European Journal of Operational Research 124 (2000) 478–494

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/dsw

Theory and Methodology

An interactive reference point approach for multiobjective mixed-integer programming using branch-and-bound [1]

Maria João Alves *, João Clímaco

Faculdade de Economia, Universidade de Coimbra / INESC, Av. Dias da Silva, 165, 3000 Coimbra, Portugal

Received 14 October 1998; accepted 22 February 1999

Abstract

We propose an interactive reference point approach for multiple objective (mixed) integer linear programming problems that exploits the use of branch-and-bound techniques for solving the scalarizing programs. At each dialogue phase, the decision maker must specify a criterion reference point or just choose an objective function he/she wants to improve in respect to the previous efficient (nondominated) solution. In the latter case, a directional search is performed adjusting automatically the reference point used at each stage. Tchebycheff mixed-integer scalarizing programs are successively solved by branch-and-bound. Postoptimality techniques have been developed enabling the algorithm to profit from previous computations to solve the next scalarizing programs. The previous branch-and-bound tree is used as a starting point and operations of simplification and branching are then performed to obtain a new efficient solution. Computational results have shown that this approach is effective for carrying out directional or local searches for efficient solutions. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Multi criteria analysis; Mixed-integer linear programming; Reference points; Tchebycheff metric; Branch-and-bound; Sensitivity analysis

1. Introduction

In this paper we propose an interactive method for multiobjective mixed-integer (and pure integer) linear programming (MOMILP) problems.

Almost all the research work on multiobjective (mixed) integer programming has been developed since the final of 1970s. Among the first developments, we can mention the work of Bowman (1976), Zionts (1977) and Bitran (1977) – the latter is devoted to multiobjective programs with zero–one variables. Interested readers may refer to Evans (1984) for an overview of techniques. Since the early 1980s some researchers' attention has focussed on the design of interactive approaches to deal with these problems. Interactive methods have shown to be more effective to deal with

* Corresponding author. Tel.: 351-39-790-500; fax: 351-39-403-511.
E-mail address: mjoao@inescc.pt (M.J. Alves).

[1] This research was partially supported by PRAXIS XXI, project 2/2.1/MAT/465/94.

multiobjective problems than generating methods because they enable to reduce the computational effort – specially relevant in large-sized problems – and generally provide means to assist the decision maker (DM) in choosing his/her preferred solution. However, most of the methods developed so far are designed for pure integer problems and many of them do not apply to the mixed-integer case. In fact, research in multiobjective mixed-integer programming has been rather scarce. Concerning just the mixed-integer case, among the first multiobjective interactive approaches there are the branch-and-bound method of Villarreal et al. (1980) later improved by Karwan et al. (1985) and Ramesh et al. (1986), and the method of Steuer and Choo (1983) that is valid for more general multiobjective problems. For a review, see Teghem and Kunsch (1986). More recent references are Aksoy (1990), Solanki (1991), Durso (1992), L'Hoir and Teghem (1995) and Ferreira et al. (1996). They all refer to interactive approaches, except Solanki's method that seeks to determine a representative subset of efficient solutions not requiring the intervention of the DM. There are some other interactive methods, which were developed for multiobjective integer linear programming (MOILP), that also fit the MOMILP case. References of such methods are Karaivanova et al. (1993), Vassilev and Narula (1993), Narula and Vassilev (1994) and Karaivanova et al. (1995).

The increasing interest in MOMILP decision aid is due to the fact that most decision problems are inherently multiobjective and many of them must include discrete variables. Examples of different applications are the multicriteria scheduling problem (Huckert et al., 1980), nursing service budgeting (Trivedi, 1981), vendor selection (Weber and Current, 1993), the groundwater pollution containment problem (Ritzel et al., 1994), location–transportation problems (Osleeb and Ratick, 1983; Ferreira et al., 1996; Coutinho-Rodrigues et al., 1997), among others. White (1990) provides a survey of applications of multiobjective mathematical programming that includes the integer and mixed-integer cases.

The interactive method which we have developed and we present herein aims to provide a simple protocol to interact with the DM, not demanding too much information about his/her preferences, and aims to reduce the computational effort, namely by taking advantage of computations previously performed for producing new efficient solutions.

These ideas also underlay our previous research on the development of an interactive method for multiobjective pure integer linear programs. That method (Alves and Clímaco, 1999) combines Tchebycheff theory with cutting plane techniques. At each interaction, the DM must specify a reference point (aspiration levels for the criteria) or just the objective function he/she wants to improve in respect of the previous efficient solution. In the latter case, the reference point is automatically adjusted and new efficient solutions are obtained through a directional search. Cutting planes, which are used to solve the pure integer Tchebycheff scalarizing programs, facilitate the incorporation of sensitivity analysis used to adjust the reference point for the next computing phase. However, if cutting planes for pure integer programs already carry numerical difficulties, mixed-integer cutting plane techniques have rather poor performance. Further, some previous assumptions for the sensitivity analysis phase, which are valid in pure integer scalarizing programs, do not hold in the mixed-integer case. Hence, we decided to use branch-and-bound techniques to solve the Tchebycheff scalarizing programs and a new challenge emerged to profit from computations previously performed within a branch-and-bound context. Thus, postoptimality techniques were developed to adjust automatically the reference point and to profit from the previous branch-and-bound tree to proceed to next computations. Whenever further branching is required, an attempt is made to first reduce the tree by operations of simplification – cutting deeper branches rather than simple pruning – to avoid an evergrowing tree. This procedure also applies to multiobjective pure integer linear programs.

The remainder of this paper is organized as follows: In Section 2 the problem is defined and some preliminary considerations are made. Section 3 describes the interactive algorithm and how the branch-and-bound tree is used to proceed to next computations. An illustrative example is presented in Section 4. Section 5 is devoted to the implementation and computational results. The paper closes with some concluding remarks in Section 6.

2. Problem definition and preliminary considerations

The MOMILP problem can be formulated as follows:

(MOMILP)

$$\begin{aligned} \max\quad & z_1 = c^1x \\ \dots \\ \max\quad & z_k = c^kx \\ \text{s.t}\quad & Ax = b, \\ & x \geqslant 0, \\ & x_i \text{ integer}, \quad i \in I \subseteq \{1,\dots,n\}, \end{aligned}$$

where k is the number of objective functions (criteria), n is the number of variables, A is a $m \times n$ matrix, b is a column m-vector and c^i, "$i = 1,\dots,k$", are row n-vectors.

Let S denote the set of all feasible solutions, that is $S=\{x\colon\ Ax=b,\ x \geqslant 0,\ x_{\mathrm{i}}$ integer, $i \in I \subseteq \{1,\dots,n\}\}$ and Z the set of the criterion points $z \in \mathbb{R}^k$ images of $x \in S$. It is assumed that S is closed and bounded. $\tilde{x} \in S$ is an *efficient* solution iff there does not exist another $x \in S$ such that $c^ix \geqslant c^i\tilde{x}$ for all i and $c^jx > c^j\tilde{x}$ for at least one j. $\hat{x} \in S$ is said to be *weakly efficient* iff there does not exist another $x \in S$ such that $c^ix > c^i\hat{x}$ for all i. The criterion points corresponding to (weakly) efficient solutions are called (weakly) *nondominated* points/solutions.

Let z^+ denote a criterion reference point that may represent aspiration levels desired by the DM for the objective functions. Without loss of generality, we will assume that z^+ satisfies $z_i^+ \geqslant z_i'$, "$i= 1,\dots,k$", $\forall z' \in Z$. This can be assured by considering $z^+ \geqslant z^*$ with z^* being the ideal criterion point.

The following program determines a solution x whose criterion vector $z \in Z$ is the closest to the reference point z^+ according to the Tchebycheff metric:

(P_1, z^+)

$$\begin{aligned} \min\quad & \alpha \\ \text{s.t.}\quad & c^ix + \alpha \geqslant z_i^+, \qquad i = 1,\dots,k, \\ & x \in S, \quad \alpha \geqslant 0. \end{aligned}$$

Although (P_1, z^+) may have optimal solutions that are weakly efficient for the (MOMILP) problem, among the alternative optima there is at least one efficient solution. Moreover, there always exist reference points $z^+ \geqslant z^*$ that yield a particular efficient solution (note that a nondominated criterion point $\tilde{z}$ is reached if a reference point z^+ such that $z_i^+ = \tilde{z}_i + \delta$, "$i=1,\dots,k$" with $\delta \geqslant \max_{i=1,\dots,k}(z_i^* - \tilde{z}_i)$ is considered in (P_1, z^+)).

An augmented Tchebycheff program, (P_2, z^+), which is obtained by replacing the objective function of (P_1, z^+) by $\alpha - \rho\sum_{i=1}^k c^ix$ with ρ small positive, always returns efficient solutions.

(P_2, z^+)

$$\begin{aligned} \min\quad & \alpha - \rho\sum_{i=1}^{k} c^ix \\ \text{s.t.}\quad & c^ix + \alpha \geqslant z_i^+, \qquad i = 1,\dots,k, \\ & x \in S, \quad \alpha \geqslant 0. \end{aligned}$$

Concerning multiobjective linear programming problems with all-continuous or all-integer variables (MOLP or MOILP), there always exist values for ρ small enough such that all the efficient solutions are reachable. In MOMILP (like nonlinear feasible regions), even considering ρ very small there may be small portions of the efficient set (close to weakly efficient solution(s)) that (P_2, z^+) is unable to compute. However, this is not a practical disadvantage because, for ρ very small, those solutions are so close to the weakly efficient ones that the DM would not discriminate them. It should also be stressed that, even in MOLP or MOILP, the existence of such ρ is mainly of theoretical interest because it is not known a priori.

Tchebycheff metric-based or more general *achievement* scalarizing functions (Wierzbicki, 1980) have the advantage over weighted-sums of being able to reach not only supported efficient solutions but also unsupported efficient solutions (solutions that do not belong to the frontier of the convex hull of the feasible region). For that reason, this type of computing process has been widely used by other authors for dealing with multiobjective problems with nonconvex feasible sets.

While many authors have considered constant reference point (usually the ideal criterion point) varying the weights (e.g. Steuer and Choo, 1983), we have opted for augmented non-weighted Tchebycheff programs parameterized on the reference point. Using this computing process, we can obtain efficient solutions that improve a specific objective function in relation to a previous efficient solution by increasing the respective component of the reference point leaving the other components unchanged. A related proposition is proved in Alves and Clímaco (1999) for MOILP being also valid for MOMILP. This change in the reference point leads to a parametric right-hand side scalarizing program. We developed a postoptimality technique that identifies ranges for the reference points that lead to the same efficient solution and uses the branch-and-bound tree that solved the previous scalarizing program as a starting point to compute the following efficient solutions. This enables to save time in computation phases promoting directional searches.

3. The proposed interactive method

Algorithm:

1. Ask the DM to specify a reference point. At the first interaction the ideal criterion point of the MOMILP problem (or its linear relaxation) is proposed by default. Let the reference point be $z^+ = (z_1^+, \dots, z_k^+)$. (If necessary, z^+ is adjusted in order to satisfy $z^+ \geqslant z^*$ by adding a constant amount to all the components of z^+ – just for technical reasons). Solve the Tchebycheff scalarizing program (P_2, z^+) using branch-and-bound to obtain an efficient solution.
2. If the DM is satisfied, stop; otherwise, if he/she wants to perform a global search by specifying other reference points, go to 1, else go to 3.
3. Ask the DM to choose an objective function he/she wants to improve in relation to the previous efficient solution. Let z_j be the objective function specified by the DM. A directional search is carried out by considering reference points of the type $z^+ + \theta^j = (z_1^+, \dots, z_j^+ + \theta_j, \dots, z_k^+)$ in (P_2, $z^+ + \theta^j$) to produce efficient solutions that improve z_j. When the DM wishes to stop the search through this direction, return to 2.

The above algorithm is just a proposal protocol to interact with the DM (like in Alves and Clímaco, 1999). The core of the method is the way the directional search of step 3 is performed. It consists in optimizing successive scalarizing programs (P_2, $z^+ + \theta^j$) where one scalarizing program only differs from the previous one in the right-hand side of the jth constraint. Postoptimality techniques have been developed to perform this task. This is an iterative process with two main phases:

(i) *sensitivity analysis*,

(ii) *updating the branch-and-bound tree*.

The sensitivity analysis (i) returns a parameter value, $\theta_j^{\max}$, that leaves unchanged the structure of the branch-and-bound tree. It means that for $0 \leqslant \theta_j \leqslant \theta_j^{\max}$ either the reference points lead to the same efficient solution or they lead to different efficient solutions that are easily computed. The branch-and-bound tree is then updated (ii) for θ_j slightly over $\theta_j^{\max}$. The process automatically returns to (i) if the same efficient solution is obtained in (ii).

Let us assume that the Tchebycheff program (P_2, z^+) was solved with $z^+ = (z_1^+, \dots, z_k^+)$ yielding an efficient solution x^0 whose nondominated criterion point is z^0. Now, let us consider that the DM chooses the objective function z_j to be improved in relation to z^0. The next efficient solutions will be obtained by solving the parametric mixed-integer Tchebycheff program where the jth component of the reference point is increased:

(P_2, z^+, θ_j)

$$f(\theta_j) = \min \left\{ \alpha - \rho \sum_{i=1}^{k} c^i x \right\}$$

$$\begin{aligned} \text{s.t.} \quad & c^i x + \alpha - s_i = z_i^+, \quad i = 1, \dots, k, \quad i \neq j, \\ & c^j x + \alpha - s_j = z_j^+ + \theta_j, \\ & x \in S, \\ & \alpha \geqslant 0, \quad s_i \geqslant 0, \quad i = 1, \dots, k. \end{aligned}$$

The s_i, "$i = 1, \ldots, k$" are the surplus variables of the first k constraints.

The parameter is $\theta_j \geqslant 0$. Concerning the first reference point, $(P_2, z^+) \equiv (P_2, z^+, 0)$.

Each node of the optimal branch-and-bound tree for (P_2, z^+) is associated with a linear sub-problem of (P_2, z^+). Regarding the parametric Tchebycheff program, (P_2, z^+, θ_j), the linear sub-problem associated with a node Q^p can be formulated as follows:

$(Q^p_{\theta_j})$

$$
\begin{aligned}
f^p(\theta_j) = {} & \min \left\{ \alpha - \rho \sum_{i=1}^{k} c^i x \right\} \\
\text{s.t.} \quad & c^i x + \alpha - s_i = z_i^+, \quad i = 1, \ldots, k, \quad i \neq j, \\
& c^j x + \alpha - s_j = z_j^+ + \theta_j, \\
& x \in S^p, \\
& \alpha \geqslant 0, \qquad s_i \geqslant 0, \qquad i = 1, \ldots, k,
\end{aligned}
$$

where $S^p = \{x : Ax = b, x \geqslant 0, L_i^p \leqslant x_i \leqslant U_i^p, i \in I\}$ (some L_i^p may be zero and some U_i^p infinite).

In what follows let *integer solution* denote a solution of a sub-problem that has integer values for all the integer-restricted variables in the multiobjective problem.

Let us analyze the effects of (P_2, z^+, θ_j) on the current branch-and-bound tree (where $\theta_j = 0$).

3.1. Infeasible sub-problems

When a sub-problem $(Q^p_{\theta_j})$ is infeasible for $\theta_j = 0$, it will also be infeasible for all $\theta_j > 0$. This proposition holds because, for $\theta_j^2 > \theta_j^1$, the feasible set of $(Q^p_{\theta_j^1})$ contains the feasible set of $(Q^p_{\theta_j^2})$. Hence, all the infeasible nodes of the current tree may be discarded from further consideration.

3.2. The behaviour of each node of the tree

The optimal objective value of a minimizing parametric right-hand side linear program is a piecewise linear convex function of the parameter. This is a well-known result for linear programming with particularities for $(Q^p_{\theta_j})$: besides being convex, the piecewise linear function is nondecreasing and the slope of the function in the last interval of θ_j is 1 (see e.g. Fig. 1).

Let $\pi_i \geqslant 0$, "$i = 1, \ldots, k$" be the dual variables associated with the first k constraints of $(Q^p_{\theta_j})$. The function $f^p(\theta_j)$ must be nondecreasing because it is convex and $0 \leqslant \pi_j \leqslant 1$ (nonnegative slopes). As θ_j grows through positive values, $(Q^p_{\theta_j})$ returns solutions with greater values of $z_j = c^j x$ until the solution that optimizes z_j in S^p is reached. If θ_j grows more, $(Q^p_{\theta_j})$ will yield the same solution, say $\hat{x}$, only varying the values of the variables s_i and α. There exists a specific value of θ_j above which the Tchebycheff distance between the reference point and the criterion point image of $\hat{x}$ is exclusively given by the jth component, i.e, $s_j = 0$ and $s_i > 0$, $i \in \{1, \ldots, k\} \setminus \{j\}$. Hence, for θ_j larger than that specific value, $\pi_i = 0$, $i \in 1, \ldots, k \setminus \{j\}$ and $\pi_j = 1$.

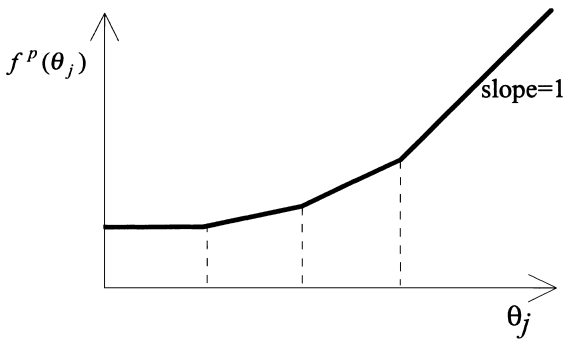

Fig. 1. Example of the behaviour of the optimal objective value of $Q^p_{\theta_j}$.

3.3. Sensitivity analysis

Consider that the efficient solution x^0 that optimizes (P_2, z^+) was produced by the node (subproblem) Q^0 of the branch-and-bound tree. The purpose of the *sensitivity analysis* is to provide a range of values $[0, \theta_j^{\max}]$ for the parameter θ_j such that the optimal solutions of (P_2, z^+, θ_j) will still be given by the node Q^0. Notice that the $\theta_j^{\max}$ returned by this procedure may be lower than the true maximum value.

Two different situations may occur depending on whether s_j is a basic variable in Q^0 or not.

(A.1) s_j is basic in Q^0. If s_j is basic in Q^0, then for θ_j up to the current value of s_j, say s_j^0, neither the value of $f^0(\theta_j)$ changes (due to $\pi_j^0 = 0$) nor x^0; Q^0 is still the optimal node of (P_2, z^+, θ_j) for $\theta_j \leqslant s_j^0$ and it yields the same efficient solution x^0. Therefore, $\theta_j^{\max} = s_j^0$ and there is no need to explore variations of θ_j under this value.

(A.2) s_j is nonbasic in Q^0. Let $\theta_j \in [0, \theta_j^{0\max}]$ be the optimality positive interval for the current basis of Q^0. We will analyze only the case when variations of θ_j within the same basis still produce integer solutions (otherwise, we consider $\theta_j^{\max} = 0$).

Under this condition, the node Q^0 provides feasible solutions of (P_2, z^+, θ_j) for $\theta_j \leqslant \theta_j^{0\max}$ and $f^0(\theta_j) = f^0(0) + \pi_j^0\theta_j$. The performance of Q^0 must be compared with other potential candidate terminal nodes of the branch-and-bound tree (containing or not an integer solution). Denoting by π_j^p the current optimal value of the *j*th dual variable in the node Q^p, the potential candidate nodes are the terminal nodes Q^p that satisfy $\pi_j^p < \pi_j^0$. Nodes for which $\pi_j^p \geqslant \pi_j^0$ need not to be considered because, for θ_j within the range $0 \leqslant \theta_j \leqslant \theta_j^{0\max}$, they cannot provide solutions of (P_2, z^+, θ_j) better than the one given by Q^0 (see Fig. 2). In fact, each linear segment of $f^p(\theta_j)$ has a slope between π_j^p (dashed line below in Fig. 2) and 1 (dashed line above in Fig. 2). Since $\pi_j^p \geqslant \pi_j^0$, $f^p(\theta_j)$ cannot be less than $f^0(\theta_j)$ for $0 \leqslant \theta_j \leqslant \theta_j^{0\max}$.

For each potential candidate terminal node Q^p, an "intersection" parameter value $\theta_j^{0,p}$ is computed. Using only the information provided by the current optimal basis of each node, $\theta_j^{0,p}$ represents a point where $f^p(\theta_j)$ "intersects" $f^0(\theta_j)$ – "real intersection" (e.g. Fig. 3(a)) or "virtual intersection" (e.g. Fig. 3(b), (c) and (d)):

$$\theta_j^{0,p} = \frac{f^p - f^0}{\pi_j^0 - \pi_j^p}, \quad f^p = f^p(0), \qquad f^0 = f^0(0).$$

Q^0 outperforms Q^{p} at least until $\theta_j^{0,p}$ if $\theta_j^{0,p} < \theta_j^{0\max}$ – see examples in Fig. 3 (a) and (b) – or at least until $\theta_j^{0\max}$ if $\theta_j^{0,p} \geqslant \theta_j^{0\max}$ – see examples in Fig. 3(c) and (d).

Hence, the parameter value that ensures that the next efficient solutions will still be given by node Q^0 is

$$\theta_j^{\max} = \min\left\{\theta_j^{0\max}, \min_p\left\{\theta_j^{0,p}\right\}\right\}.$$

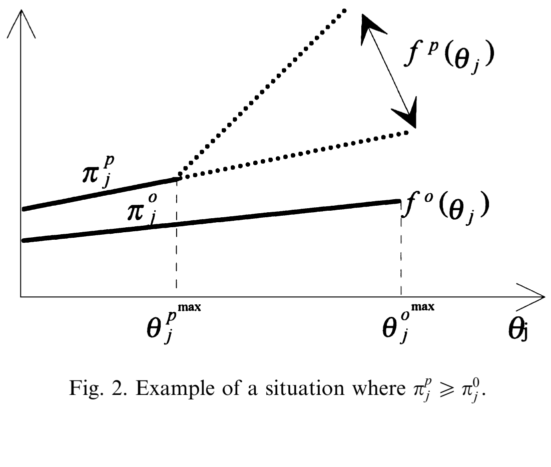

Fig. 2. Example of a situation where $\pi_j^p \geqslant \pi_j^0$.

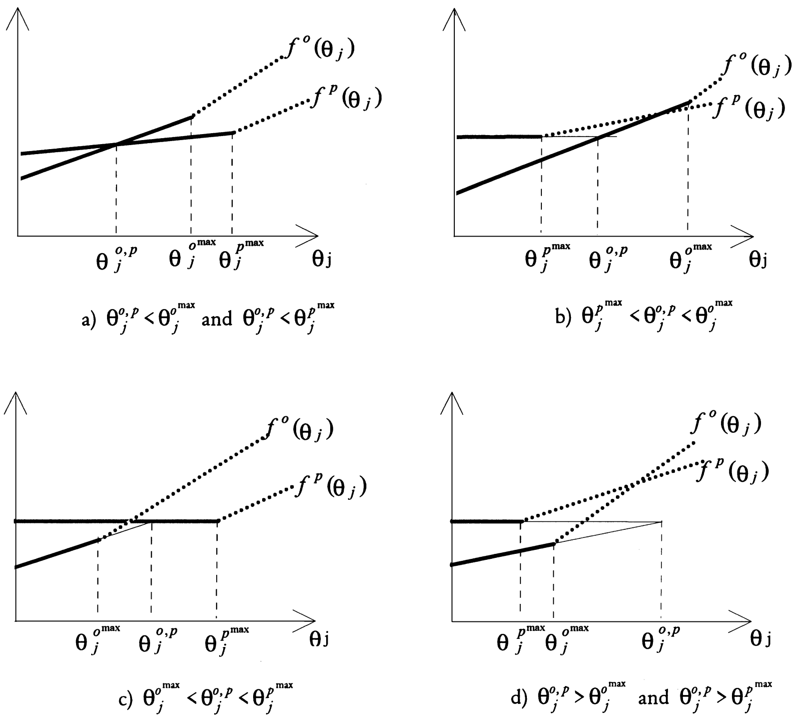

Fig. 3. Examples of intersections.

The efficient solutions that optimize (P_2, z^+, θ_j) with $0 \leqslant \theta_j \leqslant \theta_j^{\max}$ can be obtained in a straightforward way by applying classic linear programming sensitivity analysis to the simplex tableau of Q^0.

3.4. Inactive nodes

Any node Q^v (different from Q^0) for which $\pi_j^v = 1$ may be considered *inactive* while the parametric analysis refers to the jth constraint of the Tchebycheff scalarizing program. In fact, this node (or its future descendants) could not provide the optimal solution of the scalarizing program for larger values of θ_j. Q^v will be activated if the DM changes the direction of search by choosing another objective function to be improved.

3.5. Individual optima

If $\pi_j^0 = 1$ and all the other terminal nodes of the branch-and-bound tree are either *inactive* or their programs are infeasible, then the current efficient solution optimizes the objective function z_j of the MOMILP problem.

3.6. Information about the branch-and-bound tree that is preserved

Some information about the branch-and-bound tree must be preserved between two consecutive iterations of the directional search procedure. This information is used in the sensitivity analysis and to proceed to the next computations:

- The structure of the tree (node indices, links and bounding constraints, excluding infeasible subproblems). For each terminal node Q^p (tree leaf) the following data is saved:
- a basis coding, which allows the procedure to rebuild quickly the simplex tableau; the indices of the basic variables are used to code the basis, just as Steuer (1986) does to determine all the al-

ternative optima of a linear program;

- f^p, the current optimal objective value of the scalarizing sub-problem associated with Q^p;
- π_i^p and $\theta_i^{p^{\max}}$, "$i = 1, \ldots, k$";
- the current state of the node (which tells us whether the node contains an integer solution or not, or it is *inactive*).

3.7. Updating the branch-and-bound tree

As mentioned before, the efficient solutions obtained by optimizing (P_2, z^+, θ_j) within $0 \leqslant \theta_j \leqslant \theta_j^{\max}$ can be obtained in a straightforward way because they also optimize the parametric linear sub-problem associated with the node Q^0. Hence, the structure of the tree does not change for this range of parameter values.

For $\theta_j = \theta_j^{\max}$ three situations can be distinguished:

(B.1) $\theta_j^{\max} = 0$ because Q^0 yields noninteger solutions for larger values of θ_j (this situation occurs within A.2);

(B.2) Q^0 is "intersected" by other terminal node, i.e., $f^0(\theta_j^{\max}) = f^p(\theta_j^{\max})$ for any Q^p (situation (A.2) with $\theta_j^{\max} = \theta_j^{o,p}$);

(B.3) The optimal basis of Q^0 changes for $\theta_j > \theta_j^{\max}$ (situations (A.1) and (A.2) with $\theta_j^{\max} = \theta_j^{0^{\max}}$).

To continue searching for efficient solutions throughout the same direction, we can consider $\hat{\theta}_j = \theta_j^{\max} + \epsilon$, with ϵ small positive. In MOILP problems $\hat{\theta}_j$ is set to the smallest integer larger than $\theta_j^{\max}$ because we can consider only integer reference points without losing efficient solutions (proved in Alves and Clímaco, 1999).

The next reference point is thus $(z_1^+, \ldots, z_j^+ + \hat{\theta}_j, \ldots, z_k^+)$.

Regardless of the situation (B.1, B.2 or B.3) the procedure for updating the branch-and-bound tree begins by updating the simplex tableau of Q^0 and, consequently f^0, π_i^0 and $\theta_i^{o^{\max}}$, "$i = 1, \ldots, k$". The information on the other terminal nodes is also updated. In situations B.2 and B.3 this task is not done just now because it would be wasteful if parts of the tree were cut off later by a *simplification* phase. Afterwards, the information on each terminal node Q^p is updated as follows:

- if $\theta_j^{p^{\max}} \leqslant \hat{\theta}_j$, the basis is changed rebuilding the corresponding simplex tableau (profiting from the coding of the previous basis) and saving the new values of f^p, π_i^p and $\theta_i^{p^{\max}}$, "$i = 1, \ldots, k$". If the new value of π_j^p is 1 and Q^p is not the new optimal node, Q^p is assigned the *inactive* status.
- if $\theta_j^{p^{\max}} > \hat{\theta}_j$ the simplex tableau is not rebuilt, just updating $f^p \leftarrow f^p + \pi_j^p\ \hat{\theta}_j$ and $\theta_j^{p^{\max}} \leftarrow \theta_j^{p^{\max}} - \hat{\theta}_j$; The values of π_i^p "$i = 1, \ldots, k$" remain the same and $\theta_i^{p^{\max}}\, i \neq j$ become *unknown*. These values are not needed for the current direction of search and their computation would require more information about the simplex tableau that has not been explicitly saved. The *unknown* values are computed whenever the DM changes the direction of search.

Let us now analyze how to update the tree for $\theta_j = \hat{\theta}_j$ in each situation from (B.1) to (B.3).

(B.1) The solution of Q^0 is no longer integer:

After updating the information on all the terminal nodes, the branching process starts by splitting Q^0. The branch-and-bound proceeds as usual until the optimum of the scalarizing program is reached.

(B.2) Q^0 was "intersected" by other terminal node:

Let Q^p be the node that "intersected" Q^0 for $\theta_j^{\max}$. Notice that if there are other nodes that "intersect" Q^0 within the range $[\theta_j^{\max}, \hat{\theta}_j]$, they all must be taken into account.

According to the statements (A.2), the optimal basis of Q^0 remains unchanged providing a feasible solution of (P_2, z^+, θ_j). After updating the information on Q^p for $\theta_j = \hat{\theta}_j$, it is compared again with Q^0: if $f^0 \leqslant f^p$ (situation of "virtual intersection" like in Fig. 3(b)) then Q^0 still provides the optimal solution of the scalarizing program; if $f^0 > f^p$ then the solution of Q^p becomes the new optimal solution if it is integer; otherwise, Q^p is the best candidate node and should be branched.

If Q^p needs to be branched, a *low-level simplification* is first attempted. The procedure begins by examining the branching constraint that links Q^p to its parent. If this constraint is no longer active (became redundant as the result of changes on the basis that have occurred since the node was created), then a simplification is made. This is called a *low-level simplification* because it only regards the lowest link to Q^p, i.e., the link between Q^p and its direct ancestor. Since no historical information is kept, this is the only variable branching constraint that was surely active in Q^p when the node was created.

The simplification consists in the following: suppose that $x_i \leqslant K_i$ is the nonactive branching constraint that links Q^p to its parent (Fig. 4). Q^p's parent is removed, as well as its descendants from the other side, and Q^p becomes a direct descendant of its previous grandparent (Fig. 4). Notice that Q^p will possibly be branched on the variable x_i. Without this simplification this would lead to two consecutive branching constraints on the same variable!

Once the simplification is made, the information on the remaining terminal nodes is updated and the branch-and-bound proceeds as usual until the optimum of the scalarizing program is reached.

(B.3) The basis of Q^0 changes:

If the updated optimal solution of Q^0 is integer, it still optimizes the scalarizing program for $\hat{\theta}_j$. Thus, the new efficient solution is found. Otherwise, further branching is required, namely Q^0, the best candidate node, must be split. However, a simplification is first attempted in order to prevent the tree from being too large. The simplification now refers to *all* the variable bounding constraints that were active in the previous base of Q^0 and become redundant with this change of the basis. Therefore, *low* (as in situation B.2) and/or *high-level simplifications* may occur. The steps, which rule the general simplification process, are the following (illustrated in Fig. 5):

Assume that Q^0 is the optimal node of the previous scalarizing program and is now the best candidate node (with a noninteger solution).

For each branching constraint that was active in the previous optimal basis of Q^0 and is now redundant, *do*

Suppose that the branching constraint under consideration links Q^w and Q^v (Q^v being the parent of Q^w). In a *low-level simplifications*, $Q^w \equiv Q^0$.

1. Cut off the branch $Q^v - Q^w$.
2. Link Q^w directly to the parent of Q^v, say Q^u, by the branching constraint that previously linked Q^v to Q^u; if Q^v was the root then Q^w becomes the root.
3. Remove Q^v, its upper link and its descendants from the side opposite to Q^w.
4. Concerning Q^w and its descendants, leave just the intermediate nodes needed to get Q^0 but assure that they remain forked: considering that there are q intermediate nodes between Q^w and Q^0, that is $Q^w \equiv Q_0, Q_1, \ldots, Q_q, Q_{q+1} \equiv Q^0$; replace all the descendants of Q_i "$i = 0, \ldots, q$" from the side opposite to Q_{i+1} with a new single node (a temporary terminal node).

Now, neither Q^w nor another descendant of Q^w includes in its linear program the bounding constraint that linked Q^w to Q^v.

If the simplified tree has consecutive branching constraints on the same variable, *then* a further simplification should be made (following the steps above) in order to discard the constraint that is redundant for Q^0.

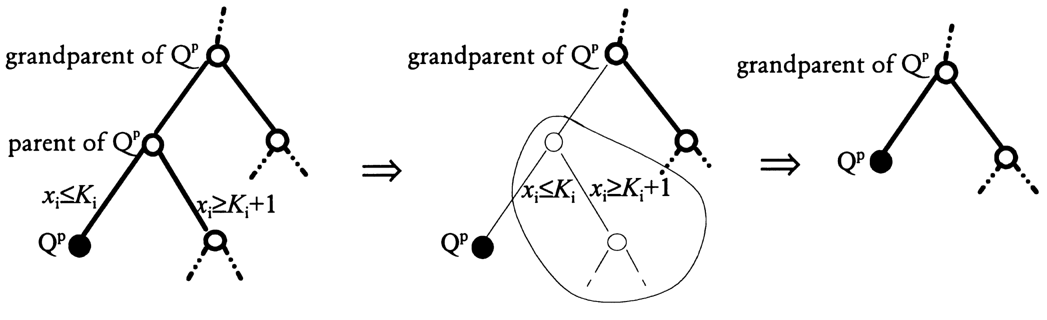

Fig. 4. The simplification process and the resulting tree.

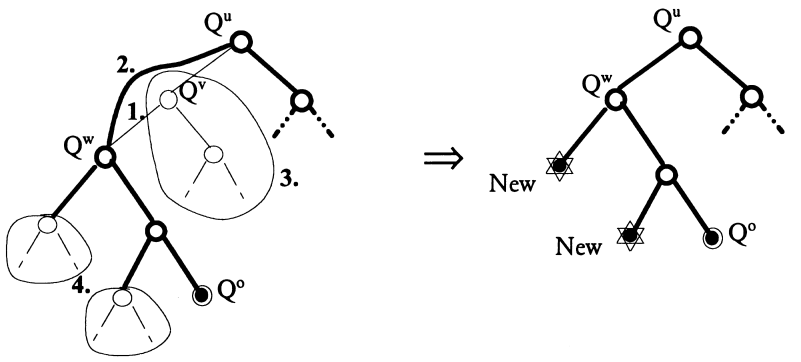

Fig. 5. Illustration of the general simplification process.

This type of additional simplification is illustrated in Fig. 6. Note that the new right branch of Q^w in Fig. 6 includes the sub-problems of the old right branch of Q^u.

To sum (B.3) up, if the updated solution of Q^0 for $\hat{\theta}_j$ is noninteger, the first stage consists in simplifying the tree (when possible). Then, the information on the remaining terminal nodes is updated and the information is built on the new terminal nodes created by the simplification process; the last stage consists in the expansion of the tree, which starts by branching Q^0. The branch-and-bound proceeds as usual until the optimum of the new scalarizing program is reached.

It should be stressed that if both (B.2) and (B.3) occur within the range $[\theta_j^{\max}, \hat{\theta}_j]$, Q^0 must be compared again with the updated terminal nodes. This means that propositions stated in (B.2) or (B.3) like "*if the solution of Q is integer, then it is the new optimal solution of the scalarizing program*", which are valid for individual situations (B.2) or (B.3), must be checked whenever both of them occur "simultaneously".

4. An illustrative example

Consider the following MOMILP problem:

$$\begin{aligned}
\max \quad & z_1 = 3x_1 + x_2 + 2x_3 + x_4 \\
\max \quad & z_2 = x_1 - x_2 + 2x_3 + 4x_4 \\
\max \quad & z_3 = -x_1 + 5x_2 + x_3 + 2x_4 \\
\text{s.t.} \quad & 2x_1 + x_2 + 4x_3 + 3x_4 \leqslant 56, \\
& 3x_1 + 4x_2 + x_3 + 2x_4 \leqslant 55, \\
& x_j \geqslant 0, \qquad j = 1, \ldots, 4, \\
& x_1 \text{ and } x_2 \text{ integer.}
\end{aligned}$$

Let the initial reference point be $z^+ = (108, 80, 75)$. The scalarizing program (P_2, z^+), considering

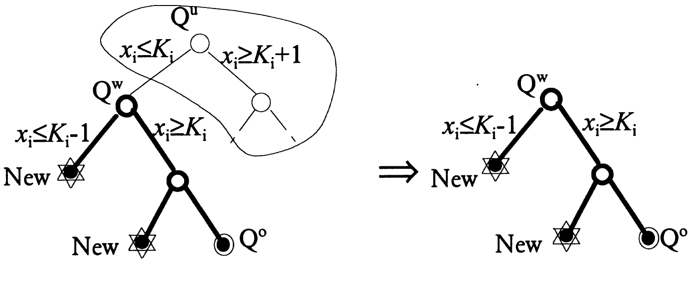

Fig. 6. An additional simplification.

$\rho = 0.001$, was solved using branch-and-bound yielding the efficient solution $x = (10,\ 4,\ 8,\ 0)$ whose criterion point is $z = (50, 22, 18)$.

Suppose that the DM wants to perform a directional search improving the objective function z_2. The second component of the reference point will be increased and $(P_2,\ z^+)$ gives place to the parametric Tchebycheff program $(P_2,\ z^+,\ \theta_2)$ with θ_2 a nonnegative parameter. The optimal solution of $(P_2,\ z^+)$ was given by the branch-and-bound tree shown in Fig. 7. (In order to improve clearness, we only show the data needed for the current directional search.)

Whenever the reference point is changed, θ_2 is reset to 0 for the next iteration.

In this example we will consider $0 \leqslant \epsilon \leqslant 0.1$ in $\hat{\theta}_j = \theta_j^{\max} + \epsilon$. For simplification reasons, ϵ is such that $\hat{\theta}_j$ has only one decimal digit, i.e., $\hat{\theta}_j = (\theta_j^{\max}$ truncated with precision $1) + 0.1$.

First iteration

Sensitivity analysis. Q^4 (in Fig. 7) is the current optimal node and the solution remains integer for variations of θ_2 within the same basis, i.e, for $\theta_2 \leqslant \theta_2^{4^{\max}} = 2.4$. The performance of each other terminal node Q^p such that $\pi_2^p < \pi_2^4$ is compared to Q^4 by computing the corresponding "intersection" parameter value: $\theta_2^{4,3} = 31.386$, $\theta_2^{4,5} = 10.25$; $\theta_2^{\max} = \min\{\theta_2^{4^{\max}},\ \theta_2^{4,3},\ \theta_2^{4,5}\} = 2.4$.

Hence, Q^4 still provides optimal solutions of the scalarizing programs for $\theta_2 \leqslant 2.4$. Since the current basis of Q^4 remains feasible for $\theta_2 \leqslant 2.4$, and the variation of θ_2 does not destroy the integer-feasibility of the solutions, the computation of nondominated solutions closest to reference points from (108, 80, 75) to (108, 82.4, 75) is straightforward.

Let us now consider $\hat{\theta}_2 = 2.4 + \epsilon^{(1)} = 2.5$ to continue the search. The next reference point is (108, 82.5, 75).

Updating the tree for $z^+ = (108,\ 82.5,\ 75)$. Situation (B.3) – The basis of Q^4 changes leading to a non-integer solution. The branching constraint $x_2 \leqslant 4$ becomes redundant. A *low-level simplification* is performed (Fig. 8) and the information on the other terminal nodes is then updated.

Starting the branch-and-bound with the simplified tree and continuing until the optimum of the scalarizing program is reached (Fig. 9), a new efficient solution is obtained: $x = (10,\ 4,\ 7.4,\ 0.8)$, $z = (49.6,\ 24,\ 19)$. As it was expected, z_2 is improved.

Suppose that the DM wishes to continue the search along the same direction (improving z_2) in this and the next interactions. Hence, the procedure returns to the sensitivity analysis phase after updating the tree in a computing phase.

Second iteration

Sensitivity analysis. Q^9 (in Fig. 9) is the new current optimal node and the solution remains integer for every positive change of θ_2. $\theta_2^{\max} = \min\{\theta_2^{9^{\max}},\ \theta_2^{9,6},\ \theta_2^{9,8},\ \theta_2^{9,3}\} = \min\{\infty, 0.929, *, *\} = 0.929$. The entries * need not be computed because $f^* > f^6$ and $\pi_2^* > \pi_2^6$ both in Q^8 and Q^3 which lead to intersection values larger than 0.929.

Thus, for $\theta_2 \leqslant 0.929$, Q^9 is still the optimal node for the scalarizing program.

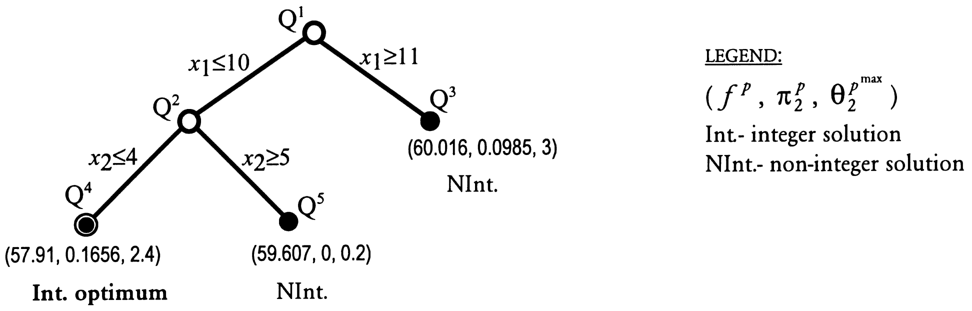

Fig. 7. The final tree for the reference point $z^+ = (108,\ 80,\ 75)$.

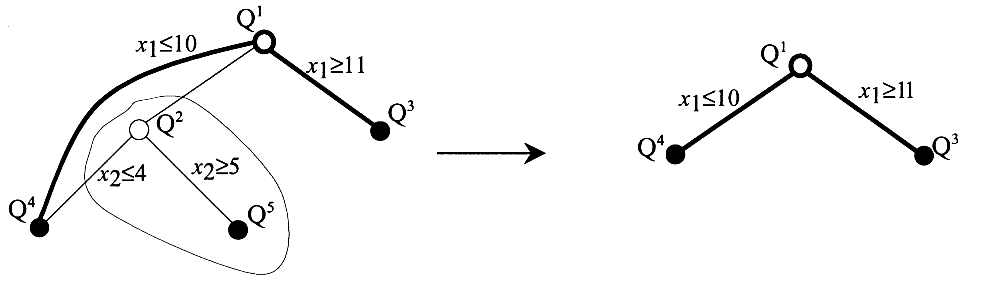

Fig. 8. Simplification yielding the starting-tree for $z^+ = (108, 82.5, 75)$.

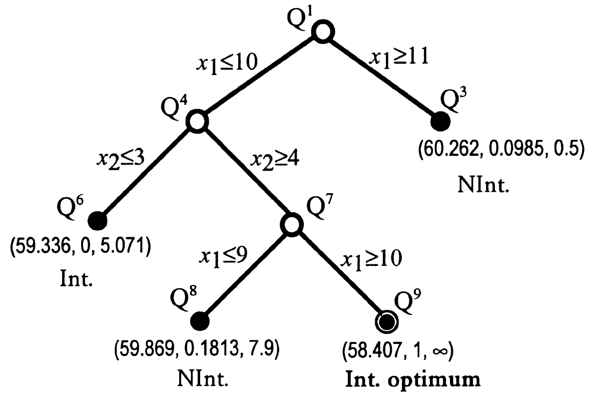

Fig. 9. The final tree for $z^+ = (108, 82.5, 75)$.

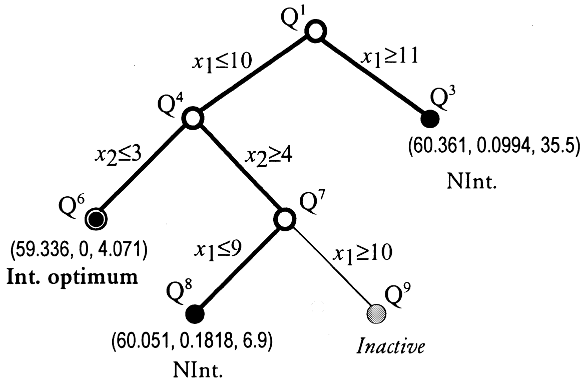

Fig. 10. The final tree for $z^+ = (108, 83.5, 75)$.

Let $\hat{\theta}_2$ be 0.929 + $\epsilon^{(2)} = 1.0$ to continue the search.

Updating the tree for $z^+ = (108, 83.5, 75)$. Situation (B.2) – "Q^6 intersected Q^9". The basis of Q^6 does not change for $\hat{\theta}_2$ and the corresponding solution is integer. The tree structure is not affected, only the information on terminal nodes must be updated (Fig. 10). The new efficient solution is given by Q^6: $x = (10, 3, 6.857, 1.857)$, $z = (48.571, 28.143, 15.571)$.

Third iteration

Sensitivity analysis. Q^6 (in Fig. 10) is the current optimal node with $\pi_2^6 = 0$. The variable s_2 (surplus variable of the constraint that represents z_2 in the scalarizing program) is basic – situation (A.1). Hence, the previous nondominated point remains the closest to reference points (108, $83.5 + \theta_2$, 75) with $0 < \theta_2 \leqslant \theta_2^{6^{\max}} = s_2 = 4.071$. Therefore, let us consider $\hat{\theta}_2 = 4.071 + \epsilon^{(3)} = 4.1$ to search for other nondominated solutions.

Updating the tree for $z^+ = (108, 87.6, 75)$. Situation (B.3) – The basis of Q^6 changes. Since Q^6 is the best candidate node and its new solution is integer, Q^6 remains the optimal node; $x = (10, 3, 6.850, 1.867)$, $z = (48.567, 28.167, 15.583)$.

In order to illustrate other situations while avoiding repetitions, we will skip several iterations assuming that the search has been, and will continue, through the same direction. Hence, consider the reference point $z^+ = (108, 120.5, 75)$ corresponding to the tree shown in Fig. 11. $x = (10, 0, 0.875, 10.833)$, $z = (42.583, 55.083, 12.542)$. Q^{22} is the current optimal node.

Sensitivity analysis. Q^{22} (in Fig. 11) is the current optimal node and the solution remains integer for variations of θ_2 within the current basis, i.e., for $\theta_2 \leqslant \theta_2^{22^{\max}} = 3.5$. In addition, no node satisfies the basic requirements for checking "intersection". So, for $\theta_2 \leqslant 3.5$, Q^{22} is still the optimal node of the scalarizing program. The computation of

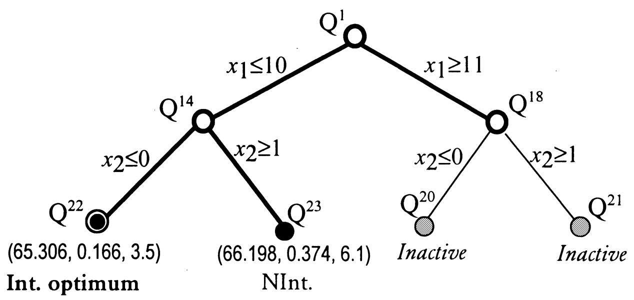

Fig. 11. The final tree for $z^+ = (108, 120.5, 75)$.

nondominated solutions closest to reference points from (108, 120.5, 75) to (50,124,18) is straightforward.

Let $\hat{\theta}_2 = 3.5 + \epsilon^{(6)} = 3.6$.

Updating the tree for $z^+ = (108, 124.1, 75)$. Situation (B.3) – The optimal basis of Q^{22} changes and the bounding constraints $x_2 \leqslant 0$ and $x_1 \leqslant 10$ become redundant for Q^{22}. Although the nonnegativity constraint for x_2 is brought active, so that $x_2 = 0$, the solution is noninteger because $x_1 = 9.975$. Q^{22} is thus the best candidate node that should be branched. Before branching, two simplifications of the tree are made. We first pick the bounding constraint $x_2 \leqslant 0$ and, afterwards, $x_1 \leqslant 10$ leading to two consecutive *low-level simplifications* (Fig. 12).

Starting the branch-and-bound with the simplified tree – just the node Q^{22} – and continuing until the optimum is reached (Fig. 13), a new efficient solution is obtained, $x = (10, 0, 0, 12)$, whose criterion point is $z = (42, 58, 14)$.

5. Implementation and computational results

We have implemented the proposed interactive multiobjective approach using the DELPHI developer for Windows 95 (in a PC – Pentium,

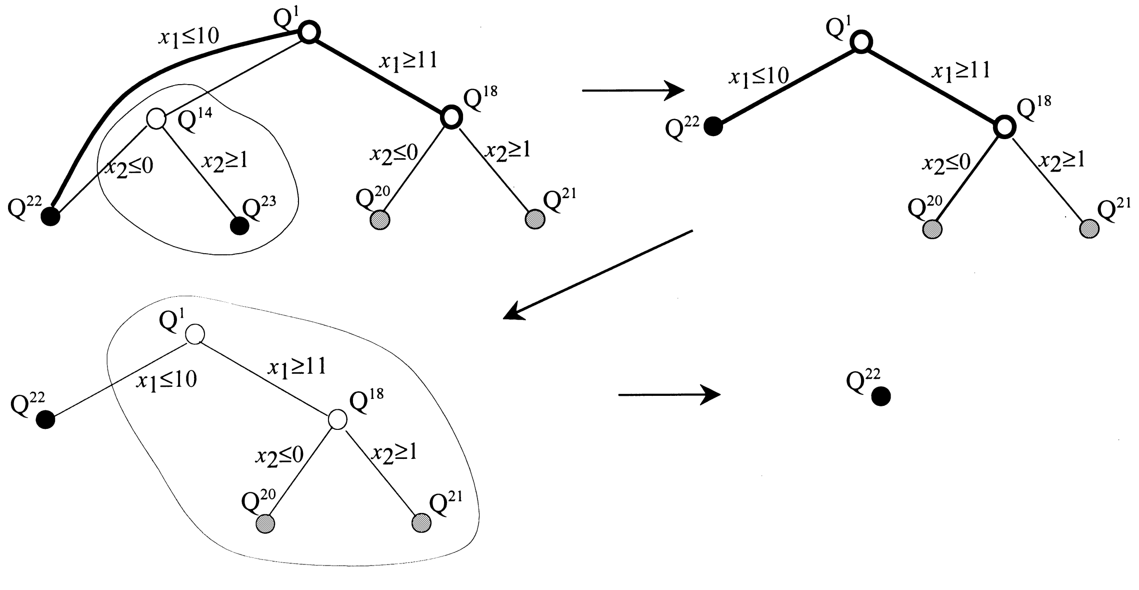

Fig. 12. Simplification yielding the starting-tree for $z^+ = (108, 124.1, 75)$.

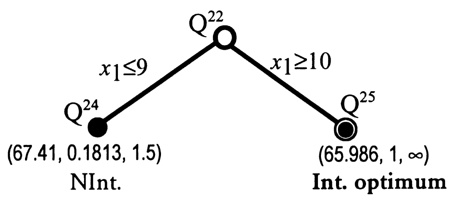

Fig. 13. The final tree for $z^+ = (108, 124.1, 75)$.

166 MHz). The main modules are: a spreadsheet-based problem editor, graphical procedures to interact with the DM and computing routines that include problem pre-processing.

The dialogue with the DM is mainly based on asking him/her to specify a new reference point (infeasible or not) or an objective function he/she wants to improve with respect to the previous efficient solution. However, the DM may start with a more global search by demanding the computation of the pay-off table and some dispersed efficient solutions. All the efficient solutions the DM considers interesting are kept in memory. Numerical and different graphical displays (with several sorter options) are available to show the values of these solutions. Also, a graphical representation of the reference point space for bicriteria and tricriteria (pure) integer programs is presented to the DM. In this graph, regions of reference points that lead to the same efficient solution are iteratively appended (an example is shown in Alves and Clímaco, 1999). All this information that is provided for the DM is of special importance to guide him/her throughout the search process of efficient solutions.

The multiobjective approach proposed herein was tested for several randomly generated problems with 2 or 3 objective functions: all-binary variable problems (knapsack, multidimensional knapsack, set covering and set packing), all-integer problems (integer knapsack and generic ones) and generic mixed-integer problems.

Several tests were carried out for the same problem in order to compare the computational times needed to obtain efficient solutions through a directional search with those required by individual optimization of the Tchebycheff scalarizing programs. These times are summarized in Table 1. Pure integer multiobjective programs often require more than one iteration of *sensitivity analysis* plus *updating the branch-and-bound tree* to escape from the previous efficient solution. The respective column of Table 1 refers to the total time spent on these iterations. Concerning mixed-integer

Table 1
Average times of computing tests[a]

Problem type	n	m (excluding bounds)	#problems	Average time	
				Sensitivity analysis + updating the tree (step 3 of the algorithm)	Solving each scalarizing program from the beginning (*Restart*: step 1 of the algorithm)
0-1 knapsack	100 B	1	6	2′30″	3′32″
	200 B	1	6	2′8″	6′45″
Integer knapsack	100 B	1	6	50″	1′
	200 B	1	6	2′56″	6′1″
0-1 multi-knapsack	50 B	10	6	1′26″	2′5″
	60 B	10	5	7′8″	11′50″
Set covering	100 B	20–30	5	19″	18″
Set packing	100 B	20	6	16″	22″
	200 B	30	6	1′8″	3′41″
Generic all-integer	30 I	15	6	2′48″	4′30″
Generic MOMILP	15 I, 15 B, 30 C	30	6	3″	39″
	20 I, 20 B, 40 C	40	6	3″	1′4″

[a] I: integer; B: Binary; C: Continuous.

Table 2
Some problem results

Directional search					Re(start) Time
Strategy	Reference point	Solution $z = (z_1, z_2, \ldots)$	*It.*	Total time	
(a) • start	(3147, 3022, 3305)	(2836, 2702, 2988)			6′0″
• $\uparrow z_3$	⇒ (″,″, 3310)	(2830, 2700, 2997)	2	1′52″	7′18″
• $\uparrow z_3$	⇒(″,″, 3324)	(2821, 2701, 3000)	6	3′2″	4′35″
(b) • start	(1514, 1372, 1492)	(1307.9, 1137.8, 1257.8)			2′2″
• $\uparrow z_3$	⇒ (″,″, 1492.6)	(1308.0, 1137.4, 1258.0)	1	4″	2′46″
• $\uparrow z_3$	⇒(″,″, 1493.3)	(1279.2, 1137.2, 1258.5)	1	2″	2′2″
• $\uparrow z_3$	⇒(″,″, 1508.6)	(1277.8, 1135.8, 1272.4)	1	17″	1′27″
(c) • start	(893240, 5058000, 1520,3746.7, 609209)	(1160200, 5376000, 4320, 6360, 996827)			0.8″
• $\downarrow z_5$	⇒(″,″,″,″, 434802)	(1162280, 5620000, 4320, 11240, 829228)	5	1.1″	1.5″
• $\downarrow z_5$	⇒(″,″,″,″, 267061)	(1162593, 5620000, 4320, 11240, 829060)	1	0.2″	0.8″

problems, we report here the average times spent on iterations that involve changes on the tree structure, i.e. "more time consuming" iterations.

Table 2 shows more detailed results for three problems: (a) 0–1 knapsack problem with 100 variables and 3 criteria; (b) generic mixed-integer problem with 100 variables (50 integer and 50 continuous), 50 constraints and 3 criteria to be maximized; (c) location-routing problem with real data (Coutinho-Rodrigues et al., 1997) – 69 variables (3 binary and 66 continuous), 57 constraints and 5 criteria to be minimized. The number of iterations (*It.*) of *sensitivity analysis* plus *updating the tree* needed to obtain each new solution is also shown in Table 2.

The times presented in the last two columns of Table 1 or Table 2 should be analyzed in relative terms rather than in absolute terms. In fact, joining a commercial MIP solver to our procedures would undoubtedly improve the times.

Our experimentation is still preliminary since more problems should be considered namely sets of problems already tested by other authors. But, unfortunately, there is a lack of multiobjective (mixed) integer problems in the literature.

6. Conclusions

In this paper we described an interactive method for MOMILP problems that combines Tchebycheff scalarizing programs with branch-and-bound techniques. This work followed our previous research on the development of an MOILP method that uses cutting planes to solve pure integer scalarizing programs. As expected, the branch-and-bound method is more effective than the cutting plane method because it is more reliable (there are, however, a few situations where the cutting plane method is faster than the branch-and-bound method).

The main feature of the branch-and-bound multiobjective method we propose consists in using the previous branch-and-bound tree to perform a sensitivity analysis phase and to proceed to next computations of efficient solutions. Rules to simplify the tree have been developed in order to avoid an evergrowing tree. This leads to storage space for the tree information and to save time in handling it.

Computational experiments have shown that this method succeeds in performing directional and local searches. In almost all the tested problems, the times of computing phases were significantly reduced by the use of the simplification/branching process. This is specially relevant in some multiobjective problems, namely in mixed-integer ones and higher-sized problems. MOILP problems, rather than MOMILP, often require changing the reference point more than once to escape from the previous efficient solution. However, the total time spent on those iterations of sensitivity analysis plus updating the branch-and-bound tree is quite good in comparison to the single optimization of the final scalarizing program of that iterative process. Moreover, as the reference point is automatically adjusted, the procedure assures that each new efficient solution obtained during the directional search is close to, but different from, the previous efficient solution and improves the objective function selected by the DM. Therefore, the DM need not attempt reference points that could lead to the previous efficient solution.

References

Alves, M.J., Clímaco, J., 1999. Using cutting planes in an interactive reference point approach for multiobjective integer linear programming problems. European Journal of Operational Research 117 (3), 565–577.

Aksoy, Y., 1990. An interactive branch-and-bound algorithm for bicriterion nonconvex/mixed integer programming. Naval Research Logistics 37, 403–417.

Bitran, G.R., 1977. Linear multiple objective programs with zero-one variables. Mathematical Programming 13, 121–139.

Bowman, V.J., 1976. On the relationship of the Tchebycheff norm and the efficient frontier of multiple-criteria objectives, In: H. Thiriez, S. Zionts, (Eds.), Multiple Criteria Decision Making, Lecture Notes in Economics and Mathematical Systems 130, Springer, Berlin, pp. 76–86.

Coutinho-Rodrigues, J., Current, J., Clímaco, J., Ratick, S., 1997. An interactive spatial decision support system for multiobjective HAZMAT location-routing problems. Transportation Research Board, Record 1602, 101–109.

Durso, A., 1992. An interactive combined branch-and-bound/Tchebycheff algorithm for multiple criteria, In: A. Goicoechea, L. Duckstein, S. Zionts, (Eds.), Multiple Criteria Decision Making, Proceedings of the 9th Conference, Springer, Berlin, pp. 107–122.

Evans, G.W., 1984. An overview of techniques for solving multiobjective mathematical programs. Management Science 30 (30), 1268–1282.

Ferreira, C., Santos, B.S., Captivo, M.E., Clímaco, J., Silva, C.C., 1996. Multiobjective location of unwelcome or central facilities involving environmental aspects: A prototype of a decision support system. Belgian Journal of Operations Research, Statistics and Computer Science 36 (2/3), 159–172.

Huckert, K., Rhode, R., Roglin, O., Weber, R., 1980. On the interactive solution to a multicriteria scheduling problem. Zeitschrift für Operations Research 24, 47–60.

Karaivanova, J.N., Narula, S.C., Vassilev, V., 1993. An interactive procedure for multiple objective integer linear programming problems. European Journal of Operational Research 68, 344–351.

Karaivanova, J., Korhonen, P., Narula, S., Wallenius, J., Vassilev, V., 1995. A reference direction approach to multiple objective integer linear programming. European Journal of Operational Research 81, 176–187.

Karwan, M.H., Zionts, S., Villarreal, B., Ramesh, R., 1985. An improved interactive multicriteria integer programming algorithm, In: Y.Y. Haimes, V. Chankong (Eds.), Decision Making with Multiple Objectives, Lecture Notes in Economics and Mathematical Systems 242. Springer, Berlin, pp. 261–271.

L'Hoir, H., Teghem, J., 1995. Portfolio selection by MOLP using an interactive branch and bound, Foundations of Computing and Decision Sciences 20/3, Institute of Computing Science, Technical University of Poznan, Poland.

Narula, S.C., Vassilev, V., 1994. An interactive algorithm for solving multiple objective integer linear programming problems. European Journal of Operational Research 79, 443–450.

Osleeb, J.P., Ratick, S.J., 1983. A mixed integer and multiple objective programming model to analyze coal handling in New England. European Journal of Operational Research 12, 302–313.

Ramesh, R., Zionts, S., Karwan, M.H., 1986. A class of practical interactive branch and bound algorithms for multicriteria integer programming. European Journal of Operational Research 26, 161–172.

Ritzel, B.J., Eheart, J.W., Ranjithan, S., 1994. Using genetic algorithms to solve a multiple objective groundwater pollution containment problem. Water Resources Research 30/5, 1589–1603.

Solanki, R., 1991. Generating the noninferior set in mixed integer biobjective linear programs: An application to a location problem. Computers and Operations Research 18 (2), 1–16.

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

Steuer, R., 1986. Multiple Criteria Optimization: Theory, computation and application, Wiley, New York.

Teghem, J., Kunsch, P.L., 1986. Interactive methods for multi-objective integer linear programming, In: G. Fandel, M. Grauer, A. Kurzhanski, A.P. Wierzbicki (Eds.), Large-Scale Modelling and Interactive Decision Analysis, Lecture Notes in Economics and Mathematical Systems 273, Springer, Berlin, pp. 75–87.

Trivedi, N.M., 1981. A mixed-integer goal programming model for nursing service budgeting. Operations Research 29 (5), 1019–1034.

Vassilev, V., Narula, S.C., 1993. A reference direction algorithm for solving multiple objective integer linear programming problems. Journal of Operational Research Society 44 (12), 1201–1209.

Villarreal, M., Karwan, H., Zionts, S., 1980. An interactive branch and bound procedure for multicriterion integer linear programming, In: G. Fandel, T. Gal (Eds.), Multiple Criteria Decision Making Theory and Application, Lecture Notes in Economics and Mathematical Systems 177, Springer, Berlin, pp. 448–467.

Weber, C.A., Current, J.R., 1993. A multiobjective approach to vendor selection. European Journal of Operational Research 68, 173–184.

White, D.J., 1990. A bibliography on the applications of mathematical programming multiple-objective methods. Journal of Operational Research Society 41 (8), 669–691.

Wierzbicki, A., 1980. The use of reference objectives in multiobjective optimization, In: G. Fandel, T. Gal, (Eds.), Multiple Criteria Decision Making, Theory and Application, Lecture Notes in Economics and Mathematical Systems 177. Springer, Berlin, pp. 468–486.

Zionts, S., 1977. Integer Programming with multiples objectives. Annals of Discrete Mathematics 1, 551–562.