Theory and Methodology

Lexicographic improvement of the target values in convex goal programming

Rafael Caballero, Lourdes Rey, Francisco Ruiz *

Departamento de Economía Aplicada (Matemáticas), Facultad de Ciencias Económicas y Empresariales, University of Málaga, Campus El Ejido s/n, 29071 Málaga, Spain

Received 8 June 1995; accepted 27 June 1996

Abstract

The aim of this paper is to carry out an exhaustive post optimization analysis in a Convex Goal Programming problem, so as to study the possible existence of satisfying solutions for different levels of the target values. To this end, an interactive algorithm is proposed, which allows us to improve the values of the objective functions, after obtaining a satisfying solution, if such a solution exists, in such a way that a Pareto optimal solution is finally reached, through a successive actualization of such target values. This way, the target values are lexicographically improved, according to the priority order previously given by the decision maker, in an attempt to harmonize the concepts of satisfying and efficient solutions, which have traditionally been in conflict. © 1998 Elsevier Science B.V.

Keywords: Goal programming; Convex programming

1. Introduction

Goal Programming has proved to be, since its birth (Charnes and Cooper, 1961; Ijiri, 1965; Lee, 1972), a very important and powerful Multi Objective technique, specially because of its practical applicability to real problems. It avoids the great inconvenience of having to choose among a large number of efficient solutions, obtained by some traditional techniques of Multiple Objective Programming, through a tacit renounce to optimization by the decision maker, who feels satisfied if certain target values imposed to each objective are achieved. This scheme is now widely used, and very important works, like Ignizio (1976) on it have been published since then.

Although it is possible to argue that the imposition of such values can be difficult in some cases, it seems even more difficult to choose among a large number of efficient solutions obtained through the use of some generating method. These efficient solutions can, however, be used in a first step, because they allow the decision maker to know the structure of the problem, so that he can afterwards choose appropriate target values.

On the other hand, the existence of both the weighted and the lexicographic schemes, let us deal with different preference relations among the objectives.

* Corresponding author.
Nevertheless, many of the criticisms addressed to Goal Programming refer to the final solution obtained (see, for example, Zeleny, 1981). Precisely speaking, they attack its relative ease to generate dominated solutions. In fact, the algorithm bases its behavior and final aim in finding a satisfying solution, that is, a combination in the decision space that satisfies all the goals of the program. This yields, in practice, to ignore the initial objectives posed by the decision maker, which consisted of the improvement or optimization, if possible, of the different objective functions.

As a result of these criticisms, it seems logical to make an attempt to develop algorithms which allow us to obtain satisfying as well as efficient solutions. In this sense, it can be pointed out the work by Hannan (1980), and a general survey of the problem for the linear case in Romero (1991). On the other hand, Caballero et al. (1996) studied the convex general case. However, within the frame of a Goal Programming problem, a search for a particular efficient solution may be senseless if we do not take into account other criteria. Thus, for example, in most of the cases it suffices to optimize one single objective in the satisfying set to obtain a satisfying and efficient solution, but it is just fair to wonder in this case why should a decision maker, who has already expressed his preferences through the specification of certain target values, and probably certain priority levels as well, feel satisfied with this efficient solution, or with any other solution obtained in a similarly arbitrary way.

This question has made us think that it must be much more logical from the point of view of a Goal Programming scheme, to carry out a post-optimization analysis which allows us, once a satisfying solution has been obtained, to study the possibility to improve each objective, according to the priority order previously established by the decision maker. This way, a lexicographic study is carried out, that informs the decision maker up to which value can he improve his objectives, so that satisfying solutions can still be found, or, in other words, which extreme values can take the target values, so that such solutions exist. As a consequence of the application of the interactive algorithm developed, the decision maker can sequentially improve his objectives according to his previously given priority order, and also be informed whether the final solution obtained is efficient or not (and, if it is not, he is also offered the possibility to obtain it).

Thus, along the present paper we defend the convenience to obtain solutions which are satisfying and efficient at the same time, because it seems clear that no rational decision maker would agree with a solution that he knows that is dominated by another one, but we propose an algorithm that seeks a 'lexicographic optimization', in the sense that it allows to improve, if possible, the values of the objectives, according to the priority order given by the decision maker, so that the original target values are sequentially actualized. The application of this algorithm, based upon the STEM method, lets the decision maker, in some way, to 'design' the efficient solution that he wishes to obtain, at the same time that it carries out a post-optimization analysis on the possible levels of the target values.

The possibility to carry out the previously mentioned post-optimization analysis in the case when the original Goal Programming problem does not have any satisfying solution is also contemplated, if the decision maker does not feel satisfied with the final solution given by the Goal Programming algorithm.

This is not, of course, the first attempt to combine Goal Programming with an interactive scheme. In this field, we can find the papers by Dyer (1972), who focuses the problem from the assumed existence of an implicit utility function of the decision maker, Masud and Hwang (1981) who describe an algorithm to obtain efficient solutions via the interactive actualization of the trade-offs, or applications of interactive goal programming to real problems like Lara and Romero (1992). Fichefet (1976) combines Goal Programming with the STEM method for linear problems. Nevertheless, our aim is just to build an algorithm which lets us carry out a post-optimization analysis, so as to improve the target values after the lexicographic Goal Programming scheme has been carried out.

2. Preliminaries

In this section, we briefly describe the algorithm which has been used to solve the convex Goal
Programming problems, which is an adaptation to this case of the lexicographic method used in the linear case.

To this end, we suppose that we consider the general problem:

\[
\begin{align*}
\min & \quad f'(x) = (f_1(x), \ldots, f_p(x)) \\
\text{s.t.} & \quad g'(x) \leq \bar{b},
\end{align*}
\]

where both the objective functions \( f_i \) \( (i = 1, \ldots, p) \) and the ones that determine the constraints \( g_j \) \( (j = 1, \ldots, m) \) are convex in their respective domains.

According to the Goal Programming philosophy, we also suppose that the decision maker expresses his intentions through the achievement of \( p \) goals, that take the following form:

\[
A(x) \leq u_i, \quad (i = 1, \ldots, p),
\]

where \( u_i \) \( (i = 1, \ldots, p) \) are the target values for each objective, and are also given by the decision maker.

Along this paper, it will be assumed that the decision preference structure is discontinuous, that is, there are strict preferences among the objectives, expressed through priority levels. This way, the preference ordering in the objective space is the lexicographic one. Then, in order to use this lexicographic algorithm, the decision maker must have previously stated the corresponding priority levels \( N_1, \ldots, N_p \). If the \( i \)th objective function belongs to the level \( l \), we will write \( i \in N_l \). It is possible that several functions share the same priority level, but this does not necessarily mean that the decision maker considers that they all have exactly the same importance. That is why the decision maker can, if he so wishes, show his relative preferences among them assigning a weight \( \omega_i \) to each one. So as to avoid bias effects towards the functions that take larger absolute values, and also to avoid using achievement functions whose terms are measured in different units, this weights are afterwards normalized. One way to carry out this normalization is dividing each weight by the target value of the corresponding goal, so that we obtain this way:

\[
\mu_i = \frac{\omega_i}{u_i}.
\]

Finally, in order to carry out the lexicographic optimisation, we introduce positive deviation vari-
ables in the model, so that the goals are transformed into:

\[
\begin{align*}
f_i(x) - p_i & \leq u_i, \quad (i = 1, \ldots, p), \\
p_i & \geq 0, \quad (i = 1, \ldots, p).
\end{align*}
\]

It must be observed that the negative deviation variable \( n_i \), traditionally used in the linear models, has not been introduced here, because this would imply considering goals with equality relations, affecting this way the convexity of the model constraints. Anyway, provided that we keep the relations \( \leq \), this formulation is equivalent to that used in the linear case. Under this scheme, the algorithm solves firstly the problem placed in the so called level 0, whose aim is to determine the existence of feasible points for the problem under study:

\[
\begin{align*}
\min & \quad \sum_{j=1}^{m} p_j \\
\text{s.t.} & \quad g_j(x) - p_j \leq b_j, \quad (j = 1, \ldots, m), \\
p_j & \geq 0, \quad (j = 1, \ldots, m).
\end{align*}
\]

After that, \( s \) problems are solved, one for each priority level, trying in each one to satisfy the corresponding goals, at the same time that the achievements of the previous levels are kept. The general problem corresponding to level \( k \) is as follows:

\[
\begin{align*}
\min & \quad \sum_{i \in N_k} \mu_i p_i \\
\text{s.t.} & \quad g(x) \leq \bar{b}, \\
f_j(x) - p_i \leq u_j, \quad (j \in N_1, \ldots, N_k), \\
f_i(x) - p_i \leq u_i, \quad i \in N_k, \\
p_i & \geq 0, \quad i \in N_k,
\end{align*}
\]

where \( u^*_j \) corresponds to the target value associated to the \( j \)th objective function, actualized after solving the corresponding level; namely, if when solving the problem associated to the level that contains the function \( f_j \) we obtain \( p^*_j = 0 \), that is, the corresponding goal is satisfied, then it is not necessary to actualize the target value, and so, \( u^*_j = u_j \). On the other hand, if we have \( p^*_j > 0 \), then the goal is not satisfied, and we actualize \( u^*_j = u_j + p^*_j \), that is, we maintain the achieved level, which is the best possible, although not satisfying. It must be observed that,
under the conditions imposed to the model, this problem is a non linear convex one, and, so, it can be solved making used of the algorithms that exist to this end. We have used the quadratic sequential scheme described in Gill et al. (1986).

This way, all the s problems are solved along the procedure, and after that, if we have that all the optimal values of the deviation variables are zero, then the original Goal Programming problem has got satisfying solutions, and \( x^{*}_{s} \) (the optimal solution corresponding to the last priority level) is one of them. In the opposite case, there do not exist satisfying solutions, and \( x^{*}_{s} \) is the solution which is ‘closest’ to the satisfaction of all the goals, taking into account the priority levels and the weights given by the decision maker.

In any case, the solution obtained, \( x^{*}_{s} \), is just determined by our main aim, that is, to satisfy the goals, so that, in principle, the algorithm can generate any solution which does so. Nevertheless, in the satisfying set there can exist other solutions that the decision maker may regard as more desirable, may be because \( x^{*}_{s} \) is not efficient, or just because there are other combinations on the decision space that improve objectives placed in higher priority levels, worsening others. In the following sections, two interactive algorithms are developed to study the sensitivity of the solution with respect to changes in the target values, so as to improve the initially obtains solution, offering the additional possibility to generate a Pareto optimal combination, in both cases when satisfying solutions exist or not.

3. Post-optimisation analysis on the target values

3.1. There exist satisfying solutions

As it has been previously mentioned, when there exist satisfying solutions, Goal Programming algorithms tend to find very frequently solutions which are not efficient. The aim of this section is to provide an algorithm that lets us to improve the objectives in a sequential way, in such a form that we finally obtain a satisfying and efficient solution, taking into account the decision maker’s expressed preferences.

In order to carry out this procedure, it is necessary, in our opinion, to take permanently into account the origin of the problem under study, which is none but a Goal Programming scheme, which in turn has associated the introduction of target values and, probably, priority levels, information that must be kept in mind along the procedure of generation of a satisfying and efficient solution. One first simple approach would consist in choosing, within the satisfying set, the combination for which the function placed in the highest priority level (if there is just one) takes the best possible value. The problem then amounts to nothing more than minimizing \( f_{1} \) within the satisfying set (if we suppose that \( f_{1} \) is the function placed in the first priority level). However, this approach is, in some way, a maximalist one, for it does not allow any flexibility in the target values, in the sense that we take for granted that the decision maker prefers to optimize as much as possible the first priority level, ignoring the rest. But we think that the lexicographic scheme does not correspond to such an strict priority organization of the objectives, because the decision maker may wish to improve firstly the first priority level, but leaving some margin to improve the rest as well.

In general, for the \( i \)th objective function, the resolution of the problem

\[
\min \quad f_{i}(x)
\]
\[
\text{s.t.} \quad \bar{g}(x) \leq \bar{b},
\]
\[
f_{j}(x) \leq u_{j}, \quad (j = 1, \ldots, p, j \neq i),
\]

gives us the best possible value that the \( i \)th objective function can take, keeping the rest of the objectives under their respective target values, or, in other words, the best possible value for \( f_{i} \) within the satisfying set. This way, if we denote by \( x^{*}_{i} \) the solution obtained, then we are given an interval

\[
[f_{i}(x^{*}_{i}), u_{i}],
\]

which gives us the possible target values for the \( i \)th objective that improve the original one, and such that there still exist satisfying solutions.

Besides, it can be observed that the problem that has been solved is a constraint one, and thus, if its solution \( x^{*}_{i} \) is unique, it can be affirmed that it is also Pareto optimal (see Sawaragi et al., 1985).

The algorithm that we develop below is based upon these ideas, and consists in improving the objective functions lexicographically, so that the sat-
satisfying set is constrained at each step, and a final solution which is satisfying and efficient at the same time is obtained. Under a first approach, we consider as an additional hypothesis that there is just one function in each priority level (and we suppose that the functions are already ordered according to these levels), and afterwards we will study the general case. This way, the interactive algorithm developed under this hypothesis is as follows:

Algorithm 1. Goal Sequential Improvement (GSI)

Step 0. Initialize $i \leftarrow 1$.

Step 1. Solve the general $i$th problem:

$$\min f_i(x)$$

subject to

$$\overline{g}(x) \leq \overline{b}$$

$$f_i(x) \leq \overline{u}_j, \quad (j = 1, \ldots, i - 1),$$

$$f_i(x) \leq u_j, \quad (j \in i + 1, \ldots, p),$$

and denote its solution by $x_i^*$.

Step 2. If $i = p$, then the algorithm ends with the final solution $x_p^*$, and the value $\overline{u}_p = f_p(x_p^*)$ for the last objective.

Step 3. If $f_i(x_i^*) = u_i$, then take $\overline{u}_i = u_i$, and go to step 5.

Step 4. The decision maker is shown the interval $[f_i(x_i^*), u_i]$ of possible actualized values of the $i$th target value. He is asked to choose a value $\overline{u}_i$ inside it.

Step 5. Actualize $i \leftarrow i + 1$ and go back to step 1.

Firstly, we have to take into account that the problems (4) which are solved at step 1 are, under the initial hypothesis, general non-linear convex programming problems, and thus they can be solved using any of the algorithms available to this end.

Step 4 of the algorithm is its interactive part. In it, the decision maker chooses a new target value for the $i$th objective function, within its possible variation margins. In general, the closer this new value $\overline{u}_i$ is taken to its minimum possible value, the less margin will remain to improve the target values of the objectives placed in the subsequent priority levels. This way, it is the decision maker who decides whether to improve as much as a possible a single objective, or to leave some margin for the remaining ones. Anyway, this target value improvement is carried out following the lexicographic Goal Programming scheme.

Step 3 refers to the possibility of not having any space left to improve a determined target value, and then it is kept at its original value and the process keeps on in case it is possible to improve the subsequent target values, although this is not very probable in practice.

Finally, it must be pointed out that the resolution of the last problem (4) (that is, the problem corresponding to level $p$) gives us a solution $x_p^*$ that satisfies the new goals imposed on the $p - 1$ functions belonging to the previous levels, and takes the least possible value for the last one within this new constrained satisfying set. Besides, as it has been previously mentioned, all the problems which are solved along the algorithm are constraint problems. Then, in particular, if $x_p^*$ is an unique solution of (4), it can be affirmed that it is also Pareto efficient. Due to the way the algorithm has been built, it is highly improbable not to get this uniqueness in practice, specially in general non-linear problems. Anyway, if it was not unique, then it would suffice to solve, for example, the auxiliary problem

$$\min \sum_{i=1}^p \mu_i f_i(x)$$

subject to

$$\overline{g}(x) \leq \overline{b}$$

$$f_i(x) \leq \overline{u}_i, \quad (i = 1, \ldots, p),$$

(where $\mu_i, i = 1, \ldots, p$ are strictly positive normalizing weights) to obtain a solution that is surely efficient (see again Sawaragi et al., 1985), at the same time that satisfies the new goals with all the target values that have been actualized along the procedure.

Therefore, the interactive algorithm GSI lets us harmonize some way the concepts of satisfying and efficient solutions, for it makes possible to improve the target values (and thus, the values of the objective functions) following the Goal Programming lexicographic scheme, and, as a result, we obtain an efficient solution that has been designed by the decision maker, according to his preferences.

Finally, we are going to consider the general case when there are $s$ priority levels, and some of them can contain more than one objective function. To this end, the steps that correspond to such levels are
modified, incorporating inside them an interactive algorithm based upon the STEM method proposed by Benayoun et al. (1971).

This second algorithm has exactly the same behavior than the preceding one in those levels that contain just one objective function. Let us suppose now that a certain level \( k \) contains more than one. If the target values for the \( k - 1 \) preceding levels have already been actualized, then, for each \( i \in N_k \), the algorithm solves the corresponding problem:

\[
\min f_i(x)
\text{s.t. } \begin{align*}
g(x) &\leq \bar{b} \\
f_j(x) &\leq \bar{u}_j, & (j \in N_1, \ldots, N_{k-1}), \\
f_j(x) &\leq u_j, & (j \in N_k, \ldots, N_s, j \neq i),
\end{align*}
\tag{6}
\]

and, as a result, we obtain the corresponding interval \([f_i(x^*_i), u_i]\) for each \( i \in N_k \). However, in this case, if the decision maker arbitrarily chooses a new target value for each function \( f_i, i \in N_k \), it can occur that there do not exist satisfying solutions for them. The idea is then, in case there do not exist such solutions, to carry out an STEM algorithm to assign the new values within the given intervals.

Namely, the subroutine corresponding to the priority levels that contain more than one objective function (which substitutes steps 1, 2, 3 and 4 of the main algorithms for these levels) is as follows:

**Step i1.** Solve the problems (6) \((i \in N_k)\), and show the decision maker the corresponding interval \([f_i(x^*_i), u_i]\) for each \( i \in N_k \). However, in this case, if the decision maker arbitrarily chooses a new target value for each function \( f_i, i \in N_k \), it can occur that there do not exist satisfying solutions for them. The idea is then, in case there do not exist such solutions, to carry out an STEM algorithm to assign the new values within the given intervals.

**Step i2.** Solve the problem

\[
\min \sum_{i \in N_k} \mu_i p_i
\text{s.t. } \begin{align*}
g(x) &\leq \bar{b} \\
f_j(x) &\leq \bar{u}_j, & (j \in N_1, \ldots, N_{k-1}), \\
f_j(x) &\leq u_j, & (j \in N_k, \ldots, N_s, j \neq i), \\
f_i(x) - p_i &\leq \bar{u}_i, & (i \in N_k), \\
p_i &\geq 0, & (i \in N_k),
\end{align*}
\tag{7}
\]

where the weights \( \mu_i \) are obtained from those given by the decision maker in the Goal Programming process, as it was described in Section 2. We denote its optimal solution by \((x^*, p^*)\). If \( p^* = 0 \), the subroutine corresponding to level \( k \) ends, and, we go to step 5 of the main algorithm with the actualized target values \( \bar{u}_i, (i \in N_k) \) for the goals in level \( k \), if \( k < s \), or, the whole procedure ends, if \( k = s \). If \( p^* \neq 0 \), then actualize \( \bar{u}_i \leftarrow f_i(x^*) \) \((i \in N_k)\) and go to the following step:

**Step i3.** The decision maker is shown the new values \( \bar{u}_i \) \((i \in N_k)\) and he classifies them in satisfying (S) and non satisfying (I). If all the target values are satisfying, the subroutine ends, and we go to step 5 of the main algorithm, if \( k < s \), or, the algorithm ends, if \( k = s \). If there exist non satisfying values, then for each satisfying one the decision maker gives a corresponding relaxation margin \( \Delta f_i > 0 \), such that \( \bar{u}_i + \Delta f_i \leq u_i \).

**Step i4.** Solve the problem

\[
\min \xi
\text{s.t. } \begin{align*}
\nu_i(f_i(x) - f_i^*) &\leq \xi, & (i \in I), \\
f_j(x) &\leq \bar{u}_i + \Delta f_i, & (i \in S), \\
f_j(x) &\leq \bar{u}_j, & (j \in N_1, \ldots, N_{k-1}), \\
f_j(x) &\leq u_j, & (j \in N_k, \ldots, N_s, j \neq i), \\
g(x) &\leq \bar{b},
\end{align*}
\tag{8}
\]

where \( f_i^* \) denotes the ideal value of the \( i \)th objective function in the initial feasible set, and the weights \( \nu_i \) are obtained according to the following expressions:

\[
\nu_i = \frac{\omega_i}{\sum_{j \in N_k} \omega_j}, \quad \omega_i = \frac{\max \left\{|f_i^*| - f_i| \right\}}{\min_j \{|f_i^*| \}},
\]

\((i \in N_k)\),

with \( f_i^* \) being the value of the \( i \)th objective function in the ideal solution of the \( j \)th objective.

**Step i5.** Actualize \( \bar{u}_i \leftarrow f_i(x^*) \) \((i \in N_k)\) and go back to i3.

So, in this subroutine, we try to achieve the goals with the new target values given by the decision

---

\(^1\) In Benayoun et al. (1971) the weights originally proposed for the linear STEM methods can be found. The weights we propose are an adaptation to the nonlinear case of these original weights.
maker just in the same way the Goal Programming algorithm would do it. In case it is not possible to satisfy them all, the subsequent interactive STEM method allows us to obtain a 'negotiated' solution that improves the target values objectives belonging to the level, within their possible limits.

3.2. There do not exist satisfying solutions

As it was stated in Section 2, in the case when there do not exist satisfying solutions for the original Goal Programming problem, the algorithm gives us a sequence of optimal values for the corresponding positive deviation variables, \( p_1^*, \ldots, p_p^* \), among which there exists at least one that is strictly positive, and this means that it has been impossible to achieve the corresponding goal, and it also gives us the least possible value that its target value has to take \( (u_i + p_i^*) \) so that we can assure that there exist solutions which satisfy the \( i \)th goal, keeping the values achieved for the previous priority levels. Besides, due to the construction of the algorithm, the subsequent levels have been built in such a way that they take into account this necessary relaxation of the target values corresponding to the unachieved goals. Therefore, the sequence of actualized target values

\[
\tilde{u}_i = u_i + p_i^*, \quad (i = 1, \ldots, p),
\]

(where those corresponding to zero optimal deviation variables remain unaltered) gives us the minimum levels that must be imposed to each goal so that there exists a satisfying solution (being the point \( x_i^* \), obtained after the Goal Programming procedure, one of them).

It must be pointed out that the 'minimum' character of these actualized target values is based on the assumption (that is, in some way, logical from the lexicographic point of view) that the decision maker is not inclined to relax one of these target values more that the strictly necessary quantity, in order to achieve better results for the goals placed in later priority levels. Anyway, it might be possible that the decision maker does not feel satisfied with these target values, specially when there are several objective functions that share the same priority level, in which case the decision maker may want to improve some values at the expense of worsening others.

For this case, an interactive algorithm is proposed, which is basically the STEM, method, modified to allow the possibility of fixing objectives which the decision maker does not want to vary which is a logical fact, for he has previously given a priority order among them. Then, in the global procedure, the decision maker is shown the results obtained in the Goal Programming algorithm and, if he does not feel satisfied with them, then he would proceed to the following interactive algorithm:

**Algorithm 2 (modified STEM)**

*Step 0.* The decision maker is shown the levels \( \tilde{u}_i \), \( (i = 1, \ldots, p) \), obtained in the Goal Programming procedure, as well as the solution \( x_i^* \) achieved.

*Step 1.* The decision maker classifies the target values into satisfying and non satisfying (\( S \) and \( I \) respectively). Among the satisfying one, he can also point those that he wishes to remain under their current values (\( F \)), that is, those which he does not want to relax. Obviously, if there exist non satisfying levels, there must be some satisfying one which can be relaxed, that is, if \( I \neq \emptyset \), then is must be \( S \setminus F \neq \emptyset \).

If all the levels are satisfying, the algorithm ends. If not, it goes to step 2.

*Step 2.* The decision maker gives a maximum relaxation margin \( \Delta f_i \), for each target value \( i \in S \setminus F \).

*Step 3.* Solve the problem:

\[
\min_{x, \xi} \xi \quad s.t. \quad \nu_i(f_i(x) - f_i^*) \leq \xi, \quad i \in I, \\
\quad f_i(x) \leq \tilde{u}_i + \Delta f_i, \quad i \in I \setminus F, \\
\quad f_i(x) \leq \tilde{u}_i, \quad i \in F, \\
\quad \bar{g}(x) \leq b,
\]

where the weights \( \nu_i \) corresponding to each objective function are defined as described at step 4 of Algorithm 1.

*Step 4.* Denote by \( x^* \) the optimal solution of problem (9), and actualize the target values: \( \tilde{u}_i \leftarrow f_i(x^*) \), \( (i = 1, \ldots, p) \). Go back to step 1.

As a result of the application of this interactive procedure, a final solution negotiated \( x^* \) is obtained, which, obviously, does not verify the original goals of the problem, but which fits better the preferences of the decision maker, within the non satisfying solutions.
4. Examples and numerical results

In this section, two examples are shown and some numerical results are displayed. The first example shows the behavior of the algorithm on a simple problem, so that a graphical interpretation of the interactive procedure can be given. The second example is a problem with more goals, so that the whole interactive procedure is used. Finally, computational results on test problems with different number of variables and objectives are displayed.

Example 1. Let us consider the following problem:

\[
\begin{align*}
\min & \quad (x-3)^4 + (y-2)^4, (x-1)^2 + (y-3)^2, y - \ln x \\
\text{s.t.} & \quad x^2 + y^2 \leq 9, \\
& \quad y - x^3 \geq 0, \\
& \quad x \geq 0.5, y \geq 0,
\end{align*}
\]

where the following goals and priority levels are considered:

Level 1: \( f_1(x,y) \leq 16 \);  
Level 2: \( f_2(x,y) \leq 1 \);  
Level 3: \( f_3(x,y) \leq 3 \).

Using the Goal Programming lexicographic algorithm described in Section 2, the final satisfying solution \( x^*_1 = (1.083, 2.003) \) is obtained, for which the objective functions take the following values:

\[
\begin{align*}
f_1(x^*_1) &= 13.507, \\
f_2(x^*_1) &= 1, \\
f_3(x^*_1) &= 1.924.
\end{align*}
\]

Once this solution has been obtained, we proceed to carry out the interactive goal sequential improvement algorithm, following the priority levels that the decision maker originally gave, so that we will have to solve three problems in order to redefine the three target values.

At level 1, the problem:

\[
\begin{align*}
\min & \quad f_1(x,y) = (x-3)^4 + (y-2)^4 \\
\text{s.t.} & \quad x^2 + y^2 \leq 9, \\
& \quad y - x^3 \geq 0, \\
& \quad x \geq 0.5, y \geq 0, \\
& \quad (x-1)^2 + (y-3)^2 \leq 1, \\
& \quad y - \ln x \leq 3,
\end{align*}
\]

is solved, and its optimal solution \( x^*_1 = (1.386, 2.661) \) is obtained, where the first objective function takes the value \( f_1(x^*_1) = 6.982 \) (see Fig. 1). Therefore, the decision maker is shown the interval of possible values for the first target value, that improve the first goal, at the same time the existence of satisfying solutions is assured: \([6.982, 16)\).

Let us now suppose that the decision maker chooses new the value \( \bar{a}_1 = 10 \) for the first target value. The algorithm then proceeds to solve the second problem, where the second objective function is minimized, taking already into account this new value for the first goal:

\[
\begin{align*}
\min & \quad f_2(x,y) = (x-1)^2 + (y-3)^2 \\
\text{s.t.} & \quad x^2 + y^2 \leq 9, \\
& \quad y - x^3 \geq 0, \\
& \quad x \geq 0.5, y \geq 0, \\
& \quad (x-3)^4 + (y-2)^4 \leq 10, \\
& \quad y - \ln x \leq 3.
\end{align*}
\]

After the resolution of this problem (which can be graphically seen in Fig. 2), we obtain a new optimal point:

\( x^*_2 = (1.235, 2.734) \), with \( f_2(x^*_2) = 0.126 \).

This way, a new interval of the form \([0.126, 1)\) is obtained for the second target value. Let us now suppose that the decision maker chooses the value \( \bar{a}_2 = 0.5 \).

Finally, the third and last problem of the interactive algorithm is solved, so as to determine the best possible value for \( f_3 \) considering the new target values for the previous priority levels (Fig. 3):

\[
\begin{align*}
\min & \quad f_3(x,y) = y - \ln x \\
\text{s.t.} & \quad x^2 + y^2 \leq 9, \\
& \quad y - x^3 \geq 0, \\
& \quad x \geq 0.5, y \geq 0, \\
& \quad (x-3)^4 + (y-2)^4 \leq 10, \\
& \quad (x-1)^2 + (y-3)^2 \leq 0.5.
\end{align*}
\]

Its resolution gives us the point:

\( x^*_3 = (1.335, 2.377) \), with \( f_3(x^*_3) = 2.088 \).

Besides, in this case, the first objective function takes in the final optimal point the value 7.713, and, so, it has been improved even more (this behavior,
Fig. 1. Level 1. The shadowed zone is the feasible set for problem 10, that is, the set of points that are feasible for the original problem, and also satisfy the second and third goals. $f_2$ is minimized in this set, obtaining the solution $x_1^*$.

Fig. 2. Level 2. The feasible set displayed in Fig. 1 is now restricted, when asking to achieve the new target value for the first goal. $f_2$ is minimized in this new feasible set, obtaining the solution $x_2^*$. 
however, is not very likely to take place in a real problem). On the other hand, it can be proved that the point obtained is the unique optimal solution of problem (12) and, thus, it is Pareto optimal (in Fig. 3 it can be seen that, really, there is not alternative solution to the problem). Anyway, in case of doubt, it would suffice to solve the auxiliary problem (5) mentioned in Section 3.1.

**Example 2.** Let us now consider the problem:

\[
\begin{align*}
\min & \quad (x + y + z)^{5/2} , (x - 4)^4 + (y - 3)^4 + z^4, \quad \frac{1}{x + y + z} , \quad 3x^3 + (y - 1)^4 + 2(z - 20)^4, \quad (x - 2)^2 - \ln(y + z), \\
\text{s.t.} & \quad (x - 3)^2 + (y - 3)^2 + (z - 3)^2 \leq 4, \\
& \quad x + 2y + z \leq 15,
\end{align*}
\]

and let us assume that the Decision Maker initially wants to achieve the following goals:

\[f_1 \leq 80, \quad f_2 \leq 20, \quad f_3 \leq 0.15, \quad f_4 \leq 75000, \quad f_5 \leq 60.\]

Finally, let us assume that the first goal is assigned to the first priority level, the second, third and fourth goals are assigned to the second priority level (with equal weights) and the fifth goal is in the third priority level.

After solving the Goal Programming problem, the final solution \((2.15, 3.48, 1.67)\) is obtained, which satisfies all the goals. Let us now describe an application of the GSI algorithm to this problem:

**Level 1.** After solving the corresponding problem, the DM is presented the interval \([47.066, 80)\) for the Table 1

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Function values</th>
<th>Satisfied</th>
<th>New value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(f_2 = 11.66)</td>
<td>No</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>(f_5 = 0.129)</td>
<td>Yes</td>
<td>0.13</td>
</tr>
<tr>
<td></td>
<td>(f_4 = 65783)</td>
<td>Yes</td>
<td>70000</td>
</tr>
<tr>
<td>2</td>
<td>(f_2 = 3.81)</td>
<td>Yes</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>(f_5 = 0.129)</td>
<td>Yes</td>
<td>0.13</td>
</tr>
<tr>
<td></td>
<td>(f_4 = 70000)</td>
<td>No</td>
<td>–</td>
</tr>
<tr>
<td>3</td>
<td>(f_2 = 5)</td>
<td>Yes</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>(f_5 = 0.129)</td>
<td>Yes</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>(f_4 = 68801.4)</td>
<td>Yes</td>
<td>–</td>
</tr>
</tbody>
</table>
first target value. Let us assume that the new value 60 is chosen.

**Level 2.** The algorithm solves the three problems corresponding to the level, and presents the intervals [1.722, 20], [0.129, 0.15] and [64602.65, 75000] for the second, third and fourth goals respectively. The DM then chooses the values 5, 0.13 and 65000, respectively. These three values are unachievable at the same time, and so, an interactive STEM procedure is carried out within the current priority level, which is displayed in Table 1.

**Level 3.** In the level three, the fifth objective function takes its best possible value, which is the one that it already took in the second level, for no more margin exists to improve it. So, the GSI procedure ends with the final solution (2.74, 3.82, 1.19), and the values 60, 5, 0.129, 68801.4 and 51.12 for the five objective functions. The global CPU time that took the resolution of this problem was 5.26 seconds.

4.1. **Numerical results**

Next, the computational results obtained with some test problems with different number of variables and objective functions are displayed in Table 2.

The third column displays the number of linear and nonlinear constraints of the test problems (in this order), and the CPU time displayed in the fourth column corresponds to the arithmetic mean of the times that took the resolution of several problems of the same dimensions. In order to homogenize these times, never more than 4 iterations were performed within each priority level.

Basically, these times are the ones taken in the resolution of the scalar nonlinear problems, and thus, they depend on the number of variables and the number of constraints (specially the nonlinear ones), as it can be deduced from Table 2.

5. **Conclusion**

Finally, the following conclusions can be stated:

- The interactive algorithms developed throughout this paper supply us with an useful tool which allow us to overcome the weakness of Goal Programming algorithms in relation with their inclination to produce dominated solution, according to the decision maker’s preferences. Besides, this post optimisation scheme by priority levels lets us, on the one hand, to carry out an exhaustive analysis of the possibility to improve the originally obtained solution, and, on the other hand, to take into account the preferences that have been expressed by the decision maker through the imposition of priority levels, so that the final solution corresponds to a lexicographic improvement of the objective functions.

- The intrinsic simplicity of these algorithms makes it possible to obtain good results with a minimum computational cost, if compared with that of carrying out a complete analysis of the structure of the satisfying and efficient set. The table displayed in the previous section shows that the computing times for the different problems are very satisfying. Basically, any goal programming scheme has to solve a small number of scalar problems, and so, its quickness and computational efficiency depends on the algorithm chosen to solve the nonlinear single objective problems. In our case, as it has been previously mentioned, a sequential quadratic scheme is used, and the algorithm has been implemented on a VAX 8530 computer, in FORTRAN language and with the aid of the subroutine library NAG, Mark 15 (see NAG, 1991).

- The construction of the algorithms lets the decision maker observe the effect of the improvement
of each objective function on the remaining margin for the following ones, and this supplies an implicit knowledge of the trade-offs among such objectives, and making use of an information which is more tangible from the user's point of view.

Finally, it must be pointed out that these algorithms harmonize in some way the concepts of efficient and satisfying solutions, which have traditionally been in conflict between them, because it obtains efficiency through a lexicographic scheme which makes possible not to abandon during the post-optimisation procedure the Goal Programming philosophy of the problem itself.

Acknowledgements

The authors wish to express their gratitude to two anonymous referees for their valuable and helpful comments. The last version of this paper was written while Francisco Ruiz was visiting the University of Georgia, USA. His research was supported by the grant number PR95-287, of the Spanish Ministry of Education and Sciences, Dirección General de Investigación Científica y Técnica (DGICYT).

References


