of $Y^{\geqq}$. Otherwise, y is a supported non-extreme vector.*

Inverse images of *supported* non-dominated criterion vectors are said to be supported efficient points and inverse images of unsupported non-dominated criterion vectors are said to be *unsupported efficient points*.

Let: $ND(Y)$ be the set of all the non-dominated vectors of Y; $NDS(Y)$ be the set of all the supported non-dominated vectors of Y; $NDU(Y)$ be the set of all the unsupported non-dominated vectors of Y, that is, $NDU(Y) = ND(Y) \setminus NDS(Y)$; $NDE(Y)$ be the set of all supported-extreme non-dominated vectors; $EF(X)$ be the set of all the efficient solutions of X; $EFS(X)$ be the set of all supported efficient solutions; $EFU(X)$ be the set of all unsupported efficient solutions, that is, $EFU(X) = EF(X) \setminus EFS(X)$.

In multiple criteria linear programming several techniques (scalar optimization problems) can be used in order to characterize efficient solutions (non-dominated vectors) like, for example, weighted-sum approaches, Tchebycheff metrics based methods, ε-constraint methods, and so on (see [12]). Among the existing methods, the ε-constraint approach can be easily used in multiple criteria integer problems without any additional restrictions. Efficient solutions can be characterized as optimal solutions for the ε-constraint problem.

The ε-constraint problem associated with bi-criteria (1) can be stated as follows,

$$\begin{array}{ll} \min & f_1(x) \\ \text{subject to :} & \\ & x \in X \\ & f_2(x) \leq \varepsilon, \end{array} \tag{3}$$

where, ε is a scalar. It varies among all the values for which (3) remains feasible. In order to identify a set of efficient solutions, a sequence of problems (3) is solved for each different value of ε ([3]). For integer bi-criteria linear programming problems, the entire non-dominated set $ND(Y)$ can be easily determined by solving a sequence of problems (3).

Theorem 1.1 ([7]) *Consider $\varepsilon \geq \min f_2(x)$. If the solution x^* solves problem (3) and when x^* is not unique it leads to a minimal value for criterion $f_2(x)$, then x^* solves (1), that is, x^* is an efficient solution for (1).*

Proof.

Suppose now that x^* does not solve the problem. Another solution $\hat{x}$ can then be considered so that only one of the following two cases can occur:

- $f_1(\hat{x}) < f_1(x^*)$ and $f_2(\hat{x}) \leq f_2(x^*)$ which contradicts the fact that x^* solves (3), or
- $f_2(\hat{x}) < f_2(x^*)$ and $f_1(\hat{x}) \leq f_1(x^*)$, which contradicts the hypothesis that x^* is optimal for (3) with the smallest value for $f_2(x)$.

The theorem is proved by the two cases above.

Problem (3) will be used in the algorithm outlined in Section 3 to determine all the non-dominated vectors and all efficient solutions for problem (1).

2 Network simplex method: A remind

Let us now succinctly recall the network simplex method on minimum cost network flow problems (see Figure 1). The basic idea for any variant of the network simplex method is a Spanning Tree Structure (STS), $(\mathcal{T}, L, U)$. Such a structure (or solution) is obtained when, for any arc not belonging to this tree, the flow value is fixed at its lower bound level or at its upper bound level. All the arcs fixed at their lower bound level belong to the set L, while all the arcs fixed at their upper bound level belong to the set U. The remaining arcs are those belonging to the spanning tree $\mathcal{T}$. A minimum cost network flow problem has always at least one STS optimal solution (see [1]). It is possible to find an optimal STS by shifting from one STS to another, successively. At each iteration, we exchange a pair of arcs (one arc entering STS and one arc coming out of STS). Any STS corresponds to one feasible basic solution in linear programming, and each shift from one STS to another coincides with one pivoting operation in the standard simplex method. The initialization of the algorithm consists of finding one feasible STS (or equivalently, a feasible basic solution in the standard simplex method). Two vectors are associated with this STS, the flow x (primal solution) and the potential π (dual solution). Each iteration of the method consists of: (1) identifying one eligible arc (k, l) with $(k, l) \notin \mathcal{T}$; (2) adding the arc (k, l) to $\mathcal{T}$ and finding an arc (p, q) coming out of $\mathcal{T}$; and, updating STS and the primal and dual solution (x, π).

An arc, (i, j), not belonging to $\mathcal{T}$ is said to be eligible if:

i) Its reduced cost, $\bar{c}_{ij}$, is strictly negative and its flow is at its lower bound, that is, $\bar{c}_{ij} < 0$ and $(i,j) \in L$.

ii) Its reduced cost is strictly positive and its flow is at its upper bound, that is, $\bar{c}_{ij} > 0$ and $(i,j) \in U$.

The reduced cost of a given arc (i,j) is defined as follows:

$$\bar{c}_{ij} = c_{ij} - \pi_i + \pi_j,$$

where, π_i and π_j are the dual variables associated with the vertices i and j, respectively. It should be noted that for all the arcs $(i,j) \in \mathcal{T}$ the reduced cost $\bar{c}_{ij} = 0$.

Simplex method.
{ Computing a minimum cost flow. }
(1) **begin**
(2) let $(\mathcal{T}, \mathcal{L}, \mathcal{U})$ be a starting feasible STS;
(3) let x be the flow and π the dual variable associated with $(\mathcal{T}, \mathcal{L}, \mathcal{U})$;
(4) **while** (not optimal solution) **do**
(5) **begin**
(6) select an entering arc (k,l) not in $\mathcal{T}$;
(7) add (k,l) to $\mathcal{T}$ and remove (p,q) from $\mathcal{T}$;
(8) update the STS and the solutions x and π;
(9) **end**
(10) **end**

Figure 1: *Network simplex algorithm.*

At each iteration, the network simplex method shown in Figure 1 always gives an integer solution for the minimum cost network flow problem. But it is possible to obtain non-integer solutions between two adjacent STSs. Let us recall that when moving from one STS to an adjacent one, an amount of flow, $\triangle$, must be sent along the orientation of cycle $\mathcal{C}$. This quantity $\triangle$ is integer. But, what happens if a non-integer amount of flow is sent along $\mathcal{C}$? It is obvious that a non-integer solution will be obtained. This solution has exactly $|\mathcal{C}|$ non-integer variables, but it does not define a spanning tree structure. This idea is very important if we wish to obtain non-integer solutions for the LP-relaxation of problem (3).

3 Outline of the method

This section outlines an approach for the search of all the non-dominated vectors, $ND(Y)$ and all efficient solutions,$EF(X)$. The method solves a sequence of problems (3) and uses the branch-and-bound algorithm to determine its integer optimal solutions. Non-dominated vectors are determined by decreasing order of the values for the second objective function. The potential zones for the search of non-dominated vectors are identified by a set of triangles built from the supported non-dominated vectors associated with adjacent STSs. This procedure can be described as follows:

1. Identify two adjacent STSs and the associated non-dominated vectors, $y' = (y'_1, y'_2)$ and $y'' = (y''_1, y''_2)$, suppose that $y'_2 \geq y''_2$.

2. If $y' \neq y''$ the triangle with vertices y', y'' and (y''_1, y'_2) is the region of potential non-dominated vectors since by definition all non-dominated vector $y = (y_1, y_2)$ such $y''_2 \leq y_2 \leq y'_2$ is in this region.

3. For each triangle, search all the non-dominated vectors using a sequence of problems (3).

   a) Solve the problem (3) using $\varepsilon = y'_2 - 0.5$. All non-dominated vectors have integer coordinates in the two-dimensional coordinate system with axes $y_1 = f_1(x)$ and $y_2 = f_2(x)$, therefore there are not non-dominated vectors $y = (y_1, y_2)$ such $y'_2 - 1 < y_2 < y'_2$ and all non-dominated vectors in this triangle have second coordinate less than $y'_2 - 0.5$.

   b) If the non-dominated vector found in 3a is y'' then identify a new pair of adjacent STSs and the associated non-dominated vectors and repeat 2, case there is some, or stop if not. Otherwise let $y''' = (y'''_1, y'''_2)$ be the non-dominated vector found in 3a. Solve the problem (3) using $\varepsilon = y'''_2 - 0.5$ and repeat 3b.

This method is actively dependent of the computation of the STSs. The network simplex method turned out to be fast and strongly enough to avoid cycling and stalling when built observing some rules such as working with *Strongly Feasible Basis* or using the *Least Recently Considered* entering rule (see [2]).

The main advantage of the algorithm is related to the way in which non-integer solutions are determined, exploiting only network structures and thus avoiding the need to solve these problems with LP-codes.

The example presented in Section 4 shows, step by step, how the method works.

4 An illustrative example

This section illustrates the way how the proposed algorithm works. Consider the bi-criteria "minimum cost" network flow problem in Figure 2. This example has

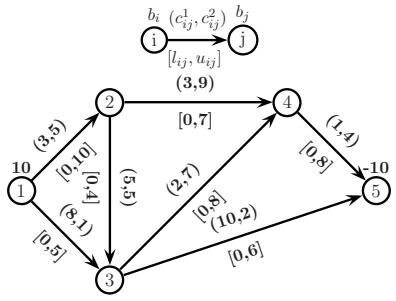


Figure 2: A bi-criteria example.

93 feasible solutions that are presented in Table 3 (Appendix A). Among the 93 solutions only 10 are efficient. These 10 efficient solutions correspond one to one to 10 non-dominated vectors . Figure 4 presents all the non-dominated vectors.

First, we shall show how the set $NDE(Y)$ can be determined by parametric programming. In the example, $NDE(Y) = \{y^{48}, y^4, y^1, y^5\}$ as can be seen in Figure 4. In order to obtain all the vectors, we first identify y^{48}, then y^4 and so on. Let us show how to determine the first two vectors of $NDE(Y)$, y^{48} and y^4:

- First, $f_1(x)$ is minimized. Its optimal value can be obtained at two different points y^{48} and y^{90}, where $f_1^* = 96$, but only y^{48} gives the minimal value for $f_2(x)$, $\hat{f}_2 = 144$. Vector y^{90} is thus discarded and y^{48} is saved. The STS corresponding to the vector y^{48} is presented in Figure 3. The arcs represented by the lines in bold are those belonging to the spanning tree, $\mathcal{T}$.

- Second, from y^{48} an adjacent STS leading to the next extreme non-dominated point of $Conv(Y)$ must be identified. STSs corresponding to the vectors y^4, y^{47} and y^{90} can be reached from y^{48} by identifying the fundamental cycle basis, but only vector y^4 is interesting. How can this vector be obtained? As we can see in Figure 4, the slope of the line connecting y^{48} and y^4 is the lowest. In order to identify this slope we may use the information given by the reduced costs on the arcs not belonging to $\mathcal{T}$. Let us recall that the slope of the line connecting y^{48} to y^{47} is $m_1 = \frac{132-144}{104-96} = \frac{-12}{8} = -1.5$, while the slope of the line connecting y^{48} to y^4 is $m_2 = \frac{135-144}{103-96} = \frac{-9}{7} = -1.286$. Both slopes can be obtained by the ratios $r(2,4) = 6/-4 = -1.5$ and $r(4,5) = 9/-7 = -1.286$,

respectively. The lowest is the ratio $r(2,4)$. So, the arc $(2,4)$ forms a cycle allowing to move from y^{48} to y^4.

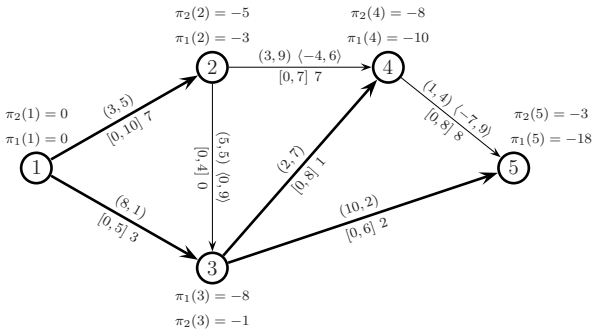


Figure 3: Primal and dual solutions for y^{48}.

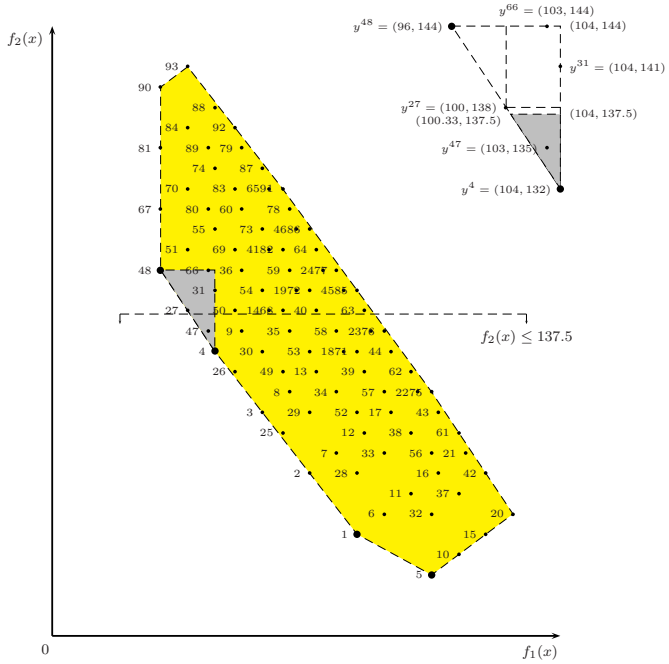


Figure 4: Points in criteria space.

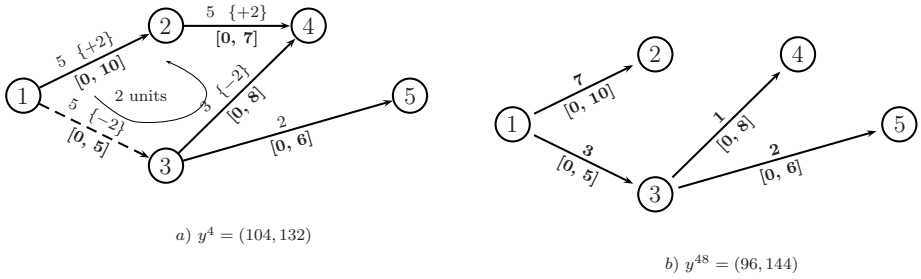


Figure 5: Adjacent spanning tree solutions 4 and 48.

Finally, to complete the illustration of the methods we need to show how the optimal integer solution for (3) can be determined by using the branch-and-bound technique and the network simplex algorithm of Figure 1.

This approach can be used to find all non-dominated solutions for problem (1) as well as a set of non-dominated solutions in a zone of interest. In practice it is frequent for decision makers to define certain zones of interest for a local search. The proposed method is appropriate to situations of this kind.

Let us suppose that we need to determine all the non-dominated vectors inside the triangle formed by the points $(96, 144)$, $(104, 144)$ and $(104, 132)$. Figure 4 presents this triangle. The same triangle is represented in detail on the upper right corner of Figure 4. Imagine we have already computed vectors y^{48} and y^4 and the associated STSs. Next step is to identify vector y^{47} which is the optimal integer solution of (3), where $\epsilon = 137.5$. Before reaching vector y^{47}, several steps were executed:

1. First, consider the problem (3) with $\varepsilon = 137.5$. In our example this problem is denoted by A, and our list W is updated so that $W = \{A\}$. The feasible region is given by the dark area in the triangle placed on the upper right of Figure 4.

2. Second, the optimal non-integer solution of A must be determined while A is removed from W. In order to compute the optimal value of A, a simple technique can be used. We only need to identify the two nearest extreme vectors of the non-integer solution for A. These two extreme vectors correspond to the STSs 48 and 4 (see Figure 4). This means that the non-integer solution for A is between STSs 48 and 4. STS 48 is on the left of the optimal non-integer solution while STS 4 is on the right. The cycle allowing to move from 4 to 48 is $\mathcal{C} = \{(1, 2), (2, 4), (3, 4), (1, 3)\}$, where the arc (k, l) is the arc $(1, 3)$. Figure

5 presents these two adjacent STSs. The amount sent along $\mathcal{C}$ is $\triangle = 2$. This means that when we send 2 units from STS 4 along $\mathcal{C}$, STS 48 is reached and the cost for the second criterion increases in the quantity $144 - 132 = 12$, that is, an increase of 6 for each unit sent along $\mathcal{C}$. So, if we want an increase of $137.5 - 132 = 5.5$ in the second criterion, we must send $\triangle = 5.5/6 = 0.9167$ along $\mathcal{C}$. In this case we obtain a non-integer solution with exactly $|\mathcal{C}| = 4$ non-integer variables (see Table 5). The branch-and-bound tree is given in Figure 7.

3. Third, we proceed to a partition of A into two subproblems, B and C. The branching variable is $x(1,3) = 4.08333$. Subproblem B is defined by introducing constraint $x(1,3) = 5$, while subproblem C is defined with the help of constraint $0 \leq x(1,3) \leq 4$. Defining these two subproblems in this way, we can guarantee that STSs 48 and 4 are always feasible for B and C, respectively.

4. Fourth, we need to determine the bounds for both subproblems, B and C. Subproblem B has an integer solution. Now, B is the incumbent problem with cost equal to 104 (see Appendix B and Table 5).

5. Fifth, let us study now subproblem C. Appendix C contains the feasible region for C in criteria space. Let us recall that STS 48 remains feasible for subproblem C. So, we can start by using this solution and then move to 27 which is now a STS, but it is still on the left (or above the line for $f_2(x) = 137.5$) of the non-integer optimal solution for C. Therefore, we need to continue in order to obtain a STS on the right of the non-integer optimal solution for C. When moving to the adjacent solution on the boundary of $Conv(Y)$, STS 25 is attained (see Figures 11 and 12 on Appendix C). Now, we proceed as in 2 and C is added to the list W for further analysis (see also Figure 6 and Table 5).

6. We proceed in the same manner until the optimal solution is obtained. The tree shown in Figure 6 gives us all the iterations needed to solve (3) where $\varepsilon = 137.5$. The corresponding solutions are presented in Table 5.

In the general case, after identifying the several triangles, it is not known where the non-dominated vectors are, inside each triangle, and we must go through all the area looking for non-dominated vectors. In our example the triangle has vertices $y^{48} = (96, 144)$, $(104, 144)$ and $y^4 = (104, 132)$ with the upper vertex with $y_2^{48} = 144$ and the lower with $y_2^4 = 132$. Between the straight lines $y_2 = 143$ and $y_2 = 144$ we know there is not any non-dominated vector since y_2 has to be integer. Thus if the problem (3) is solved with $\varepsilon = 143.5$ the non-dominated vector on the left below the

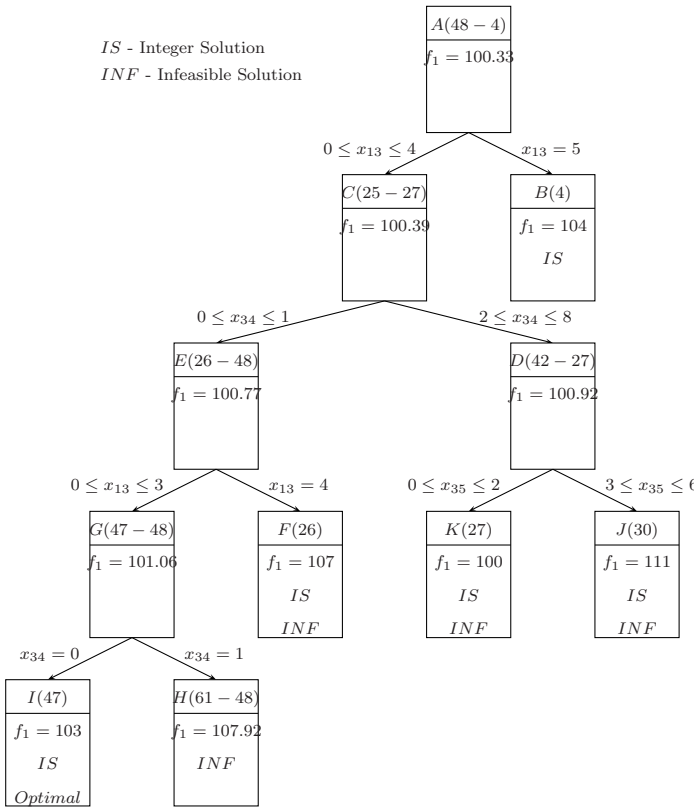


Figure 6: Branch-and-bound iterations.

straight line $y_2 = 143.5$ will be found. This non-dominated vector is $y^{27} = (100, 138)$ (see Figure 7). The next non-dominated vector is in triangle below straight line $y_2 = 138$. Consider $\varepsilon = 137.5$ and solve the problem (3). The non-dominated vector $y^{47} = (103, 135)$ is obtained. Finally, the problem (3) with $\varepsilon = 134.5$ give rise to the non-dominated vector $y^4 = (104, 132)$ the lowest vertex of the triangle. Thus our exploration of this triangle ends and the algorithm moves for the next triangle.

The algorithm computes also all the efficient solutions. As example the algorithm was run for problem in Figure 8. The set of 14 non-dominated vectors: $\{(290, 356), (292, 350), (293, 331), (295, 325), (296, 306), (298, 300), (299, 281), (301, 275), (302, 256), (304, 250), (316, 244), (328, 238), (340, 232), (352, 226)\}$ (see Figure 9) corresponds to a set of 74 efficient solutions (see Table 6). Getting all efficient solutions takes a larger CPU time. To understand how the algorithm works consider the problem (3) with $\varepsilon = 280.5$. The branch-and-bound tree is given in Fig-

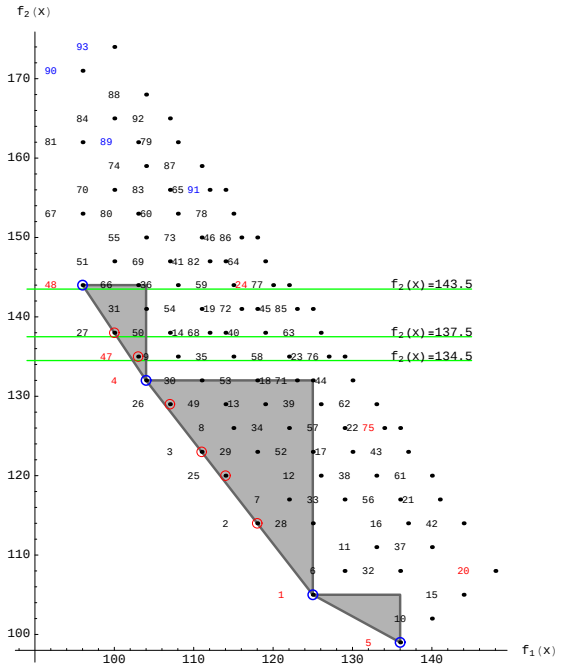


Figure 7: Non-dominated vectors.

ure 13 (see Appendix E). There are 3 efficient solutions the ones in nodes 10, 18 and 28. If we intend to compute only the set of non-dominated vectors then the parts of the tree below nodes 14 and 22 could be cut because in these nodes, $f_1 = 300.34$, and the adjacent STSs used to compute this value have $f_2 \geq 275$, which is the f_2 value for the incumbent solution. Then any solution in these branches would have $f_1 \geq 301$ and $f_2 \geq 275$. Therefore it was avoided to continue exploring these branches searching for a better solution.

5 Computational Experiments

The computational experiments were designed on the basis of a set of 30 instances for each problem type. Each instance was generated by using the *NETGEN* network generator (see [9]) after some changes for this particular problem, the bi-criteria minimum cost flow problem. Each problem type has all arcs capacitated with minimum value 0 and maximum value less or equal to 50. Tables 1 and 2 present the problem type number (N), average number of nodes (m) and arcs (n) for the 30 instances; the minimum number (Min), the average number (Av), and the maximum number

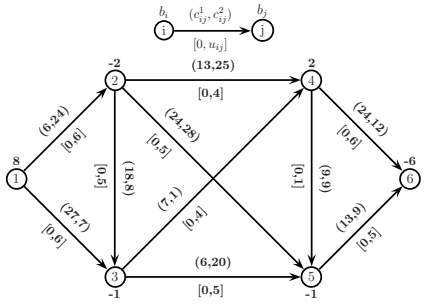


Figure 8: Bi-criteria "minimum cost" flow problem.

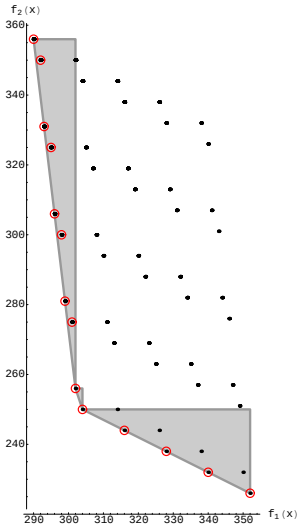


Figure 9: Non-dominated vectors.

(Max) of non-dominated vectors in each set; the minimum number, the average number and the maximum number of efficient solutions in each set; the average difference between the average number of efficient solutions and the average number of non-dominated vectors ($Dif\ Av$); the minimum, average and maximum CPU time, in seconds, to find all non-dominated vectors and to all the efficient solutions for each problem set; and the average difference between the average CPU time to find the efficient solutions and the non-dominated vectors of each set.

The following comments on the results should be pointed out:

1. The CPU time grows with the increasing of the instances size.

Table 1: Computational results.

			ND solutions			EF solutions			Dif
N	m	n	Min	Av	Max	Min	Av	Max	Av
1	10	25	4	35.50	104	4	37.50	112	2.00
2	10	40	25	85.07	215	27	155.60	1683	70.53
3	20	50	7	44.27	71	7	57.10	170	12.83
4	10	25	37	94.33	174	37	145.17	317	50.83
5	30	60	2	35.27	111	2	42.57	153	7.30
6	30	100	27	80.90	196	35	116.53	364	35.63
7	30	150	72	107.80	184	77	203.00	458	95.20
8	30	200	77	126.97	201	111	249.10	509	122.13
9	40	80	9	32.90	61	9	37.93	81	5.03
10	40	150	30	80.77	146	31	125.20	344	44.43
11	40	200	52	110.13	176	59	189.83	389	79.70
12	40	250	54	123.97	176	66	233.40	460	109.43
13	50	100	6	31.93	58	6	35.50	76	3.57
14	50	200	44	89.23	123	58	125.77	282	36.54

Table 2: Computational results (continuation).

	CPU Time (sec.)						
	ND Solutions			Ef Solutions			Dif
N	Min	Av	Max	Min	Av	Max	Av
1	0.00	0.24	2.45	0.00	0.28	2.50	0.04
2	0.25	3.09	11.11	0.49	4.58	23.75	1.48
3	0.02	1.23	3.14	0.02	1.70	5.19	0.47
4	1.14	35.05	185.28	1.88	52.63	248.58	17.58
5	0.00	1.39	8.58	0.00	1.71	11.55	0.32
6	1.05	22.49	94.70	2.09	32.36	126.06	9.87
7	32.38	150.10	531.80	64.67	247.64	828.92	97.53
8	73.42	452.14	1095.16	127.73	759.50	1801.88	307.35
9	0.09	1.91	5.84	0.17	2.47	7.83	0.57
10	12.39	83.63	236.17	15.39	124.96	405.52	41.32
11	60.33	356.29	765.31	101.48	572.29	1192.19	216.01
12	200.36	961.84	2119.11	267.61	1695.70	3486.17	733.87
13	0.09	3.69	15.03	0.13	4.93	20.64	1.24
14	48.95	229.22	658.89	69.86	363.48	821.34	134.26

2. Dense instances contains more non-dominated and efficient solutions and take more time for resolution.

3. There is a significant difference between the number of non-dominated vectors and efficient solutions, which obviously leads to higher CPU time for computing efficient solutions. The biggest difference between the number of efficient solutions, and the number of non-dominated solutions occurs for an instance with 10 nodes and 42 arcs where the former is 1683 and the last 215.

4. We imposed a CPU time of 1 hour. Only one instance of problem type number 12 approached this time. For this maximum CPU time the available memory requirements were enough.

Conclusions

Multiple criteria "minimum cost" network flow problems are known to be hard to solve. [11] proves, for a particular instance with only two criteria, that the number of extreme non-dominated vectors grows exponentially with the number of vertices of the network. However, the proposed method appears to be able to find both non-dominated vectors and efficient solutions for small and medium size instances in a small amount of time. The method is also of great interest, since the initial searching region of non-dominated vectors can be broken into small regions, exploring only the desirable regions, enable its use with intelligent interactive methods helping the choice of the best solution at any time, according with who has to choose.

References

[1] R. Ahuja, T.L. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, New Jersey, 1993.

[2] M.S. Bazaraa, J.J. Jarvis, and H.D. Sherali. *Linear Programming and Network Flows*. John Wiley & Sons, 1977.

[3] V. Chankong and Y.Y. Haimes. *Multiobjective Decision Maker: Theory and Methodology*. North-Holland, New York, 1983.

[4] J.P. Dauer. Analysis of the objective space in multiple objective linear programming. *Journal of Optimization Theory and Applications*, 126:579–593, 1987.

[5] M. Ehrgott. *Multicriteria Optimization*. Springer-Verlag, Berlin, 2nd edition, 2005.

[6] J. Figueira. On the integer bi-criteria network flow problem: A branch-and-bound approach. *Cahier du LAMSADE*, 191, 2002.

[7] Y.Y. Haimes, L.S. Lasdon, and D. A. Wismer. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, pages 296–297, 1971.

[8] H.W. Hamacher, C.R. Pedersen, and S. Ruzika. Multiple objective minimum cost flow problems: A review. *European Journal of Operational Research*, 176:1404–1422, 2007.

[9] D. Klingman, A. Napier, and J. Stutz. Netgen: A program for generating large scale capacitated assignment, transportation, and minimum cost flow network problems. *Management Science*, 20(5):814–821, 1974.

[10] V. Pareto. *Cours d'Economie Politique*. Rouge, Lausanne, 1896.

[11] G. Ruhe. Complexity results for multicriterial and parametric network flows using a pathological graph of zadeh. *Zeitschrift fr Operations Research*, 32:9–27, 1988.

[12] R.E. Steuer. *Multiple Criteria Optimization: Theory, Computation, and Application*. John Wiley & Sons, New York, 1986.

[13] M. Zeleny. *Linear Multiobjective Programming*, volume 95. Springer-Verlag, New York, 1972. Lecture Notes in Economics and Mathematical Systems.

Appendices

A Set of all Feasible Solutions

This appendix contains all the solutions concerning the example presented in Section 6. Efficient solutions (non-dominated vectors) are in bold.

Table 3: Solutions of the bi-criterion network flow on Figure 2.

l_{ij}	0	0	0	0	0	0	0		
u_{ij}	10	5	4	7	8	6	8		
Sol.	(1,2)	(1,3)	(2,3)	(2,4)	(3,4)	(3,5)	(4,5)	$f_1(x)$	$f_2(x)$
1	**5**	**5**	**0**	**5**	**0**	**5**	**5**	**125**	**105**
2	**5**	**5**	**0**	**5**	**1**	**4**	**6**	**118**	**114**
3	**5**	**5**	**0**	**5**	**2**	**3**	**7**	**111**	**123**
4	**5**	**5**	**0**	**5**	**3**	**2**	**8**	**104**	**132**
5	**5**	**5**	**1**	**4**	**0**	**6**	**4**	**136**	**99**
6	5	5	1	4	1	5	5	129	108
7	5	5	1	4	2	4	6	122	117
8	5	5	1	4	3	3	7	115	126
9	5	5	1	4	4	2	8	108	135
10	5	5	2	3	1	6	4	140	102
11	5	5	2	3	2	5	5	133	111
12	5	5	2	3	3	4	6	126	120
13	5	5	2	3	4	3	7	119	129
14	5	5	2	3	5	2	8	112	138
15	5	5	3	2	2	6	4	144	105
16	5	5	3	2	3	5	5	137	114
17	5	5	3	2	4	4	6	130	123
18	5	5	3	2	5	3	7	123	132
19	5	5	3	2	6	2	8	116	141
20	5	5	4	1	3	6	4	148	108
21	5	5	4	1	4	5	5	141	117
22	5	5	4	1	5	4	6	134	126
23	5	5	4	1	6	3	7	127	135
24	5	5	4	1	7	2	8	120	144
25	**6**	**4**	**0**	**6**	**0**	**4**	**6**	**114**	**120**
26	**6**	**4**	**0**	**6**	**1**	**3**	**7**	**107**	**129**
27	**6**	**4**	**0**	**6**	**2**	**2**	**8**	**100**	**138**
28	6	4	1	5	0	5	5	125	114
29	6	4	1	5	1	4	6	118	123
30	6	4	1	5	2	3	7	111	132
31	6	4	1	5	3	2	8	104	141
32	6	4	2	4	0	6	4	136	108
33	6	4	2	4	1	5	5	129	117
34	6	4	2	4	2	4	6	122	126
35	6	4	2	4	3	3	7	115	135
36	6	4	2	4	4	2	8	108	144
37	6	4	3	3	1	6	4	140	111
38	6	4	3	3	2	5	5	133	120
39	6	4	3	3	3	4	6	126	129

continued on the next page

l_{ij}	0	0	0	0	0	0	0		
u_{ij}	10	5	4	7	8	6	8		
Sol.	(1,2)	(1,3)	(2,3)	(2,4)	(3,4)	(3,5)	(4,5)	$f_1(x)$	$f_2(x)$
40	6	4	3	3	4	3	7	119	138
41	6	4	3	3	5	2	8	112	147
42	6	4	4	2	2	6	4	144	114
43	6	4	4	2	3	5	5	137	123
44	6	4	4	2	4	4	6	130	132
45	6	4	4	2	5	3	7	123	141
46	6	4	4	2	6	2	8	116	150
47	**7**	**3**	**0**	**7**	**0**	**3**	**7**	**103**	**135**
48	**7**	**3**	**0**	**7**	**1**	**2**	**8**	**96**	**144**
49	7	3	1	6	0	4	6	114	129
50	7	3	1	6	1	3	7	107	138
51	7	3	1	6	2	2	8	100	147
52	7	3	2	5	0	5	5	125	123
53	7	3	2	5	1	4	6	118	132
54	7	3	2	5	2	3	7	111	141
55	7	3	2	5	3	2	8	104	150
56	7	3	3	4	0	6	4	136	117
57	7	3	3	4	1	5	5	129	126
58	7	3	3	4	2	4	6	122	135
59	7	3	3	4	3	3	7	115	144
60	7	3	3	4	4	2	8	108	153
61	7	3	4	3	1	6	4	140	120
62	7	3	4	3	2	5	5	133	129
63	7	3	4	3	3	4	6	126	138
64	7	3	4	3	4	3	7	119	147
65	7	3	4	3	5	2	8	112	156
66	8	2	1	7	0	3	7	103	144
67	8	2	1	7	1	2	8	96	153
68	8	2	2	6	0	4	6	114	138
69	8	2	2	6	1	3	7	107	147
70	8	2	2	6	2	2	8	100	156
71	8	2	3	5	0	5	5	125	132
72	8	2	3	5	1	4	6	118	141
73	8	2	3	5	2	3	7	111	150
74	8	2	3	5	3	2	8	104	159
75	8	2	4	4	0	6	4	136	126
76	8	2	4	4	1	5	5	129	135
77	8	2	4	4	2	4	6	122	144
78	8	2	4	4	3	3	7	115	153
79	8	2	4	4	4	2	8	108	162
80	9	1	2	7	0	3	7	103	153
81	9	1	2	7	1	2	8	96	162
82	9	1	3	6	0	4	6	114	147
83	9	1	3	6	1	3	7	107	156
84	9	1	3	6	2	2	8	100	165
85	9	1	4	5	0	5	5	125	141
86	9	1	4	5	1	4	6	118	150
87	9	1	4	5	2	3	7	111	159
88	9	1	4	5	3	2	8	104	168
89	10	0	3	7	0	3	7	103	162
90	10	0	3	7	1	2	8	96	171
91	10	0	4	6	0	4	6	114	156
92	10	0	4	6	1	3	7	107	165
93	10	0	4	6	2	2	8	100	174

Source : *Figueira 2002*

$c_1(i,j)$	3	8	5	3	2	10	1		
$c_2(i,j)$	5	1	5	9	7	2	4		
Solution	$x(1,2)$	$x(1,3)$	$x(2,3)$	$x(2,4)$	$x(3,4)$	$x(3,5)$	$x(4,5)$	$f_1(x)$	$f_2(x)$
A (4-48)	5.91667	4.08333	0	5.91667	1.08333	2	8	100.33331	137.5
B (4)	5	5	0	5	3	2	8	104	132
C (25-27)	6	4	0	6	1.94444	2.05556	7.94444	100.38892	137.5
D (42-27)	6	4	0.08333	5.91667	2	2.08333	7.91667	100.91663	137.5
E (26-48)	6.56667	3.43333	0	6.56667	1	2.43333	7.56667	100.76663	137.5
F (26)	6	4	0	6	1	3	7	107	129
G (47-48)	7	3	0	7	0.27778	2.72222	7.27778	101.05554	137.5
H (61-48)	7	3	1.08333	5.91667	1	3.08333	6.91667	107.91663	137.5
I (47)	7	3	0	7	0	3	7	103	135
J (30)	6	4	1	5	2	3	7	111	132
K (27)	6	4	0	6	2	2	8	100	138

Table 5: Solutions A to K.

B Criteria Space for Subproblem B

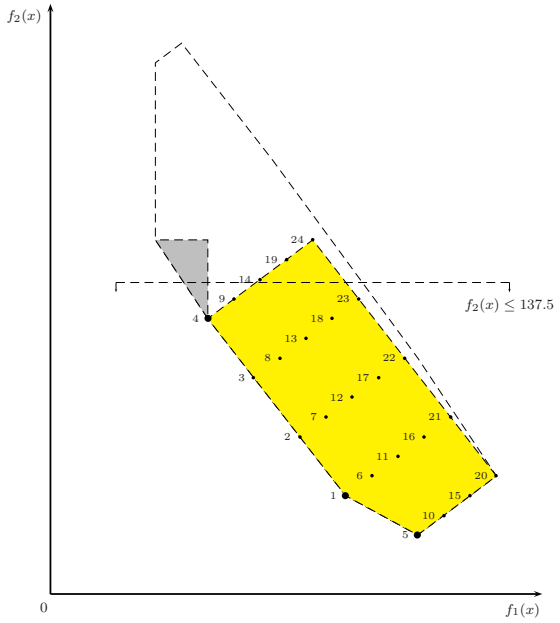


Figure 10: Points in criteria space concerning solution B.

C Criteria Space for Subproblem C

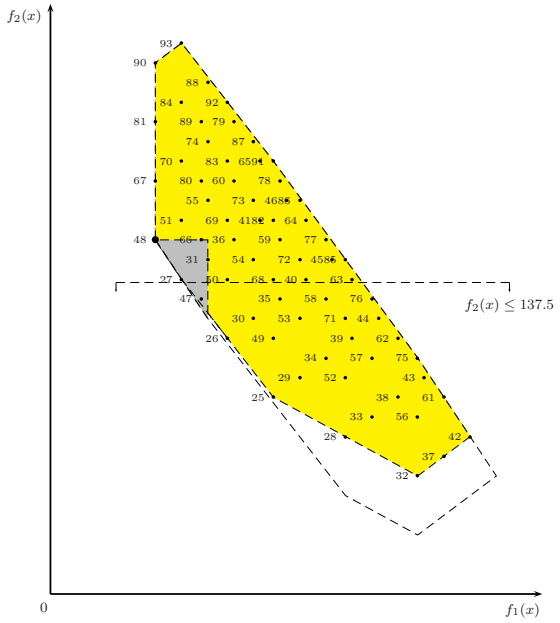
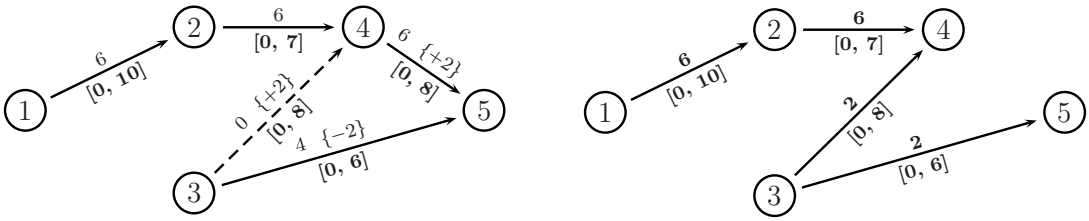


Figure 11: Points in criteria space concerning solution C.



a) $y^{25} = (114, 120)$ *b)* $y^{27} = (100, 138)$

Figure 12: Adjacent spanning tree solutions 25 and 27.

D Efficient Solutions

Table 6: Efficient solutions of problem in Figure 8.

u_{ij}	6	6	5	4	5	4	5	1	6	5		
N	(1,2)	(1,3)	(2,3)	(2,4)	(2,5)	(3,4)	(3,5)	(4,5)	(4,6)	(5,6)	y_1	y_2
1	6	2	0	0	4	0	1	1	1	5	290	356
2	6	2	0	1	3	0	1	1	2	4	290	356
3	6	2	0	2	2	0	1	1	3	3	290	356
4	6	2	0	3	1	0	1	1	4	2	290	356
5	6	2	0	4	0	0	1	1	5	1	290	356
6	6	2	1	0	3	0	2	1	1	5	290	356
7	6	2	1	1	2	0	2	1	2	4	290	356
8	6	2	1	2	1	0	2	1	3	3	290	356
9	6	2	1	3	0	0	2	1	4	2	290	356
10	6	2	2	0	2	0	3	1	1	5	290	356
11	6	2	2	1	1	0	3	1	2	4	290	356
12	6	2	2	2	0	0	3	1	3	3	290	356
13	6	2	3	0	1	0	4	1	1	5	290	356
14	6	2	3	1	0	0	4	1	2	4	290	356
15	6	2	4	0	0	0	5	1	1	5	290	356
16	6	2	0	0	4	0	1	0	2	4	292	350
17	6	2	0	1	3	0	1	0	3	3	292	350
18	6	2	0	2	2	0	1	0	4	2	292	350
19	6	2	0	3	1	0	1	0	5	1	292	350
20	6	2	0	4	0	0	1	0	6	0	292	350
21	6	2	1	0	3	0	2	0	2	4	292	350
22	6	2	1	1	2	0	2	0	3	3	292	350
23	6	2	1	2	1	0	2	0	4	2	292	350
24	6	2	1	3	0	0	2	0	5	1	292	350
25	6	2	2	0	2	0	3	0	2	4	292	350
26	6	2	2	1	1	0	3	0	3	3	292	350
27	6	2	2	2	0	0	3	0	4	2	292	350
28	6	2	3	0	1	0	4	0	2	4	292	350
29	6	2	3	1	0	0	4	0	3	3	292	350
30	6	2	4	0	0	0	5	0	2	4	292	350
31	5	3	0	3	0	0	2	1	4	2	293	331
32	5	3	0	2	1	0	2	1	3	3	293	331
33	5	3	0	1	2	0	2	1	2	4	293	331
34	5	3	0	0	3	0	2	1	1	5	293	331
35	5	3	1	2	0	0	3	1	3	3	293	331
36	5	3	1	1	1	0	3	1	2	4	293	331
37	5	3	1	0	2	0	3	1	1	5	293	331
38	5	3	2	1	0	0	4	1	2	4	293	331
39	5	3	2	0	1	0	4	1	1	5	293	331
40	5	3	3	0	0	0	5	1	1	5	293	331
41	5	3	0	0	3	0	2	0	2	4	295	325
42	5	3	0	1	2	0	2	0	3	3	295	325
43	5	3	0	2	1	0	2	0	4	2	295	325
44	5	3	0	3	0	0	2	0	5	1	295	325
45	5	3	1	0	2	0	3	0	2	4	295	325
46	5	3	1	1	1	0	3	0	3	3	295	325
47	5	3	1	2	0	0	3	0	4	2	295	325
48	5	3	2	0	1	0	4	0	2	4	295	325

continued on the next page

u_{ij}	6	6	5	4	5	4	5	1	6	5		
N	(1,2)	(1,3)	(2,3)	(2,4)	(2,5)	(3,4)	(3,5)	(4,5)	(4,6)	(5,6)	y_1	y_2
49	5	3	2	1	0	0	4	0	3	3	295	325
50	5	3	3	0	0	0	5	0	2	4	295	325
51	4	4	0	2	0	0	3	1	3	3	296	306
52	4	4	0	1	1	0	3	1	2	4	296	306
53	4	4	0	0	2	0	3	1	1	5	296	306
54	4	4	1	1	0	0	4	1	2	4	296	306
55	4	4	1	0	1	0	4	1	1	5	296	306
56	4	4	2	0	0	0	5	1	1	5	296	306
57	4	4	0	2	0	0	3	0	4	2	298	300
58	4	4	0	1	1	0	3	0	3	3	298	300
59	4	4	0	0	2	0	3	0	2	4	298	300
60	4	4	1	1	0	0	4	0	3	3	298	300
61	4	4	1	0	1	0	4	0	2	4	298	300
62	4	4	2	0	0	0	5	0	2	4	298	300
63	3	5	0	0	1	0	4	1	1	5	299	281
64	3	5	0	1	0	0	4	1	2	4	299	281
65	3	5	1	0	0	0	5	1	1	5	299	281
66	3	5	0	0	1	0	4	0	2	4	301	275
67	3	5	0	1	0	0	4	0	3	3	301	275
68	3	5	1	0	0	0	5	0	2	4	301	275
69	2	6	0	0	0	0	5	1	1	5	302	256
70	2	6	0	0	0	0	5	0	2	4	304	250
71	2	6	0	0	0	1	4	0	3	3	316	244
72	2	6	0	0	0	2	3	0	4	2	328	238
73	2	6	0	0	0	3	2	0	5	1	340	232
74	2	6	0	0	0	4	1	0	6	0	352	226

E Branch-and-bound tree.

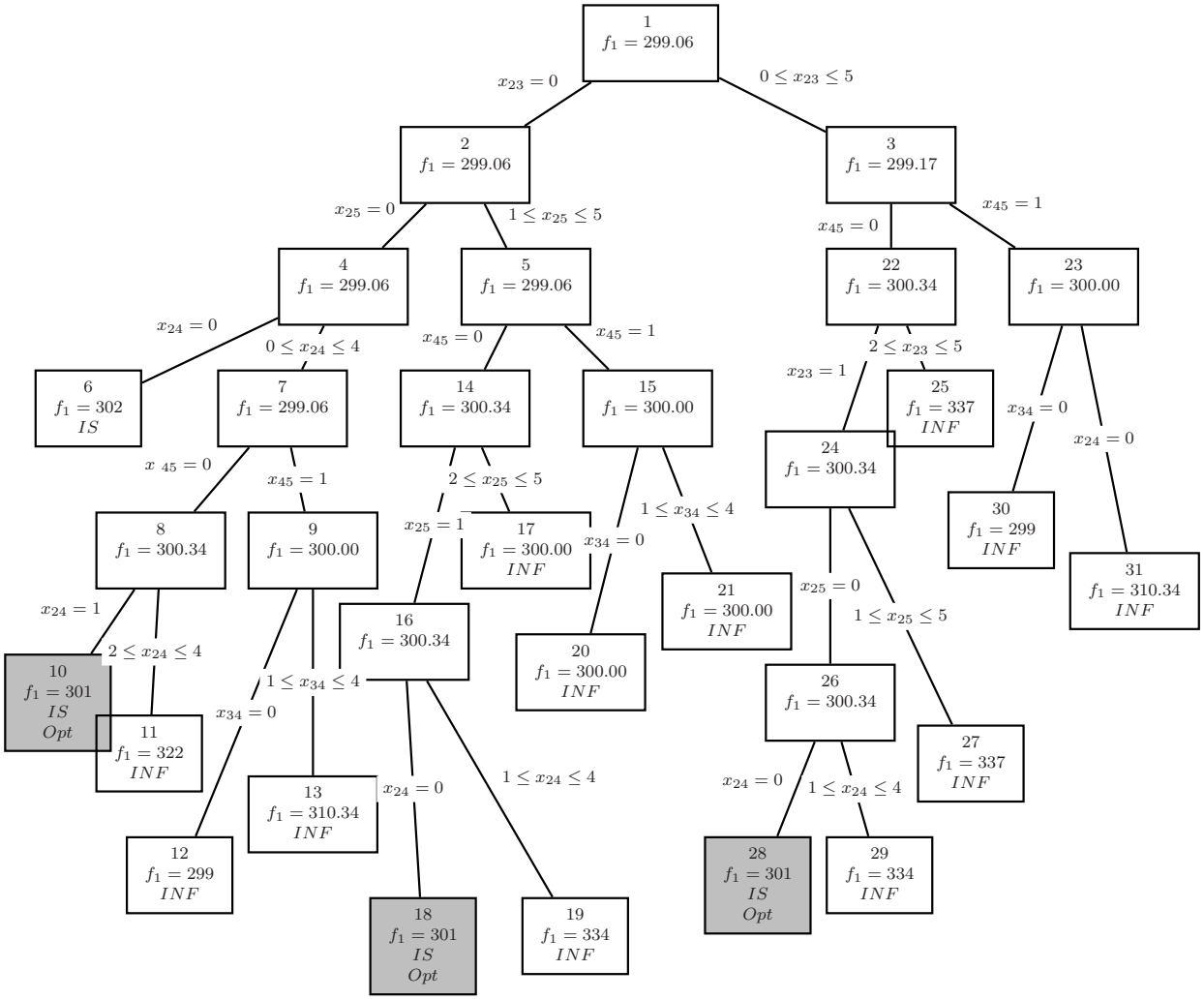


Figure 13: Branch-and-bound example for problem in Figure 8.