AN ALTERNATIVE METHOD TO SOLVE THE BIOBJECTIVE MINIMUM COST FLOW PROBLEM

Sedeño-Noda, A* and González-Martín, C.
Departamento de Estadística, Investigación Operativa y Computación (DEIOC), Universidad de La Laguna, 38271- La Laguna, Tenerife, Spain

Abstract

Single commodity flow problems simultaneously optimizing two objectives are addressed. We propose a method that finds all the efficient extreme points in the objective space. To accomplish this task, we introduce an auxiliary problem, with a single objective, that makes use of the weighted maximum distance. This problem allows making a guided search for efficient solutions taking into account the preferences of the decision-maker. The proposed method is characterized by the use of the classic resolution tools of network flow problems, such as the Network Simplex method. Multiobjective linear programming methodology is not use, and furthermore, the solution to the problem is determined directly in the objective space. Results of extensive tests are presented and discussed.

Keywords: Biobjective network flow problem; efficient extreme points; weighted maximum distance

1. Introduction

In the selection of the best solution on many network optimization problems more than one criterion can be considered. These criteria could be the minimization of arrival times at the point of destination, the cost minimization for selected routes, the minimization of the load capacity that would not be used in the selected vehicles, the minimization of the deterioration of goods, the maximization of safety, reliability, etc. The multicriteria minimum cost flow problems have already merited the attention of several authors. Aneja and Nair (1979) have studied the particular case of the bicriteria transportation problem, and have developed a procedure capable of generating the efficient extreme points in the objective space. Calvete and Mateo (1995) introduced a method for multiobjective network flow problems with preemptive priorities. This method is a lexicographic optimization approach. Figueira et al. (1998) proposed a local search procedure to test the robustness of a specific “satisfying point” neighbourhood. Isermann (1979) studied the multicriteria transportation problem and developed a procedure to identify the efficient region in the decision space, which was based on the properties of adjacency of the extreme points. Mustafa and Goh (1998) developed a method for adjusting into integral form the non-integer compromise solutions obtained via Dinas to interactively solve the bicriteria and tricriteria network flow problems. Klingman and Mote (1982) described a multiobjective Network Simplex method that is a specialization of the Multiobjective Simplex method of Yu and Zeleny (1976). Lee and Pulat (1991) developed a method to obtain all the efficient points of the bicriteria network problem in the objective space. This method will be discussed in detail later on. Also, Lee and Pulat (1993) considered a bicriteria network flow problem with flow variables restricted to integer values. Their procedure uses efficient solutions to the continuous version of bicriteria network flow problem. Maholtra and Puri (1984) introduced a method to obtain all the efficient points of

* Corresponding author.
E-mail address: asedeno@ull.es (A. Sedeño-Noda)
the bicriteria network problem in the objective space, which uses a version of the Out-of-Kilter algorithm. Pulat et al. (1992) described a bicriteria network flow procedure based on the parametric analysis to obtain all the efficient bases in the decision space. Ringuest and Rinks (1987) developed an interactive method, which also dealt with the multicriteria transportation problem. In this case, they used properties of adjacency of the extreme points on a particular auxiliary problem so that the point desired by the decision-maker could be obtained. Finally, Sedeño-Noda and González-Martín (2000b) proposed an algorithm for the biobjective integer minimum cost flow problem. A taxonomy to classify these problems is available in Current and Min (1986).

In this paper, we study the bicriteria minimum cost flow problem assuming that flows are continuous variables, taking advantage of the matrix properties of the feasible region. We propose a method that finds all the efficient extreme points in the objective space. Our method does not use the properties of adjacency of extreme points. The methods based on the exploration of adjacency must calculate degenerate solutions of the efficient basis solutions. Those methods have a very high computational cost, because more than 90% of the pivots in the Network Simplex method can be degenerate (Cunningham (1976)). We propose a method that is characterized by the use of the classic resolution tools of network flow problems, such as the Network Simplex method. It does not utilize the multiobjective linear programming methodology, as occurs in other works. We obtain efficient solutions through a single objective auxiliary problem in which search directions are incorporated. This guided search allows the generation of all efficient extreme points in the objective space. The process can be adapted, with no additional effort, to a variety of interactive schemes, in which only possible performance limits of the decision-maker are relevant.

The method of Lee and Pulat (1991) and the method of Sedeño-Noda and González-Martín (2000a) were designed to compute all extreme points in the objective space of the biobjective minimum cost flow problem. In this paper, we propose a method that spends less CPU time than the method of Lee and Pulat (1991) for sparse networks and that is slower than the method given in Sedeño-Noda and González-Martín (2000a) in all cases. In this sense, what is the advantage of the method described in this paper? The answer is that the proposed method allows making a guided search for efficient solutions taking into account the preferences of the decision-maker. That is, the method can be used to calculate the set of all efficient extreme points in the objective space or a subset of them belonging to a special region in such space. Furthermore, we use the proposed method when we are only interested in a subset of the efficient extreme points in a particular region of the objective space.

After this introduction, in the second section we give the mathematical notation of the problem. In the third, we introduce the basic concepts and propositions, which support the formalization and solving of the auxiliary problem. In this section, we develop a procedure for a directed search of the efficient solutions. In the fourth section, we present the procedure, which generates all the efficient extreme points, and we provide an example. In the fifth section, results of extensive tests are presented and discussed.

2. Formalization of the problem

Given a directed network \( G = (V, A) \), let \( V = \{1, \ldots, n\} \) be the set of nodes and \( A \) be the set of arcs. Let \( \text{Suc}(i) \) be the set \( \{j \in V \mid (i, j) \in A\} \) and \( \text{Pred}(i) \) be the set \( \{j \in V \mid (j, i) \in A\} \). Let an integer \( b_i \) be the supply/demand rate of node \( i \). Each arc \( (i, j) \in A \) has associated the following values: \( u_{ij} \), the upper bound on flow through arc \( (i, j) \) and \( l_{ij} \), the lower bound on
flow through arc \((i, j)\). Let \(c^k_{ij}\) be the cost per unit of flow on arc \((i, j)\) in the \(k\)th objective function, \(k=1,2\).

If \(x_{ij}\) denotes the amount of flow on an arc \((i, j)\), the biobjective network flow (BNF) problem can be formalized in the following way:

\[
\begin{align*}
\text{Min} & \quad f_1(x) = \sum_{i \in \mathcal{V}} \sum_{j \in \text{Suc}(i)} c^1_{ij} x_{ij} \\
\text{Min} & \quad f_2(x) = \sum_{i \in \mathcal{V}} \sum_{j \in \text{Pre}(i)} c^2_{ij} x_{ij} \\
\text{s.t.: } & \quad \sum_{j \in \text{Suc}(i)} x_{ij} - \sum_{j \in \text{Pred}(i)} x_{ij} = b_i, \forall i \in \mathcal{V} \\
& \quad l_{ij} \leq x_{ij} \leq u_{ij}, \forall (i, j) \in \mathcal{A}. 
\end{align*}
\]

Any vector \(x\) that satisfies (1.1b) and (1.1c) is called a feasible solution. The set of feasible solutions or decision space is denoted by \(X\). If we define \(f(x) = (f_1(x), f_2(x))\) then the image of \(X\) through \(f\) is \(f(X) = \{(f_1(x), f_2(x)) / x \in X\}\). Thus, \(f(X)\) is called the objective space.

3. Basic concepts and results

For \(k=1,2\), let \(x^*_k\) be a solution of the problem:

\[
\begin{align*}
f^*_k = f^*_k(x^*_k) = \min_{x \in \mathcal{X}} \sum_{i \in \mathcal{V}} \sum_{j \in \text{Suc}(i)} c^k_{ij} x_{ij} & \quad k = 1,2.
\end{align*}
\]

Then, \(f^*_k\) is the optimum value for the single objective flow problem that results when only the \(k\)th objective is taken into account.

In general, there is no feasible solution to the BNF problem that simultaneously minimizes \(f_1\) and \(f_2\). In other words, an optimum global solution does not exist. Therefore, the levels \(f^* = (f^*_1, f^*_2)\) define the ideal point.

As a result of this difficulty, the solutions of BNF problem are searched among the set of efficient points, that is, points that satisfy the following definition:

**Definition 1.** A feasible solution \(x \in X\) of the BNF problem is efficient if, and only if, there is no feasible solution \(x' \in X\) such that \(f_k(x') \leq f_k(x)\) for both \(k\) with \(f_k(x') \neq f_k(x)\) for at least one \(k\), \(k=1,2\).

We will denote by \(E[X]\) the set of efficient solutions of \(X\) and \(E[f(X)] = \{f(x) / x \in E[X]\}\). We refer to \(E[f(X)]\) as the set of efficient solutions of \(f(X)\).

The resolution of the BNF problem will be carried out by an adequate selection of one efficient solution from sets \(E[X]\) or \(E[f(X)]\). However, finding all the points in these sets can be very expensive since the number of efficient solutions in the objective space can be very large (assuming both feasibility and the ideal point to be infeasible). Nevertheless, given the characteristics of \(X\) and \(f(X)\) in this case, \(E[X]\) can be generated by a finite subset of its points: the efficient extreme points (see Yu and Zeleny (1976)). Moreover, the number of
extreme points can be very large (exponential in the number of nodes, see the example by Ruhe (1988)). Let \( E_e[X] \) be the set of the efficient extreme points of \( X \) and let \( E_{es}[f(X)] \) be the corresponding points of \( f(X) \). Our intention is to identify \( E_{es}[f(X)] \).

3.1. Generation of efficient solutions

To obtain efficient solutions the following single objective problem can be used:

\[
\min \ d_x(f^*, f(x)) \quad (3.1) \quad (P_x),
\]

where \( X \) is the decision space of BNF problem and \( d_x \) an appropriate distance, defined by a parameter given as \( \pi \) (see, for example, Steuer (1985)).

In the current paper, we propose the use of the weighted maximum distance (Tchebycheff metrics):

\[
d_x(f^*, f(x)) = \max_k \pi_k (f_k(x) - f_k^*)
\]

where the weights \( \pi \) are built in the following way: For each objective, the decision-maker introduces an aspiration level, \( z_k > f_k^*, k = 1,2 \), and then:

\[
\pi_k = \frac{1/(z_k - f_k^*)}{\sum_{i=1,2} 1/(z_i - f_i^*)}
\]

The problem \( P_x \) with metric \( d_x \) and weights \( \pi \) has a solution in the closest efficient point, in accordance with the search direction given by the ideal point and the aspiration level vector \( z \).

**Proposition 1** (see for example, Steuer (1985) or González-Martín (1986)).

- The solution of \( P_x \) is an efficient point of BNF problem on the intersection of the set of efficient points \( E[f(X)] \) with the line defined by \( f^* \) and the aspiration level vector \( z \).
- There exists a one-to-one correspondence between efficient solutions of BNF problem and weights \( \pi \) generated using the previous form.

3.2. Solving the \( P_x \) problem

In our case, the \( P_x \) problem can be expressed in the following way:

\[
\bar{y} = \min \ y \quad (3.2a)
\]

\[
s.t.: \pi_k \left( \sum_{i \in V \setminus \text{Suc}(i)} \sum_{j \in \text{Succ}(i)} c_{ij}^k x_{ij} - f_k^* \right) \leq y, \ k = 1,2 \quad (3.2b)
\]

\[
\sum_{j \in \text{Succ}(i)} x_{ij} - \sum_{j \in \text{Pred}(i)} x_{ji} = b_i, \ \forall i \in V \quad (3.2c)
\]

\[
l_j \leq x_{ij} \leq u_j, \ \forall (i, j) \in A. \quad (3.2d)
\]
The constraints (3.2c) and (3.2d) are the classic constraints of the minimum cost flow problem in a network. The restriction (3.2b) is related to the aspiration level for each objective, incorporated into the problem by the weights $\pi$. The objective (3.2a) involves minimizing the greater deviations of the values reached by the different objectives and the corresponding ideal points.

The restriction (3.2b) breaks the matrix’s unimodularity property of the minimum cost flow problem. However, this difficulty can be overcome if the constraint (3.2b) is incorporated into the objective function by the Lagrangian Relaxation:

$$\begin{align*}
\min & \quad y + \sum_{k=1}^{2} w_k \left( \sum_{j \in Suc(i)} \sum_{j \in Pred(i)} c_{ij}^k x_{ij} - f_k^* \right) \\
\text{s.t.} & \quad \sum_{j \in Suc(i)} x_{ij} - \sum_{j \in Pred(i)} x_{ji} = h_i, \forall i \in V \\
& \quad l_i \leq x_{ij} \leq u_{ij}, \forall (i,j) \in A \\
& \quad w_k \geq 0, k = 1,2
\end{align*}$$

(3.3d)

where $w$ is the Lagrangian multiplier vector.

By reordering the terms of (3.3a) the following objective function is obtained:

$$y(1 - \sum_{k=1}^{2} w_k) - \sum_{k=1}^{2} w_k \pi_k f_k^* + \sum_{k=1}^{2} \sum_{j \in Suc(i)} \sum_{j \in Pred(i)} c_{ij}^k x_{ij}.$$  

(3.3c)

It may be noted that the first two terms of the previous expression do not depend on $x$. In this case, once $w$ is fixed, the single objective minimum cost flow problem to solve is:

$$L_1(w) = \min \left( \sum_{j \in Suc(i)} \sum_{j \in Pred(i)} \sum_{k=1}^{2} w_k \pi_k c_{ij}^k x_{ij} / \text{s.t.: (3.3b), (3.3c)} \right).$$  

(3.4)

We define:

$$L(w) = L_1(w) - \sum_{k=1}^{2} w_k \pi_k f_k^* + y(1 - \sum_{k=1}^{2} w_k).$$

(3.3a)

Now, we solve the Lagrangian Multiplier (LM) problem: $\overline{L} = \max_{w \geq 0} L(w)$.

If $\overline{y}$ is the optimum solution of $P_{\overline{\pi}}$ for any choice of the Lagrangian multiplier vector $w$, and any feasible solution $x$ of $P_{\overline{\pi}}$ then $L(x) \leq \overline{L} \leq \overline{y} \leq y$. Furthermore, if $w$ is a vector of Lagrangian multipliers and $x$ is a feasible solution for the problem $P_{\overline{\pi}}$ satisfying $L(w)=\overline{y}$, then $w$ is an optimum solution for LM problem and $x$ is an optimum solution for $P_{\overline{\pi}}$.

**Proposition 2.** The optimum value of problem LM is equal to the optimum value of $P_{\overline{\pi}}$ that is, $\overline{L} = \overline{y}$. 

5
Proof. Since that the solution of $P_\pi$ is the intersection point between the set of efficient points $E[f(X)]$ with the line defined by $f^*$ and the aspiration level vector $z$ (see proposition 1), then

$$y = \pi_k \left( \sum_{j \in f} \sum_{j \in \text{set}(i)} c_{ij} x_{ij} - f_k^* \right), \forall k = 1, 2. \text{ Therefore, (3.2a) equals (3.3a) and } \bar{y} = y.$$

Furthermore, the resolution of (3.2), can be carried out by an adequate solving of (3.4) and LM problems. We give initial values to the multiplier vector $w$ (for example, equal to 1) and we modify in each iteration the values of $w$, according to the scheme of the subgradient optimization technique, until $\bar{L}$ is calculated. In this process, we use the Network Simplex Method to solve problem (3.4). In the end, we will obtain the efficient extreme point closest to the efficient solution of $P_\pi$.

Let $\tilde{f} \in f(X)$ be an efficient solution of $P_\pi$ and $\tilde{f}_{\text{ex}} \in f(X)$ be the efficient extreme point closest to $\tilde{f}$ according to the weighted maximum distance $d_\pi$ (see Figure 1).

One problem that can arise is that the Network Simplex Method, when applied to solve (3.4), is not able to calculate $\tilde{f}$ if this point is not an extreme point of $f(X)$. In order to avoid this situation, upper bounds of $\bar{y}$ are obtained. For this reason, it is necessary to identify some extreme points of the $f(X)$ polyhedron, that is, to determine the lower frontier of the closest extreme points to the solution efficient point.

Let $H_p$ be a line segment defined by two extreme points (not necessarily adjacent) in the objective space. Then, we define $\tilde{f}$ as the intersection point between the line $t = f^* + \alpha(z - f^*)$ and $H_p$. We can assign to $y$ the value of $d_\pi$ that corresponds to $\tilde{f} = f^* + \alpha(z - f^*)$. In this sense, at the end of our approach $\tilde{f}$ will be equal to $\tilde{f}$, since $H_p$ coincides with the line segment defined by the two adjacent efficient extreme points, which are closest to the solution point.

In the biobjective case, the efficient solution $\tilde{f}$ is always obtained solving $P_\pi$. This is because, at the end of resolution of $P_\pi$, segment $H_p$ coincides with an efficient edge of polyhedron $f(X)$.

By proposition 1, we have that $\bar{y} = \pi_1(\tilde{f}_1 - f_1^*) = \pi_2(\tilde{f}_2 - f_2^*)$. Considering this result in $L(w)$, we obtain that $L(w) = L_1(w) - \sum_{k=1}^{2} w_k \pi_k f_k^* + \bar{y}(1 - \sum_{k=1}^{2} w_k)$ where the two last terms are known.

The subgradient optimization technique allows us to obtain $\tilde{f}$ and, therefore, $\bar{y}$. We store in a list the two closest extreme points in non-decreasing order of the values of $d_\pi = \max_{k=1,2} \left( \pi_k \left( f_k - f_k^* \right) \right)$.

At the beginning of the procedure, the segment $H_p$ is defined by the extreme points associated with the ideal levels of each objective, therefore $H_p$ is a list that contains these extreme points. Let $\tilde{f}_{\text{ex}}$ be the current best efficient extreme point. At the end of the algorithm, this point will be the solution point. Let $\tilde{f}_{\text{ex}}$ be the new extreme point that is
calculated in each iteration. The algorithm starts with \( w = 1 \) and \( \lambda = 2.0 \). In each iteration, \( L(w) \) is solved and an extreme point \( \tilde{f}_{ex} \) is obtained. If the value of \( d_\pi = \max_{k=1,2} \left( \pi_k (\tilde{f}_{ex} - f^*_k) \right) \) for \( \tilde{f}_{ex} \) is less than the value of \( d_\pi \) for any other extreme point in \( H_p \), then \( \tilde{f}_{ex} \) is added to \( H_p \) and the extreme point in the segment with the greatest value of \( d_\pi \) is eliminated from \( H_p \). The new segment must have a non-null intersection with the line defined by \( f^* \) and \( z \). This will give a new point \( \tilde{f} \) and a new upper bound \( \tilde{y} \) which improves the previous value. This value is used to calculate the step size and the values of the multipliers of the following iteration. If the value of \( L(w) \) has not increased with respect to the values obtained in the \( r \) previous iterations, (in our performing \( r = 3 \)), the scale factor \( \lambda \) becomes equal to \( \lambda / 2 \). The step size, \( \theta \), is calculated according to Newton’s method for solving systems of non-linear equations. As the optimum \( \tilde{y} \) must coincide with \( \tilde{L} \), we can use the stop rule: \( \tilde{y} = L(w) \).

The algorithm has the following scheme:

**Algorithm SP\_π (Solving P\_π)**

\( H_p \) is a list containing the extreme points associated to the ideal levels of each objective (line segment);

\( \tilde{f} \) is the point obtained from the intersection between \( H_p \) and the search direction;

\( \tilde{y} = \pi_k (\tilde{f}_k - f^*_k) \), \( k = 1,2 \);

\( \tilde{f}_{ex} \) := First point of list \( H_p \);

\( q := 1 \); {iteration}; \( \lambda := 2.0 \); {scale factor to update the step size};

\( w^0_k := 1; k = 1,2 \); {initialization of the Lagrangian multipliers}

While (\( \tilde{y} \neq L(w) \)) do

begin

Solve \( L_1(w^0) \) using the Network Simplex method and obtain \( \tilde{f}_{ex} \);

if \( \tilde{f}_{ex} \) is closer than other extreme point in \( H_p \) then

begin

Interchange \( \tilde{f}_{ex} \) with the extreme point of \( H_p \) in such a way that the intersection with the search direction is non-null;

\( \tilde{f} \) is the point obtained from the intersection between \( H_p \) and the search direction;

\( \tilde{y} := \pi_k (\tilde{f}_k - f^*_k) \), \( k = 1,2 \);

\( \tilde{f}_{ex} := \) First point of list \( H_p \);

\( L(w^0) := L_1(w^0) - \sum_{k=1}^2 w^0_k \pi_k f^*_k + \tilde{y}(1 - \sum_{k=1}^2 w^0_k) \)

end;

if \( L(w) \) has not improved after \( r \) iterations then \( \lambda := \lambda / 2 \);

\( \theta := \frac{\lambda(\tilde{y} - L(w))}{\| \pi_k (\tilde{f}_{ex} - f^*_k) \|} \), \( k = 1,2 \); {step size}

\( w^q_{k+1} := \begin{cases} w^q_k + \theta(\pi_k (\tilde{f}_{ex} - f^*_k)) & \text{if } w^q_k + \theta(\pi_k (\tilde{f}_{ex} - f^*_k)) > 0 \\ w^q_k & \text{other case} \end{cases} \)

\( q := q+1 \) {iteration increase}

end
3.3. An example

To illustrate the general ideas of this process, we will use an example presented by Aneja and Nair (1979) and used by Ringuest and Rinks (1987). This example is a biobjective transportation problem that can be converted easily into a biobjective minimum cost flow problem. The data of the problem are the following:

<table>
<thead>
<tr>
<th>Origin points</th>
<th>Destination points</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(1,4)</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>(1,5)</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>(8,6)</td>
<td>17</td>
</tr>
<tr>
<td>4</td>
<td>(9,8)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>(4,5)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>(4,10)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>(6,1)</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>(7,3)</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>(7,4)</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>(8,5)</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>(9,2)</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>(4,5)</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>(6,1)</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>(1,4)</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>(1,5)</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>(8,6)</td>
<td></td>
</tr>
</tbody>
</table>

where the value that appears in each cell of the table is \((c^i_j, c^2_j)\).

The following extreme points: \(f^1=(143,265)\), \(f^2=(156,200)\), \(f^3=(176,175)\), \(f^4=(186,171)\), \(f^5=(208,167)\) are the solutions to this problem. The ideal point is \(f^*= (143,167)\). If the aspiration level vector is \(z = (160,180)\), then the corresponding weights are \(\pi = (0.433333, 0.566667)\). Therefore, we must obtain the closest extreme point to the solution point in the given direction. The process is shown in Table 1.

In this table, \(q\) represents the iteration, \(H_p\) is the current search segment defined by the extreme points enclosed between braces, \(\tilde{f}\) is the best known solution and \(\tilde{y}\) is its objective value associated with problem \(P_\pi\); \(f_{ex}\) is the actual best efficient extreme point; \(w^\theta\) is the Lagrangian multiplier vector obtained in the iteration \(q\); \(f_{ex}\) is the extreme point obtained in the resolution of \(L_1(w)\); \(\theta\) is the step size for the Lagrangian multipliers in the next iteration.

Table 1  
<table>
<thead>
<tr>
<th>(q)</th>
<th>(H_p)</th>
<th>(\tilde{f})</th>
<th>(\tilde{y})</th>
<th>(f_{ex})</th>
<th>(w^\theta)</th>
<th>(L_1(w^\theta))</th>
<th>(L(w^\theta))</th>
<th>(\theta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>({f^1, f^5})</td>
<td>(186.162,199.979)</td>
<td>18.688</td>
<td>(f^5)</td>
<td>(1,1)</td>
<td>(f^3)</td>
<td>174.433</td>
<td>6.672</td>
</tr>
<tr>
<td>2</td>
<td>({f^1, f^3})</td>
<td>(171.064,188.461)</td>
<td>12.161</td>
<td>(f^3)</td>
<td>(1.374,1)</td>
<td>(f^3)</td>
<td>203.967</td>
<td>7.472</td>
</tr>
<tr>
<td>3</td>
<td>({f^1, f^3})</td>
<td>(171.064,188.461)</td>
<td>12.161</td>
<td>(f^3)</td>
<td>(1.694,1)</td>
<td>(f^2)</td>
<td>227.829</td>
<td>10.300</td>
</tr>
<tr>
<td>4</td>
<td>({f^2, f^3})</td>
<td>(167.445,185.693)</td>
<td>10.593</td>
<td>(f^3)</td>
<td>(1.662,1.053)</td>
<td>(f^3)</td>
<td>231.104</td>
<td>10.374</td>
</tr>
<tr>
<td>5</td>
<td>({f^2, f^3})</td>
<td>(167.445,185.693)</td>
<td>10.593</td>
<td>(f^3)</td>
<td>(1.694,1)</td>
<td>(f^2)</td>
<td>227.829</td>
<td>10.300</td>
</tr>
<tr>
<td>6</td>
<td>({f^2, f^3})</td>
<td>(167.445,185.693)</td>
<td>10.593</td>
<td>(f^3)</td>
<td>(1.662,1.053)</td>
<td>(f^3)</td>
<td>231.104</td>
<td>10.374</td>
</tr>
<tr>
<td>7</td>
<td>({f^2, f^3})</td>
<td>(167.445,185.693)</td>
<td>10.593</td>
<td>(f^3)</td>
<td>(1.678,1.026)</td>
<td>(f^3)</td>
<td>229.723</td>
<td>10.593</td>
</tr>
</tbody>
</table>

Table 1 shows that the optimal solution of problem \(P_\pi\) is known from iteration 3, because the efficient extreme points \(f^2\) and \(f^3\) have been already obtained. The extreme points calculated in subsequent iterations will be \(f^2\) and \(f^3\) repeatedly. The algorithm ends in iteration 7, when \(L(w)\) becomes equal to \(\tilde{y}\). At the end of the algorithm, \(\tilde{f} = \tilde{f}\) is the efficient solution of the problem and \(\tilde{f}_{ex} = f^3\) is the closest extreme point to the solution point in the given direction. The
extreme point \( f^4 \) is not calculated, since it need not be computed to obtain the optimal solution of problem \( P_\pi \).

In the resolution of \( P_\pi \), the calculated extreme points belong to the cone that is defined by extreme points in the segment \( H_p \) and the ideal point. In each iteration, only an extreme point, which belongs to this cone, is calculated. If this extreme point fulfills the conditions given in the previous method, the new cone will be smaller than the previous cone. Furthermore, the points in \( \mathbb{R}^y \) belonging to the new cone have values for \( y \) lower than the points in the previous cones. In Figure 2, the successive cones for the example of Table 1 are shown.

Once \( P_\pi \) is solved, we can construct a method in order to obtain all the efficient extreme points in the objective space of BNF problem. Furthermore, we can design a method to obtain the efficient extreme points belonging to a special region of the objective space given by the decision-maker.

4. A method to obtain all the efficient extreme points of BNF problem

The method to calculate all the efficient extreme points is based on the previous ideas. Given a search direction, specified by an aspiration level vector \( z \) and the ideal point, we can find the closest efficient extreme point in accordance with the direction given by using the previous algorithm.

In the course of solving problem \( P_\pi \), we can take advantage of the fact that other efficient extreme points are calculated before the optimal solution is obtained. We can use this advantage to design a method to compute all the efficient extreme points.

At the beginning, we only know the efficient extreme points that minimize each objective separately, and the segment \( H_p \) is defined by these points. Then, given an adequate weight \( \pi \), the problem \( P_\pi \) is solved in the following way:

- If an extreme point not previously calculated is found (though it may not be the optimum for \( P_\pi \)), two segments are then built. These new segments are the result of the gradual substitution of the extreme point previously found by each one of the points of the segment.
- If a new extreme point is not found, however, it means that for the cone defined by the points in the segment and by the local ideal point, no other extreme point exists. As such, this segment is not worthy of further consideration.

The method uses a queue, \( Q \), that stores the segments that determine the cones to be examined. For each one of them, an arbitrary aspiration level vector \( z \) is given, which belongs to the examined cone; this determines the weight \( \pi \). The corresponding \( P_\pi \) problem is then solved. Once the current segment has been examined, we continue with the following segment in the queue. The method ends when the queue is empty. The proposed method has the following scheme:

\[
\text{Algorithm EEP; \{efficient extreme points\}}
\]

\[
H_{p \text{-current}} := \text{segment defined by the extreme points in the minimization for each objective;}
\]

\[
P_{\text{ex}} := \text{points of } H_{p \text{-current}};
\]

\[
\text{Init_queue (Q);}
\]

\[
\text{put in queue (} H_{p \text{-current})};
\]

\[
\text{While (} Q \neq \text{Null) do}
\]

\[
\begin{align*}
\text{begin} \\
H_{p \text{-current}} := \text{first_element (} Q); \\
f^*_z := \text{local ideal point associated with } H_{p \text{-current}}; \\
\text{New_extreme_point := false;}
\end{align*}
\]

9
\[
\pi := \text{build an adequate weight};
\]
\[
\text{SP}_\pi \left( H_{p-current}, f^*_L, \pi, \text{New\_extreme\_point}, f_{ext} \right) \text{ (stop solution of } P_\pi \text{ if a new efficient extreme point is obtained - } f_{ext} \text{ may not be the optimum for } P_\pi \};
\]
\[
\text{if New\_extreme\_point \ then}
\begin{align*}
& \text{begin} \\
& \quad \text{Pex := Pex + } f_{ext}; \\
& \quad \text{Add to the queue } Q \text{ the two segments, which result from the} \\
& \quad \text{interchange of } f_{ext} \text{ with each one of the points in } H_{p-current} \\
& \text{end}
\end{align*}
\]

The extreme points calculated by the algorithm are stored in \( \text{Pex} \). Let \( H_{p-current} \) be the search segment that is currently examined by the algorithm and that, together with the local ideal point \( f^*_L \), defines the search cone. Given a direction and the search cone, the associated \( P_\pi \) problem will then be solved. In the resolution of this problem, if a new extreme point (in the algorithm \( f_{ext} \)) is obtained, the process will be terminated. If this occurs, \( f_{ext} \) is introduced in \( \text{Pex} \), and the newly created segments are added to the queue \( Q \).

The vector \( z \) in each examined cone can be chosen in the following way. Let \( f^r \) and \( f^s \) be the extreme points that, along with the local ideal point \( f^*_L \), define the examined cone. Then, each \( z_k \) is equal to \( z_k = \left| f^r_k - f^s_k \right| / 2 + f^*_L, k = 1,2 \). There is no problem if vector \( z \) lies inside or outside the efficient boundary due to proposition 1 (\( z \) determines one weight \( \pi \)).

Figure 3 illustrates the idea of the method used for the example previously introduced. In Figure 3a), the first segment is defined by \( f^1 \) and \( f^5 \), because these are the points that minimize each objective separately. In this case, the local ideal point coincides with the ideal point \( f^* \). We suppose that the search direction is as indicated in the figure, and that by solving \( P_\pi \), \( f^2 \) is obtained (though the extreme point solution is \( f^3 \)). In figure 3b), once \( f^2 \) is found, the generated segments become visible. We now have two problems similar to the original, and as such, we give a direction for each cone and we solve the \( P_\pi \) problem associated with each of them.

In figure 3c), the two new segments (\( H_{p3} \) and \( H_{p4} \)) can be observed when the point \( f^3 \) has been found. We should therefore observe that in the cone defined by \( H_{p1} \), no other extreme point has been found, so, the method does not examine more than that region. In the figure 3d), two new segments, \( H_{p3} \) and \( H_{p6} \), are shown when \( f^4 \) is found. No other extreme point is found in \( H_{p3} \), and therefore, no new segments are generated. There are no new extreme points found in the cones \( H_{p3} \) and \( H_{p6} \) and so, the method ends having identified all the efficient extreme points.

**Proposition 3** (Validation of the algorithm). The previous algorithm finds all the efficient extreme solutions that belong to the search cones.

**Proof.** Let us suppose that the segment \( H_p \) and the local ideal point \( f^*_i \) define one of the search cones, and that only the extreme point \( f^q \) belongs to this cone. If there are more than one extreme points in the search cone, we are only interested in calculating at least one of them, because the remainders belongs to the new search cones generated from the previous calculated cone. Thus, we only have to prove that, if only the extreme point \( f^q \) belongs to this
cone, the algorithm obtains this point. In addition, because $f^q$ is an extreme point in the objective space, it does not belong to segment $H_p$. In other words, if $f^q$ belongs to segment $H_p$, then $f^q$ is not extreme in the objective space, although it can be extreme in decision space. Furthermore, in this case at the beginning of the algorithm, to solve $P_\pi$ we already have $\bar{f} = \bar{f}$ and therefore $\bar{y} = \bar{y}$.

Let an arbitrary aspiration level vector $z$ belong to this cone, the vectors $z$ and $f^*$ determine a vector $\pi$. Let $\bar{f}$ be the resulting point of the intersection of the direction with the segment $H_p$, and let $\bar{y}$ be its corresponding value for the $P_\pi$ problem. Let $\bar{f}$ be the efficient solution and $\bar{y}$ the optimum value of $P_\pi$. Clearly, $\bar{f}_k < \bar{f}_k$, $k = 1, 2$ and, therefore, $\bar{y} < \bar{y}$. Note that, $\bar{f}$ is obtained from the intersection of the direction with the segment $H_p$ containing $f^q$. For this reason, when solving $P_\pi$, the extreme point $f^q$ is calculated because it is closer than at least one of the points of $H_p$ to the solution efficient point. □

4.1. An example

Let us consider the example presented by Pulat et al. (1992) given in Figure 4.

The problem has the following four extreme points in the objective space $f^1 = (640, 2170)$, $f^2 = (660, 2060)$, $f^3 = (810, 1660)$, $f^4 = (1030, 1484)$ (in the decision space there are 8 efficient extreme points). The EEP algorithm identifies them, when the extreme points $f^1$ and $f^4$ are known.
In Table 2, \( \text{iter} \) represents the iteration, \( H_{\text{p-current}} \) is the current search segment defined by the extreme points enclosed between braces, \( \pi \) stores the weights that define the search direction. \( Q \) is the queue where all the segments are to be examined are stored, including the current one. \( f_{\text{ext}} \) is the new extreme point and \( P_{\text{ex}} \) is the list where the extreme points are stored.

Table 2, shows that in iterations 2, 4, and 5, no extreme point is found, because in the search cone \( H_{\text{p-current}} \) there are no extreme points. In iteration 3, all the extreme points are known.

<table>
<thead>
<tr>
<th>( \text{iter} )</th>
<th>( H_{\text{p-current}} )</th>
<th>( \pi )</th>
<th>( Q )</th>
<th>( f_{\text{ext}} )</th>
<th>( P_{\text{ex}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{ ( f^1 ), ( f^4 ) }</td>
<td>(0.637546,0.362454)</td>
<td>{ { ( f^1 ), ( f^4 ) } }</td>
<td>( f^3 )</td>
<td>{ ( f^1 ), ( f^4 ), ( f^3 ) }</td>
</tr>
<tr>
<td>2</td>
<td>{ ( f^3 ), ( f^4 ) }</td>
<td>(0.239130,0.760870)</td>
<td>{ { ( f^3 ), ( f^4 ) }, { ( f^1 ), ( f^3 ) } }</td>
<td>{ ( f^1 ), ( f^4 ), ( f^3 ) }</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>{ ( f^1 ), ( f^3 ) }</td>
<td>(0.750000,0.250000)</td>
<td>{ { ( f^1 ), ( f^3 ) } }</td>
<td>( f^2 )</td>
<td>{ ( f^1 ), ( f^4 ), ( f^3 ), ( f^2 ) }</td>
</tr>
<tr>
<td>4</td>
<td>{ ( f^1 ), ( f^2 ) }</td>
<td>(0.984399,0.015601)</td>
<td>{ { ( f^1 ), ( f^2 ) }, { ( f^2 ), ( f^3 ) } }</td>
<td>{ ( f^1 ), ( f^4 ), ( f^3 ), ( f^2 ) }</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>{ ( f^2 ), ( f^3 ) }</td>
<td>(0.833702,0.166298)</td>
<td>{ { ( f^2 ), ( f^3 ) } }</td>
<td>{ ( f^1 ), ( f^4 ), ( f^3 ), ( f^2 ) }</td>
<td></td>
</tr>
</tbody>
</table>

5. Computer implementation

In this section, we comment on the implemented code and give the computational results. A Network Simplex method for single objective network problems is implemented making use of the tree indices (\( \text{Pred} \), \( \text{Depth} \), \( \text{Thread} \)) to keep and update the feasible spanning tree. The prevention of cycling and stalling have been considered too. Our method uses Cunningham’s cycle prevention rule (Cunningham (1979)) and the LRB rule shown in Goldfarb et al. (1990) for anti-stalling. We have chosen the Network Simplex method for two reasons. The first reason is that the Network Simplex method permits us to solve any flow problem (the Out-of-Kilter method only solves the Circulation Flow Problem). The second reason is that computational studies show that the Network Simplex method is substantially faster than the Primal-Dual and Out-of-Kilter methods (see Glover et al. (1974)).

EEP is a PASCAL code that has been tested in a HP9000/712 workstation running at 60 MHZ. We have also implemented the method of Lee and Pulat (1991) and the method of Sedeño-Noda and González-Martín (2000a) in Pascal code. We refer to these methods as the L&P method and EEPO method, respectively.

The test problems were generated using NETGEN (Klingman (1974)). The cost values of the second objective function were uniformly generated in the interval \([-1000,1000]\) and the
cost values of the first objective are given by NETGEN. These values are integers. Table 3 shows the levels of the number of nodes \((n)\), the levels of the number of arcs \((m)\), the ratio \(m/n\) and the maximum capacity of arcs \((U)\):

Through the combinations of nodes \((n)\), arcs \((m)\) and maximum capacity \((U)\), we have obtained 36 network parameters. For each one of the previously network parameters, we have generated 5 replications by using NETGEN and the following seeds: 12345678, 36581249, 23456183, 46545174, 35826749. Therefore, the number of instances of the problem was 180.

| \(n\) | \(100, 175, 250\) | \(4, 7, 10\) | \(10, 1000, 100000\) |
| \(m\) | \(120, 210, 300\) | \(4, 7, 10\) | \(10, 1000, 100000\) |
| \(m/n\) | \(140, 245, 350\) | \(4, 7, 10\) | \(10, 1000, 100000\) |
| \(U\) | \(160, 280, 400\) | \(4, 7, 10\) | \(10, 1000, 100000\) |

In Table 4, the experimental results of the comparison among EEP, L&P and EEPO methods are shown (average of CPU time in seconds). From this table it can be observed that the EEP method spends less CPU time than the L&P method for sparse networks and that EEP is slower than the EEPO method in all cases. In Figure 5 the CPU time of EEP algorithm with respect to the efficient extreme points is shown.

**EEP**

![Fig. 5. CPU time with respect to mean of the efficient extreme points.](image)

6. **Conclusions**

In this paper we present a method to solve the biobjective minimum cost flow Problem. This method depends on the number of extreme points contained within the solution set of BNF problem. In other words, the worst-case complexity of the algorithm is the number of extreme points multiplied by the time used in the resolution of the \(P_\pi\) problem. This is because, each time a new extreme point is found, we do not persist in examining the corresponding cone. For each new extreme point, two new search segments are taken into account and for each of them, the \(P_\pi\) problem is solved. The worst-case complexity of the method is \(O(E_{\pi}[f(X)]S(P_{\pi}))\) where \(S(P_{\pi})\) is the worst-case complexity in the solving of \(P_{\pi}\).
In the resolution of problem $P_\pi$, the method takes at most 15 iterations (if $L(w)$ has not improved after 15 iterations, the resolution of $P_\pi$ is stop), because of that, the worst-case complexity of $S(P_\pi)$ is the complexity of the Network Simplex algorithm. The empirical behavior of this method is particularly good for instances of the BNF problem with a low number of efficient extreme points.

The presented method is generally applicable to any biobjective programming problems whose efficient region in the objective space is $R^2$-convex (Yu (1985)). Furthermore, it works in the objective space, and this is of interest in a number of applications. In addition, this method allows making a guided search for efficient solutions in this space taking into account the preferences of the decision-maker. That is, the method can be used to calculate the set of all efficient extreme points in the objective space or can be adapted to compute a subset of the set of efficient extreme points belonging to a special region in the objective space. For example, the algorithm can be used to search for an efficient solution according to certain aspiration levels. In addition, the algorithm $S\pi$ is easy to use in an interactive framework, where the decision maker repeatedly modifies $z$.

### Table 4
EEP test results

<table>
<thead>
<tr>
<th>$n$</th>
<th>$m$</th>
<th>$U$</th>
<th>EEP CPU time (s)</th>
<th>L&amp;P CPU time (s)</th>
<th>EEPO CPU time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>100</td>
<td>10</td>
<td>54,582</td>
<td>146,708</td>
<td>5,63</td>
</tr>
<tr>
<td>25</td>
<td>100</td>
<td>1000</td>
<td>94,364</td>
<td>189,094</td>
<td>6,782</td>
</tr>
<tr>
<td>25</td>
<td>100</td>
<td>10000</td>
<td>79,118</td>
<td>188,42</td>
<td>6,716</td>
</tr>
<tr>
<td>25</td>
<td>175</td>
<td>10</td>
<td>202,148</td>
<td>388,632</td>
<td>11,362</td>
</tr>
<tr>
<td>25</td>
<td>175</td>
<td>1000</td>
<td>333,132</td>
<td>496,594</td>
<td>12,728</td>
</tr>
<tr>
<td>25</td>
<td>175</td>
<td>10000</td>
<td>352,96</td>
<td>500,772</td>
<td>13,396</td>
</tr>
<tr>
<td>25</td>
<td>250</td>
<td>10</td>
<td>451,286</td>
<td>667,108</td>
<td>16,894</td>
</tr>
<tr>
<td>25</td>
<td>250</td>
<td>1000</td>
<td>889,368</td>
<td>899,994</td>
<td>18,658</td>
</tr>
<tr>
<td>25</td>
<td>250</td>
<td>10000</td>
<td>821,35</td>
<td>941,116</td>
<td>16,002</td>
</tr>
<tr>
<td>30</td>
<td>120</td>
<td>10</td>
<td>101,056</td>
<td>215,522</td>
<td>6,146</td>
</tr>
<tr>
<td>30</td>
<td>120</td>
<td>1000</td>
<td>139,898</td>
<td>293,36</td>
<td>6,37</td>
</tr>
<tr>
<td>30</td>
<td>120</td>
<td>10000</td>
<td>133,654</td>
<td>266,538</td>
<td>6,662</td>
</tr>
<tr>
<td>30</td>
<td>210</td>
<td>10</td>
<td>469,568</td>
<td>649,652</td>
<td>15,923</td>
</tr>
<tr>
<td>30</td>
<td>210</td>
<td>1000</td>
<td>572,69</td>
<td>878,166</td>
<td>16,454</td>
</tr>
<tr>
<td>30</td>
<td>300</td>
<td>10</td>
<td>1047,242</td>
<td>1396,16</td>
<td>24,894</td>
</tr>
<tr>
<td>30</td>
<td>300</td>
<td>1000</td>
<td>1527,454</td>
<td>1907,172</td>
<td>23,974</td>
</tr>
<tr>
<td>30</td>
<td>300</td>
<td>10000</td>
<td>1657,6</td>
<td>1421,592</td>
<td>25,943</td>
</tr>
<tr>
<td>35</td>
<td>140</td>
<td>10</td>
<td>125,764</td>
<td>302,828</td>
<td>8</td>
</tr>
<tr>
<td>35</td>
<td>140</td>
<td>1000</td>
<td>193,204</td>
<td>364,554</td>
<td>8,814</td>
</tr>
<tr>
<td>35</td>
<td>140</td>
<td>10000</td>
<td>194,402</td>
<td>389,442</td>
<td>8,912</td>
</tr>
<tr>
<td>35</td>
<td>245</td>
<td>10</td>
<td>552,332</td>
<td>680,866</td>
<td>16,278</td>
</tr>
<tr>
<td>35</td>
<td>245</td>
<td>1000</td>
<td>881,318</td>
<td>938,01</td>
<td>17,584</td>
</tr>
<tr>
<td>35</td>
<td>245</td>
<td>10000</td>
<td>988,812</td>
<td>1036,924</td>
<td>17,576</td>
</tr>
<tr>
<td>35</td>
<td>350</td>
<td>10</td>
<td>1441,506</td>
<td>1407,68</td>
<td>23,668</td>
</tr>
<tr>
<td>35</td>
<td>350</td>
<td>1000</td>
<td>2610,078</td>
<td>2007,482</td>
<td>29,16</td>
</tr>
<tr>
<td>35</td>
<td>350</td>
<td>10000</td>
<td>2491,548</td>
<td>2010,344</td>
<td>28,978</td>
</tr>
<tr>
<td>40</td>
<td>160</td>
<td>10</td>
<td>177,996</td>
<td>329,616</td>
<td>11,666</td>
</tr>
<tr>
<td>40</td>
<td>160</td>
<td>1000</td>
<td>425,458</td>
<td>557,298</td>
<td>13,666</td>
</tr>
<tr>
<td>40</td>
<td>160</td>
<td>10000</td>
<td>465,29</td>
<td>554,164</td>
<td>13,148</td>
</tr>
<tr>
<td>40</td>
<td>280</td>
<td>10</td>
<td>771,656</td>
<td>755,752</td>
<td>22,022</td>
</tr>
<tr>
<td>40</td>
<td>280</td>
<td>1000</td>
<td>1598,526</td>
<td>1300,388</td>
<td>24,978</td>
</tr>
<tr>
<td>40</td>
<td>280</td>
<td>10000</td>
<td>1557,068</td>
<td>962,248</td>
<td>24,33</td>
</tr>
<tr>
<td>40</td>
<td>400</td>
<td>10</td>
<td>1845,906</td>
<td>1411,896</td>
<td>32,98</td>
</tr>
<tr>
<td>40</td>
<td>400</td>
<td>1000</td>
<td>4027,008</td>
<td>2586,296</td>
<td>39,604</td>
</tr>
<tr>
<td>( n )</td>
<td>( m )</td>
<td>( U )</td>
<td>EEP CPU time (s)</td>
<td>L&amp;P CPU time (s)</td>
<td>EEPO CPU time (s)</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>40</td>
<td>400</td>
<td>100000</td>
<td>4580.706</td>
<td>2654.442</td>
<td>39.5</td>
</tr>
</tbody>
</table>

Acknowledgments

The authors are grateful to M. Colebrook for his remarks on the draft version of this paper. This work has been supported by a research project grant number 180221024 of the University of La Laguna.

References

The authors wish to thank the Associate Editor and the anonymous referees for their comments. Their suggestions have resulted in an improved paper. Also,