Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 6, No. 1, pp. 21–38, January 2010

A BRANCH & CUT TECHNIQUE TO SOLVE A WEIGHTED-SUM OF LINEAR RATIOS*

João Paulo Costa

Abstract: In this paper we present a new technique to compute the maximum of a weighted sum of the objective functions in multiple objective linear fractional programming (MOLFP). This problem is equivalent to the 'sum-of-ratios case' [14] problem. The new technique is an improvement of the technique presented in Costa [5] which is basically a Branch & Bound approach. Now a cut is introduced. This cut proves to speed up the technique in all the tests. On average the improvement ranges from 20% to 70% both in terms of running time and in the number of sub-regions Some computational results highlighting the performance of the technique are presented.

Key words: *multiobjective linear fractional programming (MOLFP), sum of linear ratios*

Mathematics Subject Classification: *90C32, 90C29*

1 Introduction

The weighted sum of functions is probably the widest multicriteria approach used in practice to aggregate the objective functions according to the preferences of the decision maker (DM). In multiple objective linear fractional programming (MOLFP) problems this aggregation leads to a fractional function where the linear numerator and denominator of each objective function turn out to be (in the general case) polynomials, the degree of which equals the number of objective functions. Thus, this transformation of an MOLFP problem into a single criterion problem leads to a very difficult problem to solve by the nowadays existing techniques. Schaible and Shi [14] consider it one of the most difficult fractional problems encountered so far – it is much more removed from convex programming than other multi-ratio problems. They provide a survey of applications and various algorithmic approaches for this problem.

In this paper we present a new technique (a new and faster version of the algorithm presented in Costa [5]) for computing the maximum of a weighted sum of the linear fractional objective functions. This means to compute the non-dominated solution of the MOLFP problem associated with a given weight vector for the objective functions. The basic idea of the technique presented in [5] is to divide (approximately by the middle) the non-dominated region in two sub-regions and to analyze each of them in order to discard one if it can be proved that the maximum of the weighted sum is in the other. The process is repeated with the remaining region.

*The work presented here was partially supported by FCT and FEDER, under research project POCI/EGE/58828/2004.

The technique is basically a branch and bound algorithm. Pardalos *et al.* [13] present a good outline of branch and bound principles used in global optimization. A synthesis of these principles is presented as follows. To solve a problem the search space is partitioned into subsets. For every subset an upper bound is estimated for the maximum of the objective function over the feasible points of that subset. On the basis of the information currently available (bounds and incumbent solutions) some subsets are discarded from further consideration, while the most promising subset is selected and further divided. This gives raise to a more refined partition, and the process is repeated. As Pardalos *et al* [13, p. 13] noticed "the computational burden of a branch and bound process usually increases exponentially with the dimension of the space in which branching is performed. Therefore, (...) it is important to have branching performed in a space of lowest possible dimension." The computational burden of branch and bound processes also increases (usually not exponentially) with the range of the search. It usually compensates to find constraints that reduce the number of potential solutions to search.

In the technique presented in this paper a cut is introduced which allows for smaller search trees. In the computational tests an improvement ranging from 20% to 70%, on average, was found. The biggest improvement was in the order of 90%. Indeed the new technique never performed worse than the one presented in [5]. The overall results suggest that the technique performs very well when compared with the nowadays existing techniques. However, it is very difficult to compare techniques, because several of the existing ones attempt to also solve other kinds of problems.

Kornbluth and Steuer [9] can be considered the seminal work in MOLFP. In their paper they present its main characteristics and difficulties. They also present one method to compute all the weakly non-dominated solutions of an MOLFP problem. Reference should be made to Steuer [16] for a good introduction to MOLFP. Stancu-Minasian ([15], Chap. 6) extends these concepts.

Schaible and Shi [14] present a good survey of techniques to the sum of ratios case and try to compare them. According to them, Kuno's technique [12] seems to be the existing technique that proved to have the highest performance so far. We would like to also emphasize the work of Falk and Palocsay [7], Konno and Yamashita [11], Konno and Fukaishi [10], Freund and Jarre [8], and Dai *et al.* [6] due to their important contributions to the field.

Finally, the algorithm presented by Benson [3] is probably the closest one to the technique presented in Costa [5]. It is an adaptation of the algorithm presented in Benson [2] to the particular case of the linear sum of ratios problem and it alleviates some of the needed assumptions to solve non-linear problems (namely the numerators being larger than zero). Section 4.3 details some of its characteristics, comparing them with the characteristics of the technique presented in this paper. The breakthrough was achieved by the explicit consideration of a sum of linear ratios as a multiobjective problem.

The paper will proceed as follows. Firstly, we present the notation and formulation of the problems to be solved. Secondly, we give an outline of the new computation technique. After that, we formally describe the technique. In Section 4, we present theoretical results supporting the validity of the technique and discuss some important issues. Section 5 presents some computational results. Finally, in the last section, we present the conclusions.

2 Notation

In this paper we formulate MOLFP problems in the following way:

$$\begin{array}{l} \max\left\{z_1 = \frac{c^1 x+\alpha_1}{d^1 x+\beta_1}\right\} \quad \cdots \quad \max\left\{z_p = \frac{c^p x+\alpha_p}{d^p x+\beta_p}\right\} \\ s.t. \quad x \in S = \{x \in R^n | Ax = b, x \geq 0\} \end{array} \tag{2.1}$$

Where c^k, $d^k \in R^n$, $A \in R^{m\times n}$, $b \in R^m$ and α_k, $\beta_k \in R$, $k = 1, \ldots, p$ and $d^k x + \beta_k > 0$, $\forall_k$, $\forall x \in S$. We assume that S is non-empty and bounded.

We will differentiate between weakly non-dominated solutions – a point $x' \in S$ is weakly non-dominated if and only if there does not exist another point $x \in S$ such that $z_k(x) > z_k(x')$, for all $k = 1, ..., p$ – and non-dominated solutions – a point $x' \in S$ is non-dominated if and only if there does not exist another point $x \in S$ such that $z_k(x) \geq z_k(x')$, for all $k = 1, ..., p$, and $z_k(x) > z_k(x')$ for at least one k. We will use the some denomination – non-dominated – in both the decision and objective spaces. That is, the image, z', in the objective space of the non-dominated solution $x' \in S$will also be called 'non-dominated'.

The weighted sum of the objective functions can be formulated as:

$$\begin{array}{l} \max\left\{\lambda_1 \frac{c^1 x+\alpha_1}{d^1 x+\beta_1} + \cdots + \lambda_p \frac{c^p+\alpha_p}{d^p x+\beta_p}\right\} \\ s.t. \quad x \in S \end{array} \tag{2.2}$$

In the last equation $\lambda \in R^p_+$ is defined according to the preferences of the decision making (DM). Usually we impose

$$\sum_{k=1}^{p} \lambda_k = 1 \tag{2.3}$$

in order to normalize the weights and $\lambda_k > 0$, k =1, ..., p, in order to prevent the result from being a weakly non-dominated solution. The maximum of a weighted sum of linear fractional functions is a non-dominated solution (if all the weights are strictly positive), but in general not all the non-dominated solutions can be computed using weighted sums of the linear fractional functions (only the supported solutions can be computed).

The ideal point, $z*$, is the point of the objective functions space whose coordinates are equal to the maximum that can be achieved separately by each objective function in the feasible region. $z*$ is computed through the determination of the pay-off table, that is, computing $z^k = z\left(x*^k\right)$, $k = 1, \ldots, p$, where $x*^k$ is non-dominated and optimizes the program:

$$\max z_k(x) \quad s.t. \quad x \in S \tag{2.4}$$

There is a variable change technique [4] that turns a linear fractional problem into a plain linear program. Consider the following single objective problem:

$$\begin{array}{l} \max\left\{z = \frac{cx+\alpha}{dx+\beta}\right\} \\ s.t. \quad x \in S = \{x \in R^n | Ax = b, x \geq 0\} \end{array} \tag{2.5}$$

Where $c, d \in R^n$, $\alpha, \beta \in R$ and $dx + \beta > 0, \forall x \in S$.

We define the new variables: $t = \frac{1}{dx+\beta}$ and $y = xt$

Making the variable substitution we arrive at the following linear program:

$$\begin{array}{l} \max\{z = cy + \alpha t\} \\ s.t. \quad Ay - bt = 0 \\ \qquad dy + \beta t = 1 \\ y \in R^n, y \geq 0, t \in R, t \geq 0 \end{array} \tag{2.6}$$

This variable change is extensively used by the technique presented in this paper (Section 3) in order to compute the maximum of each objective function of each sub-problem.

3 The Technique

The technique is a new and faster version of the technique presented in [5] for computing the maximum of the weighted sum of linear fractional objective functions. This means to compute the non-dominated solution of the MOLFP problem associated with a given weight vector for the objective functions. The basic idea of the Costa's technique [5] is to divide (approximately by the 'middle') the non-dominated region in two sub-regions and to analyze each of them in order to discard one if it can be proved that the maximum of the weighted sum is in the other. The process is repeated with the remaining region. It is not always possible to discard one of the regions and so the process must be repeated for both, building a search tree. In most problems, only after a certain level of the search tree regions can be discarded. The process ends when the remaining regions are so little that the differences among their non-dominated solutions are lower than a pre-defined error.

The division of the non-dominated region – Step 4, Section 3.1 – is modified in the technique presented in this paper because a cut is introduced. This cut also divides the non-dominated region in two sub-regions but Theorem 4.2, Section 4.1, guarantees that one of the sub-regions can be discarded. This speeds up the technique because the search tree is trimmed.

One region can be discarded when the value of the weighted sum of its ideal point is worse than the value of the weighted sum of a non-dominated solution belonging to another region not yet discarded – Theorem 4.1, Section 4.1. In order to speed up the computations, an incumbent non-dominated solution is maintained: the solution that has the best weighted sum computed so far. This incumbent solution is used to make the necessary comparisons with the ideal point of the regions that can potentially be discarded and to define the cut.

The cut is only introduced once in every second iteration of the technique. In some occasions the generated constraint does not effectively cut the non-dominated region. Step 4, Section 3.1, is divided in two parts: Step 4.1 – to divide the non-dominated region in two sub-regions – and Step 4.2 – to introduce the cut. These steps alternate.

It is also relevant to note that when analyzing a region, that is, when computing the pay-off table corresponding to that region, it is not necessary to perform all the pay-off table calculations. If we have the pay-off table of one region and we divide it in two sub-regions, we only need to compute half of the pay-off table solutions of the two new sub-regions. The solutions of the preceding pay-off table will be present in the pay-off tables of the two new sub-regions – Step 5, Section 3.1.

Having a tree to search, a criterion to choose the next region to divide is needed: the technique chooses the one having the best ideal point – Step 3, Section 3.1. Other criteria have already been tested: 1) to choose the region having the best non-dominated solution; and 2) to choose the region having the lowest index (next region). Some computational tests showed that the best performance is achieved by choosing the region with the best ideal point.

When there are no more regions to divide, the remaining regions have the range of their pay-off tables lower than the error and the technique stops – Section 3.1, Step 3.1.

3.1 Pseudo Code

Step 1 – Initializing

ε # is the pre-defined error.
λ # is the given weight vector.
t^I # is the incumbent region index.
V # is the index set of regions that can be further sub-divided.
$Q = \emptyset$ # is the index set of regions for which the range is lower than the error.
$t = 0$ # is the index of the current region.
$V = \{t\}$ # is the initial region index set.
$g = 0$ # is the region counter.
$S(t) = S$# is the feasible region of the t^{th} region.
$z^I = (-\infty, -\infty, \ldots, -\infty)$ ..# is the incumbent solution.
Branch $\leftarrow$ true # is the variable to decide either to Branch or to Cut

Step 2 – Analyzing the first region
 # Computing the pay-off table of the region t=0.
For k = 1, ..., p do
$x*^k = \arg\max_x \{z_k(x) : x \in S(t)\}$;
 # Note: The solutions can be weakly non-dominated.
$z^{kt} = z\left(x^{*k}\right)$;
End for
$z*^t = \left(z_1^{1t}, z_2^{2t}, ..., z_p^{pt}\right)$ # is the ideal point of region t=0;
 # Initializing the incumbent solution.
For j = 1, ..., p do
If $\sum_{k=1}^{p} \lambda_k z_k^{jt} > \sum_{k=1}^{p} \lambda_k z_k^I$ then $z^I \leftarrow z^{jt}$;
End for
$t^I \leftarrow t$.

Step 3 – Choosing the next region
 # If there are regions that can be further divided the algorithm chooses the one having the best ideal point to proceed, i.e., it defines the new t.
If $V \neq \emptyset$ then $t = \arg\max_{v \in V} \left\{ \sum_{k=1}^{p} \lambda_k z_k^{kv} \right\}$; $V \leftarrow V \setminus \{t\}$;
Else
Step 3.1 – Stopping
 # If there is no other region to further divide it is necessary to recalculate the pay-off tables of the not discarded regions (the ones remaining in Q), taking into consideration that we do not want weakly non-dominated solutions.
For all $q \in Q$ do:
Compute $z^{kq} = z\left(x*^k\right), k = 1, \ldots, p$
where $x*^k$ is non-dominated and optimizes the program:
$\max \{z_k(x) : x \in S(q)\}$;
End For
The non-dominated solution, $\bar{z}$, that maximizes the weighted sum of the objective functions is the one that maximizes: $\max_{q \in Q; j=1,\ldots,p} \sum_{k=1}^{p} \lambda_k z_k^{jq}$;
The algorithm stops.
End Else.

Step 4 – Branch or Cut

If *Branch* = *true* then

Step 4.1 – Branch

 # The index of the objective function to constrain, denoted by r, in order to sub-divide the region t, corresponds to the one having the largest range in the pay-off table, i.e.:

$r = \arg \max_{k=1,...,p} \left\{ \Delta z_k^t = \left(z_k^{kt} - \min_{j=1,...,p;j\neq k} \left\{ z_k^{jt} \right\} \right) \right\}$;

 # Creating two new regions:

$g \leftarrow g+1; \quad S(g) = S(t) \cap \{x \in R^n | z_r(x) \geq z_r^{rt} - 1/2 \Delta z_r^t\}$;

$g \leftarrow g+1; \quad S(g) = S(t) \cap \{x \in R^n | z_r(x) \leq z_r^{rt} - 1/2 \Delta z_r^t\}$;

Branch ← false *# Next time it will cut.*

Else

Step 4.2 – Cut

 # The cut corresponds to a constraint on one of the objective functions, denoted by r. It is guaranteed (Theorem 4.2, Section 4.1) that the r^{th} objective of the non-dominated solution that optimizes the weighted-sum is not bellow a value computed in the following way:

$r = \arg \max_{k=1,\cdots,p} \left\{ \tilde{z}_k^t - \min_{j=1,\cdots,p;j\neq k} \left(z_k^{jt} \right) \right\}$ where $\tilde{z}_k^t = \frac{\sum_{i=1}^{p} \lambda_i z_i^I - \sum_{i=1;i\neq k}^{p} \lambda_i z_i^{it}}{\lambda_k}$;

 # Creating one new region by introducing the cut:

$g \leftarrow g+1; \quad S(g) = S(t) \cap \{x \in R^n | z_r(x) \geq \tilde{z}_r^t\}$;

Branch ← *true; # Next time it will Branch*

End Else

Step 5 – Analyzing Regions

If *Branch* = *false* then

Step 5.1 – Analysing the two new regions resulting from the branch

 # Computing the pay-off table of the two new regions. Note that the maximum of each objective function in the previous region must belong to one of the new regions, and so there is no need to compute it.

For $k = 1, ..., p$ do

If $z_r^{kt} \geq z_r^{rt} - 1/2 \Delta z_r^t$ then

$z^{k(g-1)} = z^{kt}$;

$z^{kg} = z\left(x*^k\right)$ with $x*^k = \arg\max_x \{z_k(x) : x \in S(g)\}$;

 # The solutions can be weakly non-dominated.

Else

$z^{kg} = z^{kt}$;

$z^{k(g-1)} = z\left(x*^k\right)$ with $x*^k = \arg\max_x \{z_k(x) : x \in S(g-1)\}$;

 # The solutions can be weakly non-dominated.

End Else

End For

$z*^g = \left(z_1^{1g}, z_2^{2g}, ..., z_p^{pg}\right) \quad ; \quad z*^{(g-1)} = \left(z_1^{1(g-1)}, z_2^{2(g-1)}, ..., z_p^{p(g-1)}\right) \quad ;$

 # If g is an interesting region (a region where it is still possible to find the non-dominated solution that optimizes the weighted sum) it will be further analyzed, otherwise it will be just ignored (discarded).

If $\sum_{k=1}^{p} \lambda_k z_k^{*g} \geq \sum_{k=1}^{p} \lambda_k z_k^I$ then

 # Being an interesting region the incumbent solution will be compared with the solutions of the pay-off table of region g.

For $j = 1, ..., p$ do

If $\sum_{k=1}^{p} \lambda_k z_k^{jg} > \sum_{k=1}^{p} \lambda_k z_k^I$ then $z^I \leftarrow z^{jg}$; $t^I \leftarrow g$;

End For

 # Being an interesting region it will be further divided (if the range in the pay-off table of any objective function is larger than the error) or classified as a region to search for the non-dominated solution in the end.

If $\left(\exists_{k,j}\, k = 1, ..., p; j = 1, ..., p | z \underset{k}{*}^{g} - z_k^{jg} > \varepsilon\right)$ then $V \leftarrow V \cup \{g\}$;

Else $Q \leftarrow Q \cup \{g\}$;

End If

 # The same as above for the region (g-1).

If $\sum_{k=1}^{p} \lambda_k z_k^{*(g-1)} \geq \sum_{k=1}^{p} \lambda_k z_k^I$ then

For $j = 1, ..., p$ do

If $\sum_{k=1}^{p} \lambda_k z_k^{j(g-1)} > \sum_{k=1}^{p} \lambda_k z_k^I$ then $z^I \leftarrow z^{j(g-1)}$;$t^I \leftarrow (g-1)$;

End For

If $\left(\exists_{k,j}\, k = 1, ..., p; j = 1, ..., p | z \underset{k}{*}^{(g-1)} - z_k^{j(g-1)} > \varepsilon\right)$ then $V \leftarrow V \cup \{(g-1)\}$;

Else $Q \leftarrow Q \cup \{(g-1)\}$;

End If

Else

Step 5.2 – Analysing the region resulting from the cut

 # Computing the pay-off table of the new region.

For $k = 1, ..., p$ do

If $z_r^{kt} \geq \tilde{z}_r^t$ then $z^{kg} = z^{kt}$;

Else $z^{kg} = z\left(x*^k\right)$ with $x*^k = \arg\max_x \{z_k(x) : x \in S(g)\}$;

 # The solutions can be weakly non-dominated.

End For

$z*^g = \left(z_1^{1g}, z_2^{2g}, ..., z_p^{pg}\right)$;

 # If it is an interesting region, it will be further analyzed, otherwise it will just be ignored (discarded).

If $\sum_{k=1}^{p} \lambda_k z_k^{*g} \geq \sum_{k=1}^{p} \lambda_k z_k^I$ then

For $j = 1, ..., p$ do

If $\sum_{k=1}^{p} \lambda_k z_k^{jg} > \sum_{k=1}^{p} \lambda_k z_k^I$ then $z^I \leftarrow z^{jg}$ $t^I \leftarrow g$;

End For

If $\left(\exists_{k,j}\, k = 1, ..., p; j = 1, ..., p | z \underset{k}{*}^{g} - z_k^{jg} > \varepsilon\right)$ then $V \leftarrow V \cup \{g\}$;

Else $Q \leftarrow Q \cup \{g\}$;

End If

End Else

Step 6 – Discarding regions

 # If the incumbent solution has changed it pays to check if some more regions can be discarded

If ($t^I = g$) or (($t^I = (g - 1)$) and ($Branch = false$)) then

For all $v \in V$ do

If $\sum_{k=1}^{p} \lambda_k z \overset{v}{*}_k < \sum_{k=1}^{p} \lambda_k z_k^I$ then $V \leftarrow V \setminus \{v\}$;
End For
For all $q \in Q$ do
If $\sum_{k=1}^{p} \lambda_k z \overset{q}{*}_k < \sum_{k=1}^{p} \lambda_k z_k^I$ then $Q \leftarrow Q \setminus \{q\}$;
End For
End If
Return to Step 3.

4 Important Issues

4.1 The Condition to Discard Regions and the Cut Constraint

Let us introduce the set:

$$\Lambda = \left\{ (\lambda_1, ..., \lambda_p) : \lambda_i > 0, i = 1, ..., p, \sum_{j=1}^{p} \lambda_j = 1 \right\}$$

Theorem 4.1 (Condition to discard a region [5]). *Consider that $z*$ is the ideal point of region A of S and that z^1 can be achieved in region B of S.*

If $\sum_{k=1}^{p} \lambda_k z_k^* < \sum_{k=1}^{p} \lambda_k z_k^1$, $\lambda \in \Lambda$, then the non-dominated solution, $\bar{z}$, that maximizes $\sum_{k=1}^{p} \lambda_k z_k(x)$, $x \in S$, cannot be achieved in region A.

Proof. Consider that $\bar{z}$ can be achieved in region A. Thus,
$\sum_{k=1}^{p} \lambda_k \bar{z}_k \geq \sum_{k=1}^{p} \lambda_k z_k^1 > \sum_{k=1}^{p} \lambda_k z_k^*$ that is $\sum_{k=1}^{p} \lambda_k (\bar{z}_k - z_k^*) > 0$.
This last expression means that there is at least one k' for which $\bar{z}_{k'} > z_{k'}^*$ and so $z*$ would not be the ideal point of region A. This proofs the theorem. □

Theorem 4.2 (Cut constraint (Step 4.2)). *Consider that $z*$ is the ideal point of region A of S, that z^1 can be achieved in region B of S and that $\bar{z}$ is the non-dominated solution that maximizes $\sum_{k=1}^{p} \lambda_k z_k(x)$, $x \in S$, $\lambda \in \Lambda$.*

If $\tilde{z}_k = \frac{\sum_{i=1}^{p} \lambda_i z_i^1 - \sum_{i=1;i\neq k}^{p} \lambda_i z_i^*}{\lambda_k}$ then $\bar{z}$ cannot be achieved in region $C = A \cap \{x \in R^n : z_k(x) \leq \tilde{z}_k\}$.

Proof. $\tilde{z}_k = \frac{\sum_{i=1}^{p} \lambda_i z_i^1 - \sum_{i=1;i\neq k}^{p} \lambda_i z_i^*}{\lambda_k} \Leftrightarrow \lambda_k \tilde{z}_k + \sum_{i=1;i\neq k}^{p} \lambda_i z_i^* = \sum_{i=1}^{p} \lambda_i z_i^1$.

Because $\bar{z}$ is the solution that maximizes $\sum_{k=1}^{p} \lambda_k z_k(x)$ then $\lambda_k \tilde{z}_k + \sum_{i=1;i\neq k}^{p} \lambda_i z_i^* = \sum_{i=1}^{p} \lambda_i z_i^1 \leq \sum_{i=1}^{p} \lambda_i \bar{z}_i$.

Considering region C, then $z_k(x) \leq \tilde{z}_k$ and so the coordinate k of the ideal point of region C, z_k^{*C}, will also be $z_k^{*C}(x) \leq \tilde{z}_k$. The other coordinates of the ideal point of region C are equal to or lower than the respective coordinates of the ideal point of region A. Then $\lambda_k z_k^{*C} + \sum_{i=1;i\neq k}^{p} \lambda_i z_i^{*C} \leq \sum_{i=1}^{p} \lambda_i \bar{z}_i$. By Theorem 4.1 $\bar{z}$ cannot be achieved in region C. □

Corollary 4.3. *Consider that $z*$ is the ideal point of region A of S, that z^1 can be achieved in region B of S and that $\bar{z}$ is the non-dominated solution that maximizes $\sum_{k=1}^{p} \lambda_k z_k(x)$, $x \in S$, $\lambda \in \Lambda$.*

If $\tilde{z}_k = \frac{\sum_{i=1}^{p} \lambda_i z_i^1 - \sum_{i=1;i\neq k}^{p} \lambda_i z_i^*}{\lambda_k}, k = 1, \ldots, p.$ then $\bar{z}$ cannot be achieved in region $D = \bigcup_{k=1}^{p} (A \cap \{x \in R^n; z_k(x) \leq \tilde{z}_k\})$.

Proof. The proof stems directly from applying, p times, Theorem 4.2. □

This corollary was not used in the algorithm presented in Section 3. For implementation reasons (simplicity) only one constraint is considered in Step 4.2 and not the constraints on all the objective functions. All the constraints are computed, but then only the one that produces the deepest cut is considered at every second iteration of the algorithm.

4.2 The Condition to Stop

When there are no more regions to divide, the remaining regions have ranges in their pay-off tables lower than the error and the technique stops – Section 3.1, Step 3.1. These pay-off tables were computed in a straightforward way allowing for weakly non-dominated solutions. So the considered incumbent solution may be weakly non-dominated.

In Step 3.1 the pay-off tables of the remaining regions are recomputed, now guaranteeing that the solutions are non-dominated. This is achieved, in the current software implementation, through a lexicographic approach. For each solution z^k of the pay-off table, the maximum of each objective function (z_i^k, i= 1, ..., p, $i \neq k$) is recomputed individually, considering the remaining objective functions as constraints, defined by the so far achieved maximum. Other approaches could be used. This additional computation is not necessary to guarantee that the predefined error is not exceeded but it was considered, by the author, that this issue should not be disregarded due to the actual possibility of the incumbent solution to be weakly non-dominated. In the framework of multiobjective programming it is important to guarantee non-dominance. Moreover, computational tests were carried out to assess the computational burden of this task and it was considered negligible – usually there are only a few remaining regions (neither discarded nor divided). Notice that in Step 6 the set Q (remaining regions) is examined in order to further discard regions.

The pay-off tables can overestimate the nadir point for multiobjective problems. The minima of the objective functions over the non-dominated set are called nadir values and are the coordinates of the nadir point. Whereas the values of the ideal point are easy to obtain by simply maximizing each objective function individually over the feasible region, the nadir values are very hard to determine in the general case [1]. This implies that the range of the pay-off tables of the remaining regions could be underestimated and consequently the actual error could be bigger than the predefined error. However, with a small predefined error the remaining regions are severely constrained and this situation is atypical.

The remaining regions $S(q)$, $q \in Q$, are defined by S and several constraints on each objective function – Step 4, Section 3.1. Only two of these constraints, per objective function, can be non-redundant. So $S(q)$ can be defined as:

$$S(q) = S \cap \{x \in R^n : z_k(x) \leq U_k^q, z_k(x) \geq L_k^q, k = 1, ..., p\}$$

with $L_k^q, U_k^q \in R, k = 1, ...p$, where some L_k^q can be -∞ and some U_k^q can be $+\infty$.

Some computational tests were performed using an earlier version of the presented technique. In this earlier version the following algorithm was at the beginning of Step 3.1:

Step 3.1 – Stopping
For all $q \in Q$ do
For k = 1, ..., p do
If $\left(z *_{k}^{q} - L_{k}^{q}\right) > \varepsilon$ then # *Redefine the lower bounds of S(q)*
$L_{k}^{q} = z_{k}\left(x^{kq}\right)$ with $x^{kq} = \arg\min_{x} \{z_{k}(x) : x \in S(q)\}$
End if
End for
For k = 1, ..., p do
If $\left(z *_{k}^{q} - L_{k}^{q}\right) > \varepsilon$ then # *Branch again*
$g \leftarrow g+1; \quad S(g) = S(q) \cap \left\{x \in R^{n} : z_{k}(x) \geq 1/2\left(z *_{k}^{q} - L_{k}^{q}\right)\right\}; \quad V \leftarrow V \cup \{g\};$
$g \leftarrow g+1; \quad S(g) = S(q) \cap \left\{x \in R^{n} : z_{k}(x) \leq 1/2\left(z *_{k}^{q} - L_{k}^{q}\right)\right\}; \quad V \leftarrow V \cup \{g\};$
If $q \in Q$ then $Q \leftarrow Q \backslash \{q\}$;
End if
End for
End for
If $V \neq \emptyset$ then
Do the general case to compute the pay-off tables and analyse the regions belonging to V
Else
The algorithm proceeds as it is in Step 3.1, Section 3.1.
End if

The computation time of this task was considered negligible, but this additional complexity was dropped because all performed tests showed that the error is bounded by the range of the pay-off tables if the predefined error is small.

4.3 Comparing the Technique

Benson [3] presents an algorithm that is probably the closest to the technique presented in Costa [5]. It is an adaptation of the algorithm presented in Benson [2] for the particular case of the linear sum of ratios problem and it alleviates some of the needed assumptions to solve non-linear problems (namely the numerators being positive). The space to be branched is also the objective space (dimension p, being p the number of objective functions or ratios). However, while our technique considers only the non-dominated region, which is further trimmed by the cut, Benson's algorithm considers a box defined by the minimum and maximum of the denominators of the ratios over the feasible region. For each generated subregion the technique of Benson [3] solves one linear program with (p+1)×(n+1) constraints and 2×(p+1)+m variables. This linear program is obtained by applying some Lagrangean weak duality results. The technique presented in this paper solves on average p/2 linear programs, per each sub-region, with 1+m+2×p constraints and n+1 variables. The comparison of the

computational performances of the algorithms is very difficult because there are no computational results in [3]. More important than this issue is, probably, how to combine both techniques in order to achieve the solution of problems of higher dimension in a reasonable time.

5 Computational Tests

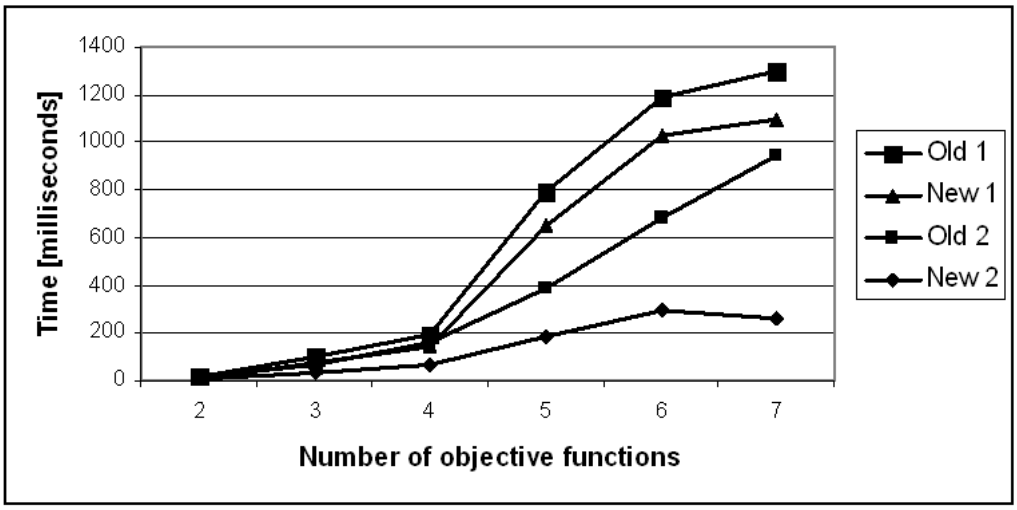

Figure 1: Performance of the Costa [5] and the new techniques according to time.

We will report some computational results from tests performed on randomly generated problems. The technique was coded in Delphi Pascal 5.0 for Microsoft Windows and a simplex code for solving linear problems was obtained through URL http://www.netcologne.de/~nc-weidenma/readme.htm, and adapted to the present case. The application was limited to problems with 200 (m) constraints and 140 (n) decision variables. There was no limit on the number of objective functions (p). The used data structures were dynamic arrays. All the data generated by the technique is kept in the data structures.

The tests used randomly generated problems according to Kuno [12]: data $c_{kj}, d_{kj} \in [0.0, 0.5]$ and $a_{ij} \in [0.0, 1.0]$ were uniformly distributed random numbers; b was set to constant and equals to one. In [12] all constant terms of denominators and numerators were the same number, which ranged from 2.0 and 100.0. Instead, in our tests $\alpha_k, \beta_k \in [2.0, 100.0]$ were also uniformly distributed random numbers. Each performance measure was obtained through the average of 20 runs, ignoring the two worst and two best values.

We used an Intel 6700, with 2 Gbytes of RAM, under the Windows XP Pack 2 operating system.

Fig. 1 presents the performance of the technique according to the elapsed time in milliseconds. The problems were generated with 10 decision variables and 10 constraints. The error was set equal to 0.001. Fig. 2 presents the number of generated regions for the same problems. The standard deviation of the measures whose averages are depicted in both figures is approximately equal to, but lower than the averages. These figures compare the performance of the algorithm presented by Costa [5], identified by 'Old', and the new algorithm presented in this paper, which is identified by 'New'. '1' refers to the performance of

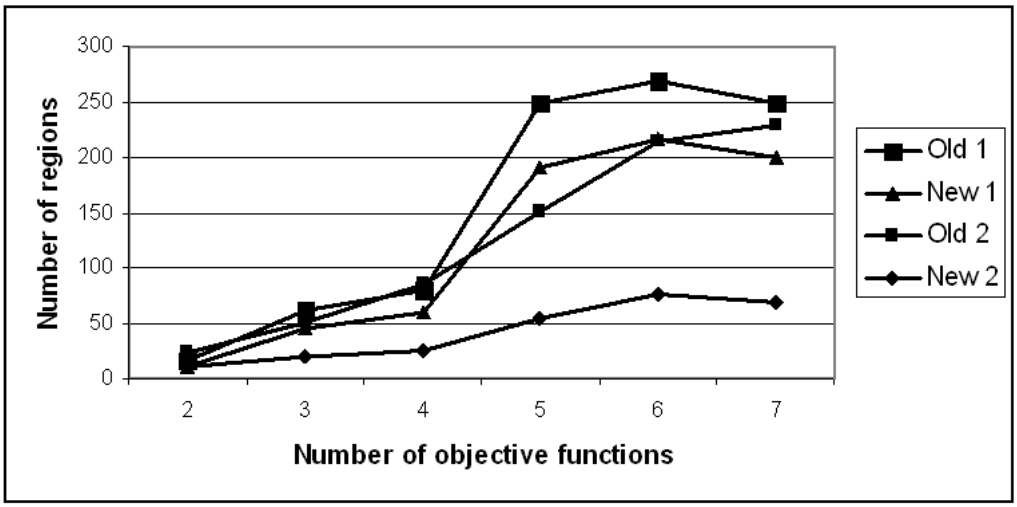

Figure 2: Performance of the Costa [5] and the new techniques according to the number of searched regions.

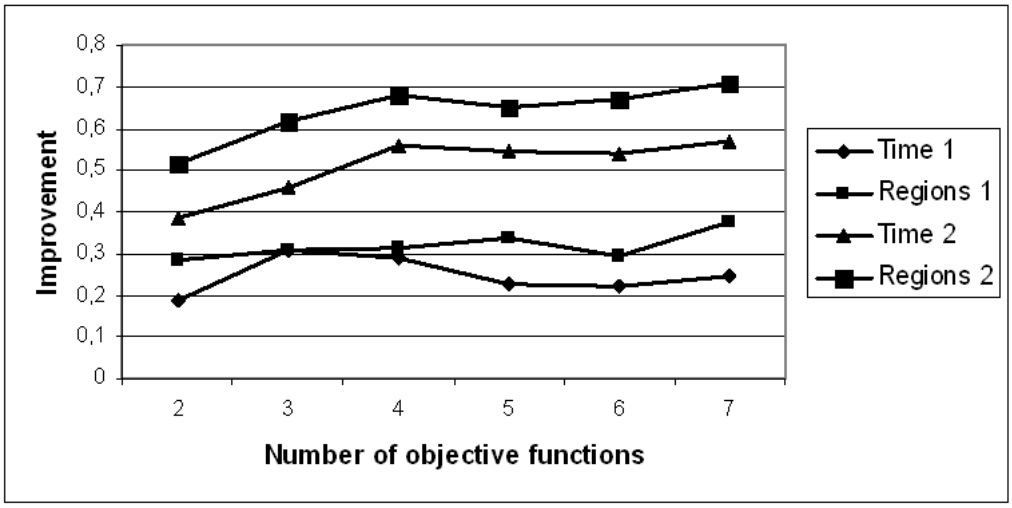

Figure 3: Improvement found in the tests reported in Fig. 1 and Fig. 2.

the algorithms using a weight vector for the objective functions with all components being equal, i.e. $\lambda_i = 1/p$, $i = 1, ..., p$. '2' refers to the performance of the algorithms with one of the weights much bigger than the others, all the others being equal among them, e.g. $\lambda_1 = 0.9$, $\lambda_i = 1/(p\text{-}1)$, $i = 2, ..., p$. Costa [5] noticed that the worst situation for the performance occurred when the weights were all equal. The results presented here confirm this situation.

Fig. 3 presents the achieved improvement with the new algorithm for the results presented in Fig. 1 and Fig. 2, which as been computed as follows: let N^{old} and N^{new} be the number of regions explored by the old and the new algorithm, respectively. Then the improvement is given by $(N^{old} - N^{new})/N^{old}$. The improvement on time is computed in the same way. One can see that the achieved improvement is approximately 20% for the situation where all the weights are equal and approximately 70% for the situation where there is one weight much bigger than the others. This was expected because the cut introduced in the algorithm presented in this paper is particularly good if the values of one of the objective functions are higher than the values of the others objective functions. It is worth noting here that the introduced cut is a constraint on the k^{th} objective function considering a division by its weight λ_k . In all the tests the lower improvement was 0% (cases where the algorithms were so fast that achieved the solutions in less than 1 millisecond) and the biggest was 90%. These last cases will be carefully studied in the future in order to explain why such great improvement is happening.

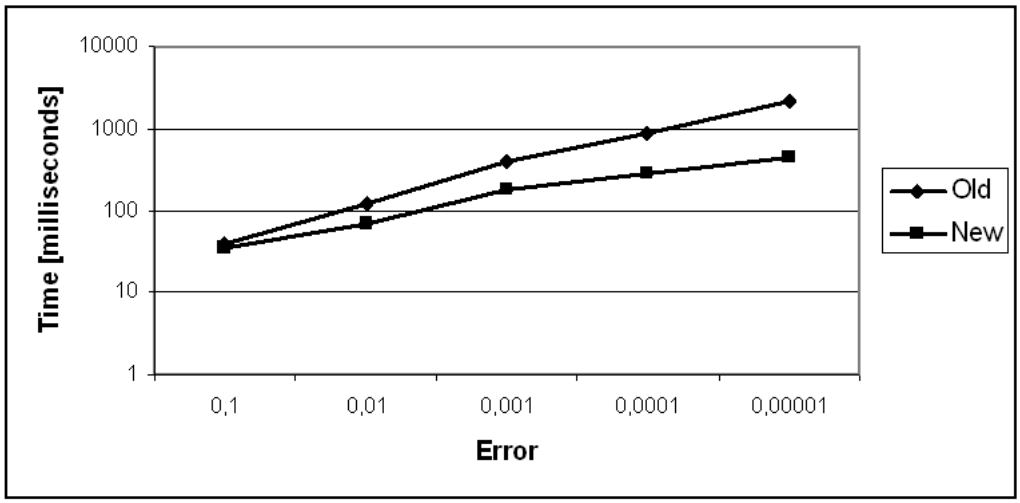

Figure 4: Performance of the Costa [5] and the new techniques according to time.

Fig. 4 and Fig 5 present the performances of the Costa [5] algorithm and the new algorithm when the error is changed. Please notice that both axes, on Figs 4 and 5, are on a logarithmic scale. The number of objective functions for all problems was set to 5. The problems were generated with 10 decision variables and 10 constraints. One can see that the improvement is greater when the error is reduced, that is in problems that consume more time. The weight vector of the objective functions was set for the best situation, that is one of the weights is much bigger than the others. The results presented in Fig. 4 and Fig. 5 confirm the conclusion already reached: the cut turns the algorithm faster. In problems that need more computation resources the improvement is more significant than in the ones requiring less resources.

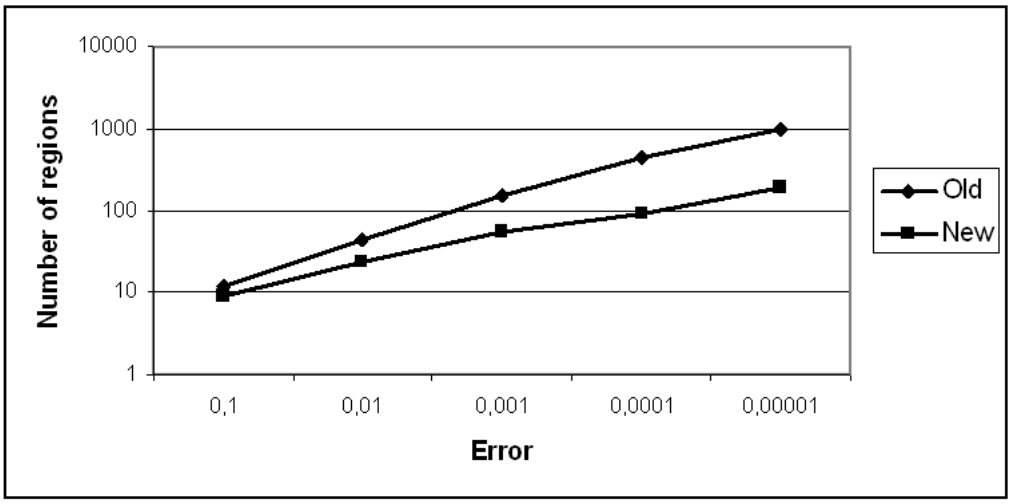

Figure 5: Performance of the Costa [5] and the new techniques according to the number of searched regions.

Fig. 6 and Fig 7 depict the results of a study of the Costa [5] algorithm and the new algorithm varying the number of constraints and variables of the problems. The number of objective functions for all problems was set to 3 and the error was set to 0.001. The weights for the objective functions were set to the worst situation, which is setting the weights all equal among themselves. The study was conducted with these parameters in order to assess the improvement in adverse circumstances.

Table 1: Improvement found in the tests reported in Fig. 6 (Time).

Improvement time	No. Constraints (m)				
No. Variables (n)	200	140	80	40	10
10	40,72%	30,81%	42,00%	30,11%	33,13%
40	38,11%	36,80%	35,49%	31,45%	30,25%
80	34,12%	33,27%	34,31%	35,23%	30,35%
140	34,11%	35,40%	34,57%	37,40%	34,12%

Note that the axe of 'Time' in Fig. 6 is on a logarithmic scale. The axe of 'Number of Regions' in Fig. 7 is not on a logarithmic scale. The increase of time with the growth of the number of variables and constraints is enormous when compared with the increase of the number of searched regions. This indicates that the computational burden associated with each region grows fast with the dimension of the original problem to be solved. It also indicates that there is much to gain in time on using already known optimal bases for a region as starting points for the needed computations of its child(ren). This was not included in the current implementation of the technique.

Tables 1 and 2 present the improvement of the new technique in relation to the old one in time and in the number of regions, respectively. Although being interesting, these results do not allow extracting any pattern or conclusion but the already known: the introduced cut works in all the considered situations.

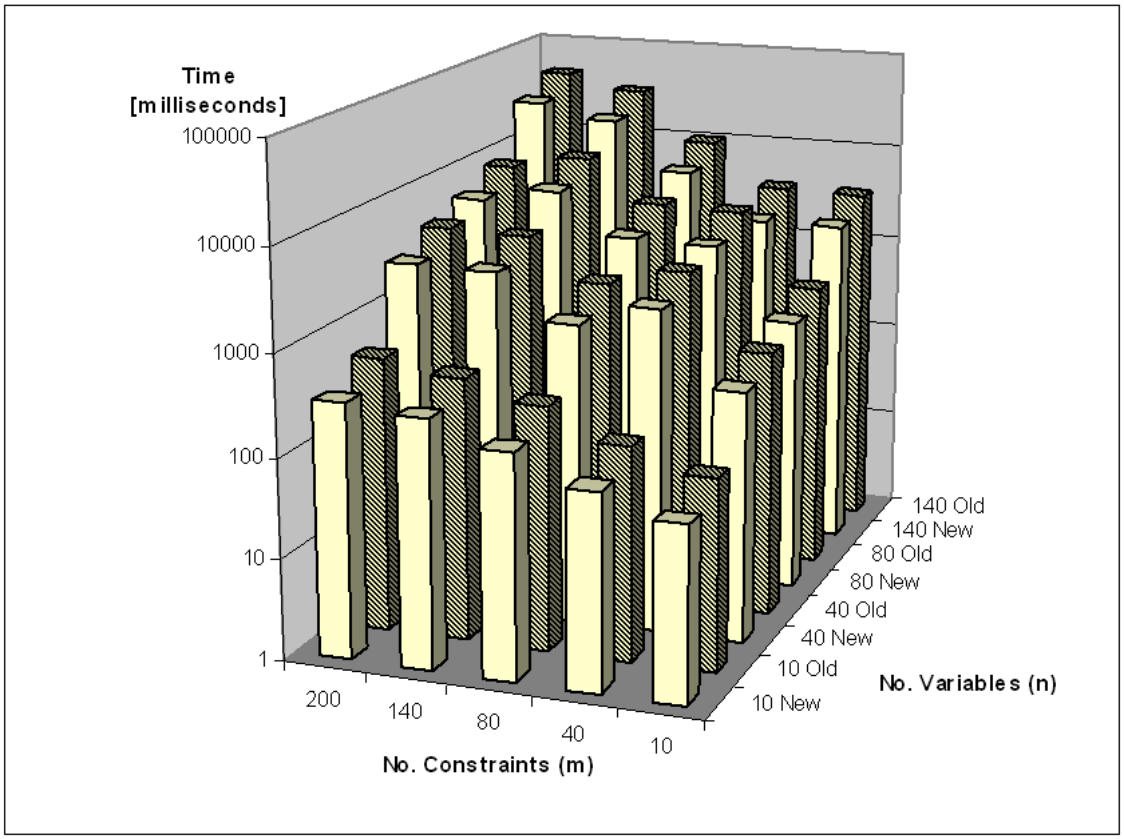

Figure 6: Performance of the Costa [5] and the new techniques according to time.

Table 2: Improvement found in the tests reported in Fig. 7 (Regions).

Improv. no. regions	No. Constraints (m)				
No. Variables (n)	200	140	80	40	10
10	40,86%	31,51%	39,28%	32,91%	40,72%
40	31,87%	28,79%	29,85%	24,89%	25,47%
80	25,95%	22,18%	27,61%	25,07%	27,37%
140	25,52%	28,58%	29,24%	31,39%	24,48%

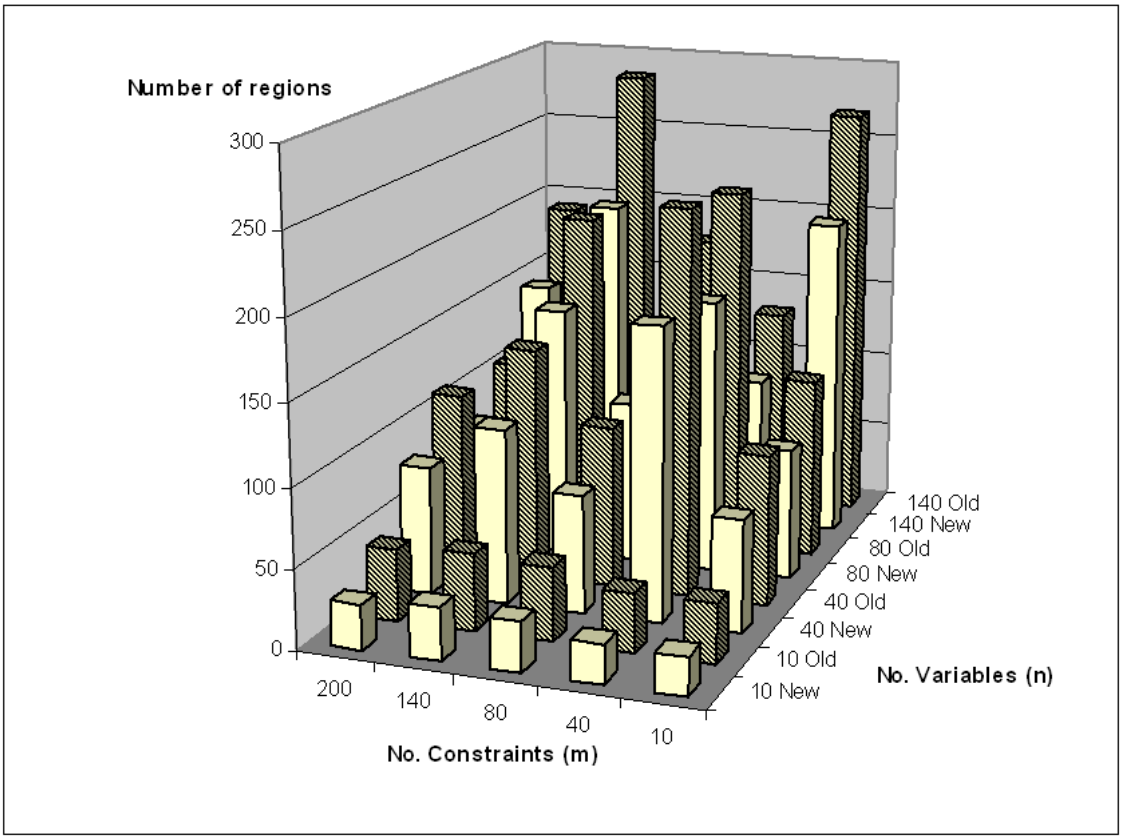

Figure 7: Performance of the Costa [5] and the new techniques according to the number of searched regions.

6 Conclusions

In this paper we presented a new technique to compute the maximum of a weighted sum of the objective functions in multiple objective linear fractional programming (MOLFP). This is an improvement of the technique presented in Costa [5] which is basically a Branch & Bound approach. Now a cut is introduced. Some computational results indicating the performance of the technique were presented. The cut proves to speed up the technique in all the tests. On average the improvement ranges from 20% to 70%. Nevertheless, we believe that the technique can be further improved by carefully choosing when the cut should be introduced. The cut is, by now, introduced in a 'blind' way.

Acknowledgement

The author gratefully acknowledges the helpful comments and suggestions of the anonymous referees. He would like to also thank Professor Maria João Alves for her help concerning the final version of the paper.

References

[1] M. Alves and J. Costa, An exact method for computing the Nadir values in multiple objective linear programming, *European Journal of Operational Research* 198 (2009) 637–646.

[2] H. Benson, Global optimization algorithm for the nonlinear sum of ratios problem, *Journal of Optimization Theory and Applications* 112 (2002) 1–29.

[3] H. Benson, A simplicial branch and bound duality-bounds algorithm for the linear sum-of-ratios problem, *European Journal of Operational Research* 182 (2007) 597–611.

[4] A. Charnes and W. Cooper, Programming with linear fractional functions, *Naval Research Logistics Quarterly* 9 (1962) 181–186.

[5] J. Costa, Computing non-dominated solution in MOLFP, *European Journal of Operational Research* 181 (2007) 1464–1475.

[6] Y. Dai, J. Shi and S. Wang, Conical partition algorithm for maximizing the sum of dc ratios, *Journal of Global Optimization* 31 (2005) 253–270.

[7] J. Falk and S. Palocsay, Optimizing the sum of linear fractional functions, in *Recent Advances in Global Optimization*, C Floudas and P Pardalos (eds.), Princeton Univ. Press, Princeton N.J, 1992, pp. 221–258.

[8] R. Freund and F. Jarre, Solving the sum-of-ratios problem by an interior-point method, *Journal of Global Optimization* 19 (2001) 83–102.

[9] J. Kornbluth and R. Steuer, Multiple objective linear fractional programming, *Management Science* 27 (1981) 1024–1039.

[10] H. Konno and K. Fukaishi, A branch and bound algorithm for solving low rank linear multiplicative and fractional programming problems, *Journal of Global Optimization* 18 (2000) 283–299.

[11] H. Konno and H. Yamashita, Minimizing sums and products of linear fractional functions over a polytope, *Naval Resarch Logistics* 46 (1999) 583–596.

[12] T. Kuno, A branch-and-bound algorithm for maximizing the sum of several linear ratios, *Journal of Global Optimization* 22 (2002) 155–174.

[13] P. Pardalos, H. Romeijin and H. Tuy, Recent developments and trends in global optimization, *Journal of Computational and Applied Mathematics* 124 (2000) 209–228.

[14] S. Schaible and J. Shi, Fractional programming: the sum-of-ratios case, *Optimization Methods and Software* 18 (2003) 219–229

[15] I. Stancu-Minasian, *Fractional Programming: Theory, Methods and Applications*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997.

[16] R. Steuer, *Multiple Criteria Optimization: Theory, Computation and Application*, Wiley, New York, 1986.

Manuscript received 30 March 2008
revised 27 October 2008
accepted for publication 12 March 2009

João Paulo Costa
Fac. Economia, Univ. Coimbra and INESC, Dias da Silva, 165, 3004-512 Coimbra, Portugal
E-mail address: jpaulo@fe.uc.pt