Journal of the Operational Research Society (2001) 52, 1359–1369
www.palgrave-journals.com/jors

Goal programming using multiple objective tabu search

A Baykasoğlu*
University of Gaziantep, Gaziantep, Turkey

Goal programming (GP) is one of the most commonly used mathematical programming tools to model multiple objective optimisation (MOO) problems. There are numerous MOO problems of various complexity modelled using GP in the literature. One of the main difficulties in the GP is to solve their mathematical formulations optimally. Due to difficulties imposed by the classical solution techniques there is a trend in the literature to solve mathematical programming formulations including goal programmes, using the modern heuristics optimisation techniques, namely genetic algorithms (GA), tabu search (TS) and simulated annealing (SA). This paper uses the multiple objective tabu search (MOTS) algorithm, which was proposed previously by the author to solve GP models. In the proposed approach, GP models are first converted to their classical MOO equivalent by using some simple conversion procedures. Then the problem is solved using the MOTS algorithm. The results obtained from the computational experiment show that MOTS can be considered as a promising candidate tool for solving GP models.

Keywords: goal programming; multiple objective tabu search; multi-objective optimisation

Introduction

Charnes and Cooper[1] proposed using GP to solve multi objective optimisation (MOO) problems. GP has been studied by many researchers[2,3] and successfully applied[4,5] to many problems. There are two basic types of GP. These are pre-emptive goal programming and weighted goal programming. Pre-emptive goal programming is a special case of GP, in which the more important (upper level) goals are optimised before lower level goals are considered. In non-pre-emptive models, the goals are assigned weights and considered simultaneously. There are several classical techniques for solving GP models. Some of these are: simplex based approaches (separable programming, approximation programming, etc[3]), modified pattern search,[6] interactive approach,[7] and gradient-based approach.[3] Unfortunately, classical techniques impose several limitations on solving mathematical programming models including the GP ones.[8] The problem is mainly related to inherent solution mechanisms of these techniques. Their solution strategies are generally dependent on the type of objective and constraint functions (linear, non-linear, etc) and the type of variables used in the problem modelling (integer, real, etc).[9] Their efficiency is also dependent on the size of the solution space, number of variables and constraints used in the problem modelling, and the structure of the solution space (convex, non-convex, etc).[10] They also do not offer a general solution strategy that can be applied to problem formulations where different type of variables, objective and constraint functions are used.[11] For example, *simplex algorithm* can be used to solve models with linear objective and constraint functions; or *geometric programming* can be used to solve non-linear models with a posynomial or signomial objective function, etc. However, most of the engineering problems require different types of variables, objective and constraint functions simultaneously in their formulation. Therefore, classic optimisation procedures are generally not adequate or easy to use for their solution.[8]

Researchers have spent a great deal of effort in order to adapt many engineering problems to the classic optimisation procedures. Many examples can easily be found in the literature. An interesting application from the author's previous research, which was related to cutting conditions optimisation using geometric and dynamic programming might be an interesting example.[12] It is not easy to formulate a real life problem that suits a specific solution procedure. In order to achieve this, it is necessary to make some modifications and/or assumptions on the original problem parameters (rounding variables, softening constraints, etc). This certainly affects the solution quality.

A new set of problem and model independent heuristic optimisation techniques were proposed by researchers to overcome drawbacks of the classical optimisation procedures. These techniques are efficient and flexible.[13] They can be modified and/or adapted to suit specific problem requirements (see Figure 1). Three of these widely accepted and applied techniques are known as *genetic algorithms*,[14] *tabu search*[15] and *simulated annealing*.[16] Research on these techniques is still continuing all around the world.

In this study the focus is on TS. As mentioned above, TS belongs to the class of problem-independent, heuristic

*Correspondence: *A Baykasoğlu, Department of Industrial Engineering, University of Gaziantep, 27310 Gaziantep Turkey.*
E-mail: baykasoglu@gantep.edu.tr

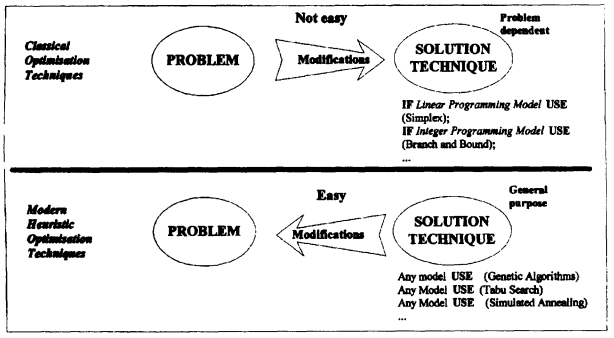

Figure 1 A pictorial comparison of classical and modern heuristic optimisation strategies.

global optimisation methods and can handle any kind of objective functions and constraints.[10] TS is generally applied to single objective optimisation problems such as cell formation, scheduling and plant layout, etc. TS has the potential to solve MOO problems because it works with multiple solutions at the same time. This gives TS the opportunity to evaluate multiple objective functions simultaneously. Based on this observation, TS has recently been applied to MOO problems by the author. In his previous studies, the author[17] extended the original TS algorithm to find the Pareto optimal set in MOO by including several additional features in it. Results of this study proved that the multiple objective version of TS is effective in finding Pareto optimal solutions in MOO. The author[18] also developed a specific TS algorithm for solving pre-emptive GP models. This algorithm was able to find a single solution to pre-emptive GP in one run. Comparative studies showed that the algorithm was effective. Using this algorithm several design problems[19,20] were also solved.

This paper uses the MOTS algorithm proposed previously by the author to solve GP models. For this purpose, the GP models are first converted to their classical MOO equivalent by using some simple conversion procedures, which were proposed by Deb.[21] Then the MOTS algorithm is used for solving the model. Problem specific modifications to the MOTS algorithm are presented in this paper. By using the proposed approach, it is possible to find a set of candidate solutions to a given goal programme. The proposed approach also avoids the problem of weighting the goals in 'weighted GP' and ordering the goals in 'pre-emptive GP'. The results obtained from the test problems have shown that MOTS can successfully find many alternative solutions to a given GP.

Goal programming and its conversion to an equivalent MOO program

In its general form a GP problem can be formally stated as follows:

$$z_q = \sum_{i=1}^{m}(w_{qi}^{+}d_i^{+} + w_{qi}^{-}d_i^{-}) \qquad q = 1, 2, \ldots, t$$

such that

$$\begin{aligned} f_i(X) + d_i^{-} - d_i^{+} &= b_i & i &= 1, 2, \ldots, m \\ g_j(X) &\leqslant 0 & j &= 1, 2, \ldots, m_1 \\ h_k(X) &= 0 & k &= 1, 2, \ldots, m_2 \\ d_i, d_i^{+} &\geqslant 0 & i &= 1, 2, \ldots, m \end{aligned} \tag{1}$$

where t is the total number of priority levels, q is the priority level, $z_1, z_2, \ldots, z_q$ are the goal deviations that they will be optimised, d_i^{+} is the positive deviation variable representing the over-achievements of goal i, d_i^{-} is the negative deviation variable representing the under-achievements of goal i, w_{qi}^{+} is the positive weight assigned to d_i^{+} at priority q, w_{qi}^{-} is the negative weight assigned to d_i^{-} at priority q, X is an n-dimensional decision vector, f_i is a function of goal constraints, g_i is a function of real inequality constraints, h_i is a function of real equality constraints, b_i is the target value of goal i, m, is the number of goal constraints, m_1 is the number real inequality constraints and m_2 is the number of real equality constraints

In pre-emptive GP, the user is able to prioritise the order in which goals are to be achieved.[2,3] It finds an optimal solution that satisfies as many of the goals as possible, in the order specified (lexicographic order). In a pre-emptive GP, the function to be optimised is put in a form as given in Equation (2). In Equation (2), lexmin means optimising in the lexicographic order:

$$\text{lexmin}\{z_1, z_2, \ldots, z_q\} \tag{2}$$

where $z_1 >>> z_2 >>> \cdots >>> z_q$. Several researchers argue that such consideration of goals is most practical; however some others criticise this approach.

In weighted GP, there is no pre-emption the function to be optimised is defined as the weighted summation of goal deviations as shown in Equation (3):

$$\min\{\lambda_1 z_1 + \lambda_2 z_2 + \cdots + \lambda_q z_q\} \tag{3}$$

where $\lambda_1, \lambda_2, \lambda_q$ are the weights. The main criticism of this approach is the determination of weights.

In a GP implementation, there are generally three cases of concern for the minimisation of deviations (d_i^{+}, d_i^{-}). These are minimise both negative and positive deviations (d_i^{+}, d_i^{-}), minimise only the positive deviation (d_i^{-}) and minimise only the negative deviation (d_i^{-}). Minimisation of both deviations means that it is required to equate the objective function to its target value (ie goal$(f_i(X) = b_i)$). This goal can be converted to the following form: minimise $|f_i(X) - b_i|$. Minimising only the positive deviation means that it is required to satisfy the following goal $f_i(X) - d_i^{+} \leqslant b_i$. This goal can be converted to the following form: $\min\langle f_i(X) - b_i\rangle$. The bracket operator $\langle\ \rangle$ returns the value of the operand if the operand is positive, otherwise it returns zero. Minimising only the negative deviation means that it is required to satisfy the following goal:

$f_i(X) + d_i^- \geqslant b_i$. This goal can be converted to the following form: minimise $\langle b_i - f_i(X) \rangle$.

If one makes these conversions then the deviational variables can be excluded from the GP formulation. This actually converts a GP formulation to its classical MOO equivalent.[22] For example consider the following GP formulation:

$$\text{lexmin}\{z_1 = (d_1 + d_1^+), \quad z_2 = (d_2^-), \quad z_3 = (d_3^+)\}$$

subject to

$$\begin{aligned} &7x_1 + 3x_2 + d_1^- - d_1^+ = 40 \\ &10x_1 + 5x_2 + d_2^- - d_2^+ = 60 \\ &5x_1 + 4x_2 + d_3^- - d_3^+ = 35 \\ &100x_1 + 60x_2 \leqslant 0 \\ &x_1, x_2, d_1^-, d_1^+, d_2^-, d_2^+, d_3^-, d_3^+ \geqslant 0. \end{aligned} \tag{4}$$

After conversion, the equivalent MOO problem become as follows:

$$\begin{aligned} &\min |7x_1 + 3x_2 - 40| \\ &\min\langle 60 - 10x_1 + 5x_2 \rangle \\ &\min\langle 5x_1 + 4x_2 - 35 \rangle \end{aligned} \tag{5}$$

subject to

$$\begin{aligned} &100x_1 + 60x_2 \leqslant 0 \\ &x_1, x_2 \geqslant 0. \end{aligned}$$

Although the converted objective functions cannot be easily handled with classical optimisation routines, TS can handle them easily, because having differentiable objective functions is not compulsory in TS. In the following section the MOTS algorithm[17] that is used for the solution of GP models is presented.

The multiple objective tabu search algorithm (MOTS)

TS is a heuristic neighbourhood search algorithm, which has been developed for solving single objective optimisation problems. The basic TS algorithm operates in the following way: it starts from a randomly selected, feasible seed solution (s); from this seed solution, a set of neighbourhood solutions (s^*) is generated using a number of previously determined movement strategies. The objective function is evaluated for each solution in s^* and the best neighbour replaces the seed solution, even though it may be worse than s: in this way it is possible to escape from local minima (or maxima) of the objective function. The algorithm iterates until some given stopping condition is reached. There is a danger that the algorithm as described above may recycle old solutions and become trapped in a loop. To avoid this situation, the last m seed solutions are stored in a list, which is called the *tabu list*. The neighbours of the current solution that appear in the tabu list are systematically eliminated unless they meet an aspiration criterion, so at each iteration the algorithm is forced to select a solution not recently explored. The main stages of the basic TS algorithm are: *initial solution*, *generation of neighbours*, *selection and updating*.

The idea of applying TS to MOO comes from its solution structure, in working with more than one solution (neighbourhood solutions) at a time. In fact, any solution methodology that works with more than one solution vector at a time can be effectively used for MOO, such as genetic algorithms.[14] Due to its population-based search characteristic, the genetic algorithms are frequently applied to MOO problems. To enable the TS algorithm to work with more than one objective, selection and updating stages of the basic TS are redefined. Other stages are similar to the original algorithm. In contrast to the original TS algorithm, the MOTS algorithm has two more lists in addition to the tabu list. The first one is the *Pareto list*, which collects selected non-dominated solutions found by the algorithm. The second one is the *candidate list*, which collects all other non-dominated solutions, which are not selected as Pareto optimal solutions in the current iteration. These solutions may become seed solutions if they maintain their non-dominated status in later iterations. The candidates list also plays an important role, as it gives the opportunity to diversify the search.[17]

The elements of the proposed TS algorithm for finding Pareto optimal solutions in MOO problems with any type of variables (integer, zero-one, discrete or continuous) and performance functions (linear, non-linear, convex, non-convex) are defined as follows.[17]

Initial solution

Any randomly generated or known feasible solution vector is the initial solution.

Generation of neighbourhood solutions

To generate a neighbour for any type of variable, new values are formulated as follows:[17,18]

integer variable

$$x_i^* = x_i + \text{integer}[(2 \times \text{random}() - 1) \times stepi_i]$$

zero-one variable

$$x_i^* = \begin{cases} 1 & \text{if } x_i = 0 \\ 0 & \text{if } x_i = 1 \end{cases} \tag{6}$$

discrete variable

$$x_i^* = d_{(l + \text{integer}[(2 \times random() - 1 \times stepd_i])} \qquad \text{if } x_i = d_l$$

continuous variable

$$x_i^* = x_i + (2 \times random() - 1) \times stepc_i$$

where:

x_i: value of the ith variable prior to the neighbourhood move,

x_i^*: value of the ith variable after the neighbourhood move,
random( ): random number generator, where *random*() ∈ (0,1)
$stepi_i$, $stepd_i$, $stepci_i$: step size for integer, discrete and real variables,
d_l: the lth element of the discrete variable subset X^d,
integer[ ]: function to convert a real value to an integer value.

According to the types of variables used in the model, the appropriate movement strategies are used to generate a previously determined number of feasible, non-tabu, neighbourhood solutions from the current seed solution. Neighbourhood solutions must also be non-dominated by the current seed solution.

Selection of the seed solution

Selection of the seed solution is performed using the Pareto optimality logic (domination and non-domination). Pareto optimality is an economics term for describing a solution for multiple objectives. It is generally used to characterise optimal solutions to a MOO problem. The Pareto optimal (non-dominated) solution is defined as follows: a solution $x^* \in s$ is Pareto optimal if and only if there exists no $x \in s$ such that $f_i(x) \leqslant f_i(x^*)$ for $i = 1, 2, 3, \ldots, m$ with $f_i(x) < f_i(x^*)$ for at least one value of i. In other words, the solution x^* is Pareto optimal if no objective function can be improved without worsening at least one other objective function. Based on the Pareto optimality logic, the selection of the best neighbourhood solution as the new seed solution is performed in the following manner.

(i) For each neighbourhood solution vector, the corresponding objective function values are calculated. In the example given below, the neighbourhood size is three and there are two real variables and two objective functions to be maximised (see 1 below).
(ii) Candidate seed solutions within the neighbourhood solutions are identified. Candidate seed solutions should not be dominated by other neighbourhood solutions, solutions in the Pareto list or solutions in the candidate list. This process is illustrated below, using the same example (see 2 below).
(iii) One of the candidate solutions is randomly selected as the new seed solution. This process is illustrated below using the same example.

If there are no candidate solutions in the current neighbourhood, the oldest solution from the candidate list is selected as the seed solution.

1.

Seed solution followed by objective function values			3 neighbourhood solutions (non-tabu, feasible and not dominated by the seed solution)		Corresponding objective function values of neighbourhood solutions
			(6.3 6.1)	→	(60.08 47.09)
(4.8 4.6)	(52.4 40.93)	→	(6 6)	→	(58.78 46.54)
			(6.4 6)	→	(60.39 46.86)
Variable values	Objective values		Variable values	→	Objective values

2.

Seed solution followed by objective function values			Neighbourhood solutions (non-tabu, feasible and not dominated by the seed solution)		Objective function values of neighbourhood solutions
			(6.3 6.1)	→	(60.08 47.09) ○
(4.8 4.6)	(52.4 40.93)	→	(6 6)	→	(58.79 46.54)
			(6.4 6)	→	(60.39 46.86) ○

(○ : Candidate solutions)

Pareto list

(0 0)(0 0)×
(0.5 0.5)(16.97 13.44)×
(1 1)(24 19)×
(2 2)(33.94 26.87)×
(3 3)(41.57 32.91)×
(3.8 3.6) (46.57 36.26)×
(4.8 4.6) (52.4 40.93)
(×: Eliminated solution from previous iterations)

Candidate list

(4 3)(46.93 33.98)×
(3 4) (42.64 36.93)×

It can be seen from the above selection strategy that the dominated solutions are not taken into consideration, because the purpose is to find the Pareto optimal solutions, which do not dominate each other. This approach is totally different from the single objective and some MOO approaches, which reduce the problem to a single objective optimisation problem. In these approaches a non-improving solution (which is the best current neighbour in the TS) can be accepted as the current solution if it is not tabu. If it is tabu and passes a previously defined aspiration criterion it is still accepted, even if it is worse than the previous current solution. This strategy works well and has the ability to find a good solution in the single objective optimisation. But it is not easy (or may not be possible) to evaluate multiple objectives simultaneously and determine the Pareto optimal solutions with this strategy because it concentrates only on single objective function evaluations. For this reason the MOTS algorithm offers the present intelligent strategy, which works with two more dynamic lists, namely the Pareto list and candidate list. As was mentioned at the beginning of this section, the candidate list (which collects potential candidate Pareto optimal solutions and updates their status dynamically) enables the search process to avoid abandoning them while searching and diversify the search (this case can also be imagined as avoiding to fall into the trap of local optima in global optimisation). The Pareto list collects the seed (or currently selected) potential Pareto optimal solutions and dynamically updates their status.

Updating the lists

The initial feasible solution vector is recorded as the first known Pareto solution vector. The solutions, which are dominated by any neighbourhood solution, are removed from both Pareto and candidate lists in each iteration. Then the seed solution is added to the Pareto list, and other candidate solutions are put into the candidate list. This process is shown on the same example below.

Pareto list	*Candidate list*
(0 0)(0 0)×	(4 3)(46.93 33.98)×
(0.5 0.5)(16.97 13.44)×	(3 4)(42.64 36.93)×
(1 1)(24 19)×	(6.3 6.1)(60.08 47.39)
(2 2)(33.94 26.87)×	
(3 3)(41.57 32.91)×	
(3.8 3.6)(46.57 36.26)×	
(4.8 4.6)(52.4 40.93)×	
(6.4 6)(60.39 46.86)	

(×: Eliminated solution)

Selected seed solutions for an arbitrarily defined number of previous moves are considered as tabu, since reusing one of them may trap the algorithm into cycling through recent moves. In our algorithm, the tabu list contains m solutions, corresponding to the last m seed solutions. The tabu list is circular, ie when it is full a new item replaces the head of the list.

Aspiration criteria

In combinatorial optimisation problems, solution vectors are generally generated indirectly using several features of the problem at hand. For example, in a manufacturing cell formation problem a new solution can be generated by randomly reassigning part-machine pairs to different cells in each iteration in order to obtain a new solution.[19] In such a case, instead of putting the whole solution vector in the tabu list as tabu solution only, indices (features) of randomly selected and reassigned part-machine pairs are put into the tabu list. However, reselection of these features for new solution generation in later iterations might generate different solution vectors, as these features themselves are not the solution vectors. Therefore, optimal solutions may be missed if these features are considered strictly as tabu. In order to prevent this situation in TS applications an aspiration criterion needs to be defined to override the tabu status of features when necessary. But if the entire solution vector is put into to the tabu list then it is not necessary to define an aspiration criterion.

In this study, the aspiration criterion is defined as follows: any neighbour that is not dominated by the seed solution and solutions in the Pareto and candidate lists is accepted, even if it is tabu.

Termination

If a previously determined number of iterations is reached, or if the candidate list is empty and the algorithm cannot find any new candidate solutions, the program terminates.

Guidelines for the determination of tabu search parameters

Tabu list size (m), number of iterations (*iter*), convergence criteria (t), and neighbourhood size (*nneigh*) and step size for the variables constitute tabu search parameters. Determination of these parameters is generally problem dependent. There is no unique analytic strategy or methodology to fix these parameters in the current literature. It is a common practice to solve the problem with different sets of these parameters to find a good combination of the parameter set. Within these parameters, determination of step sizes and neighbourhood size for real and integer variables is particularly important. If the ranges of variables are too wide, neighbourhood size is small and the corresponding step sizes are chosen very small then computational time increases considerably. With the same conditions, if step sizes are chosen too big then it may be possible to miss the optimal solution. As guidance for setting up the tabu search parameters, the following rule can be used.[18]

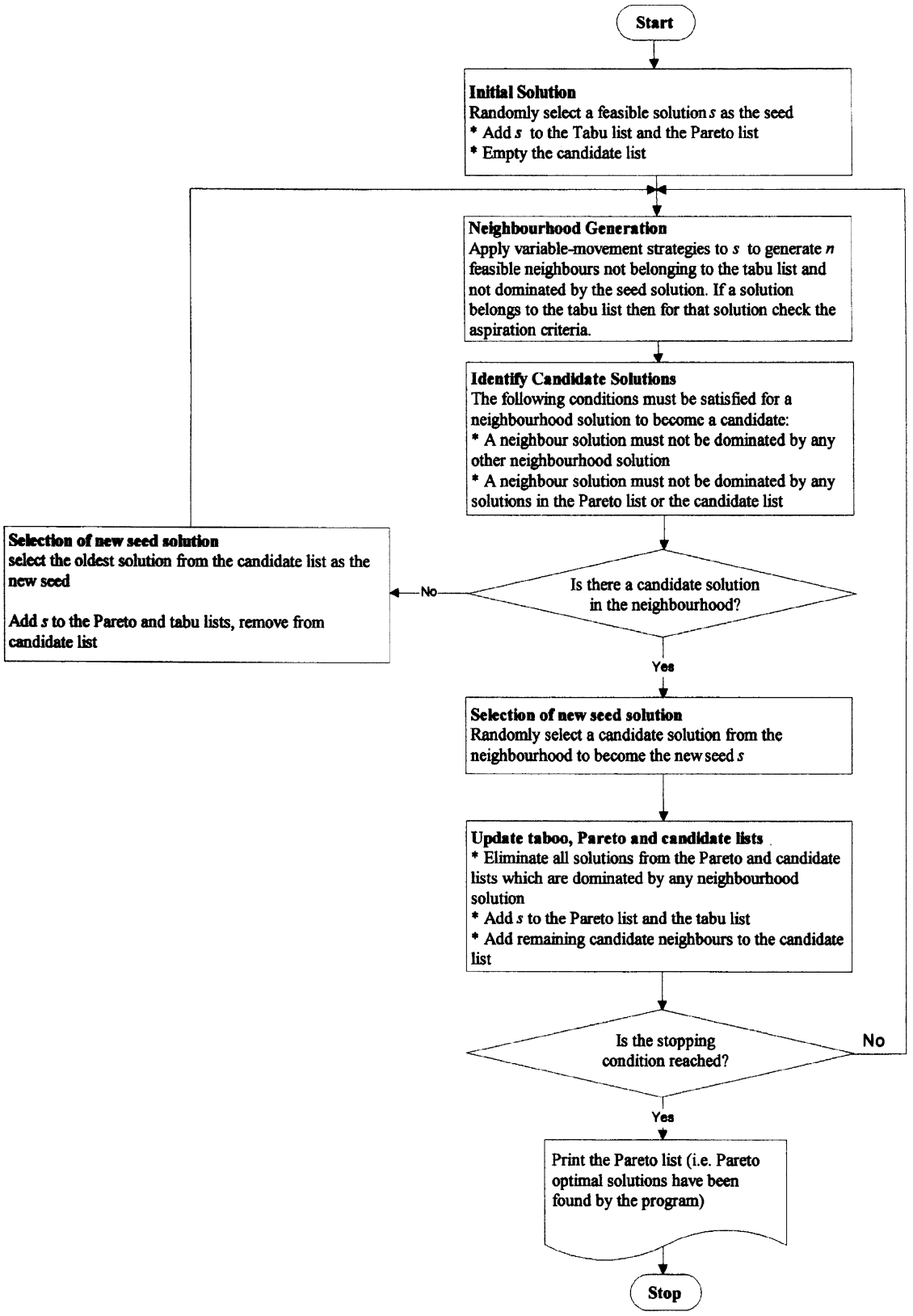

Figure 2 The flowchart of the tabu search algorithm to solve MOO problems.

MOTS parameters setting rule. If the ranges of variables are too wide then use big step sizes, otherwise prefer smaller step sizes. If step sizes are big then increase the number of neighbourhood solutions in each iteration, otherwise smaller number of neighbourhood solutions can safely be used. Make sure that converge criteria is satisfied before terminating, therefore number of iterations should be big enough to assure convergence.

The general flowchart of the algorithm is given in Figure 2.

In order to present the behaviour of the MOTS algorithm while searching for the Pareto optimal set, a schematic representation of a problem with two maximisation objectives is shown in Figure 3.[17]

In Figure 3, number 1 shows the initial seed solution. Solutions 2, 3 and 4 are generated from solution 1, all of which are feasible and not dominated by solution 1. Solution 4 is dominated by other neighbour solutions so it is not a candidate seed solution. There are two candidate seed solutions, namely solutions 2 and 3.

In the next iteration, solution 3 is selected as the seed solution. Solutions 5, 6 and 7 are generated; solutions 5 and 6 are candidates. Solution 2 loses its candidate status and solution 3 loses its Pareto status (both are dominated by solution 5). Solution 6 is randomly selected as the seed solution for the next iteration.

Solutions 8, 9 and 10 are generated from solution 6. Solutions 8, 9 and 10 are all candidate solutions ie they are not dominated by the current seed, Pareto solutions or other candidate solutions) and solution 6 loses its Pareto solution status. Solution 10 is randomly selected as the current seed solution. Solutions 11, 12 and 13 are generated. There is no candidate seed solution in this iteration, so the oldest candidate solution (solution 5) is selected as the seed solution. Solutions 14, 15 and 16 are generated from solution 5. Solutions 1 and 5 lose their Pareto solution status. Solution 15 is selected as the current seed solution and so on. As can be seen from Figure 3 in every iteration solutions move towards the trade-off curve simultaneously.

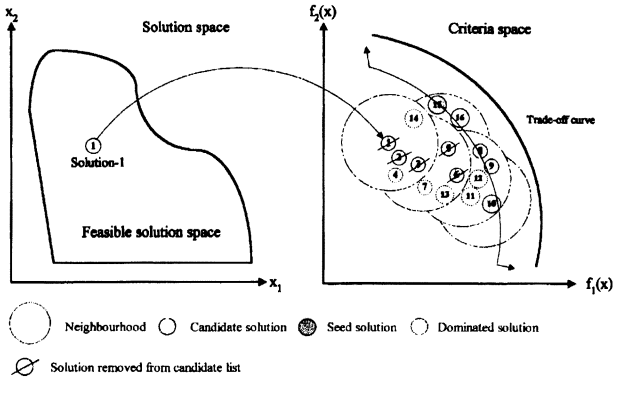

Figure 3 Behaviour of the MOTS algorithm while searching.

After finding the multiple solutions to a given GP using the MOTS, it is possible to choose one particular solution using compromise programming or several other decision-making approaches. Each solution x found by MOTS can be analysed to find relative importance of each objective function as follows.[21]

$$\gamma_i = (|b_i|/|f_i(x) - b_i|)/\left(\sum_{j=1}^{m} |b_j|/|f_j(x) - b_j|\right) \qquad (7)$$

In other words, Equation (7) determines a normalised weight for each obtained solution by evaluating the distance between the corresponding objective function values and the target objective function values.

Example applications

The proposed MOTS algorithm was applied to several GP problems collected from the literature. In each application the algorithm successfully found good solutions in comparison to the reported solutions. The current improved version of MOTS algorithm is programmed using C++. The program is an object oriented one and uses advanced linked list constructs. The program is tested on a Pentium III MMX model PC at 450 MHz (64 MB RAM).

Example 1

The problem given in Equation (8) is taken from Deb.[21] He solved this problem using the non-dominated sorting genetic algorithm (NSGA). For solving this problem MOTS parameters are set as follows: neighbourhood size = 10, tabu list size = 20, step size for the first and second variables are 0.2 and 3, respectively, maximum number of iterations is set to 1000. Using these parameters MOTS found 148 solutions after 398 iterations in 8 s.

$$\begin{aligned} &\min\langle 10x_1 - 2\rangle \\ &\min\langle((10 + x_2 - 5)^2)/10x_1) - 2\rangle \end{aligned}$$

subject to

$$\begin{aligned} &0.1 \leqslant x_1 \leqslant 1 \\ &0 \leqslant x_2 \leqslant 10 \\ &x_1, x_2 \geqslant 0 \quad \text{and continuous.} \end{aligned} \qquad (8)$$

All obtained non-dominated solutions are shown in Figure 4, which also marks the true solutions of the given GP with different weighting coefficients by solid and dashed lines. The figure shows that MOTS is able to find many different solutions in the desired range. Although other regions (for $f_1 < 2$ and $f_2 < 2$) on the hyperbola are Pareto optimal solutions of the problem of minimising f_1 and f_2, the reformulation of the objective function allows MOTS to find only the required region, which is also a solution of the GP problem. As can be seen from Figure 4, MOTS found only the true solutions of this GP in the desired range in a

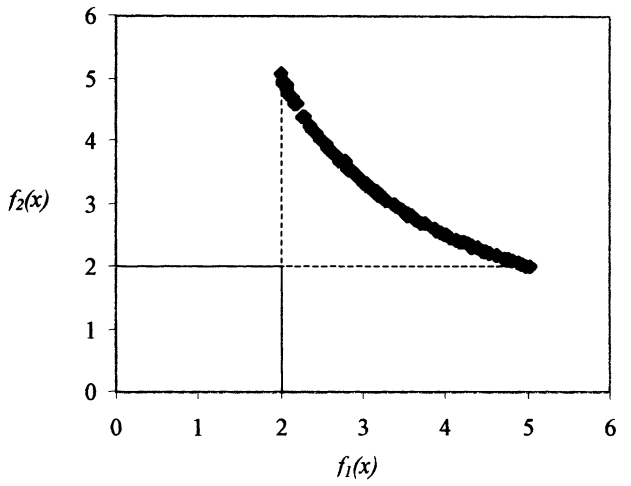

Figure 4 MOTS solutions for test problem 1.

single run. The MOTS is also run with different parameter settings and no significant differences were observed on the solution quality. Although Deb[21] did not give much details about the solutions obtained from his NSGA, no significant difference between the quality of the solutions obtained from MOTS and NSGA is observed.

In Table 1, six different solutions obtained using MOTS is shown. This table also shows relative weight factors for each solution computed using Equation (7). If the first objective is considered more important then the solutions in the first and second row can be chosen. If the second objective is considered more important then the solutions in the third and fourth rows can be chosen. If both objectives are considered of more or less equal importance then solutions in the fifth and sixth rows can be chosen. It can be said here that there is an advantage of using a technique like MOTS in finding all such solutions simultaneously in one single run. This certainly enhances the use of GP in many applications.

Example 2

The problem given in Equation (9) is also taken from Deb.[21] He solved this problem using NSGA. For solving this problem MOTS parameters are set as follows: neighbourhood size = 10, tabu list size = 20, step size for all variables is taken as 0.2, and maximum number of iterations is set to 1000. Using these parameters MOTS found 481 solutions after 576 iterations in 11 s:

$$\begin{aligned} &\min\langle 0.9 - x_1\rangle \\ &\min|((1 - \sqrt{x_1(1-x_1)})(1 + 10x_2^2)) - 0.55| \\ &\text{subject to} \\ &\quad 0 \leqslant x_1 \leqslant 1 \\ &\quad 0 \leqslant x_2 \leqslant 1 \\ &\quad x_1, x_2 \geqslant 0 \quad \text{and continuous.} \end{aligned} \tag{9}$$

All obtained solutions are shown in Figure 5. The figure also marks with a solid line the criterion space of the given GP. The solution to this GP is the region marked with dashed lines. Each solution in this region corresponds to a GP problem with a specific set of weights. The MOTS is also run with different parameter settings for this problem. No significant differences observed on the solution quality. In comparison to NSGA the number of solutions found by MOTS is higher, but both techniques successfully detected the trade-off curve.

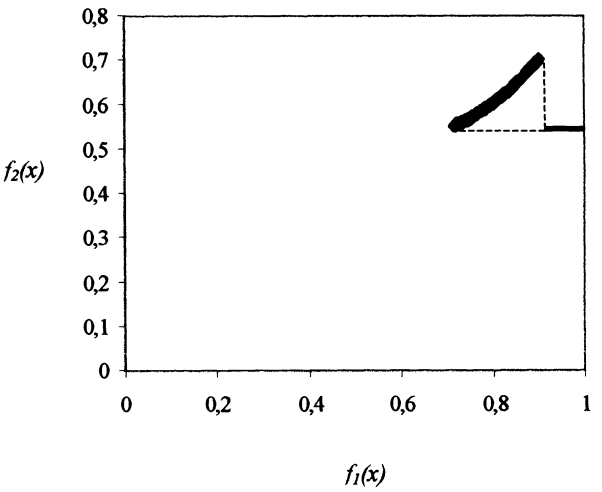

Figure 5 MOTS solutions for test problem 2.

Table 1 Six solutions to GP 1 and their relative weight factors

x_1	x_2	$f_1(x)$	$f_2(x)$	γ_1	γ_2
0.217 368	5.011 64	2.173 682	4.600 55	0.937	0.063
0.214 243	4.839 65	2.142 425	4.679 61	0.950	0.050
0.480 766	4.887 27	4.807 66	2.082 658	0.029	0.971
0.477 167	4.761 65	4.771 67	2.107 609	0.037	0.963
0.305 324	4.805 85	3.053 24	3.287 55	0.550	0.450
0.319 278	4.971 67	3.192 78	3.132 32	0.487	0.513

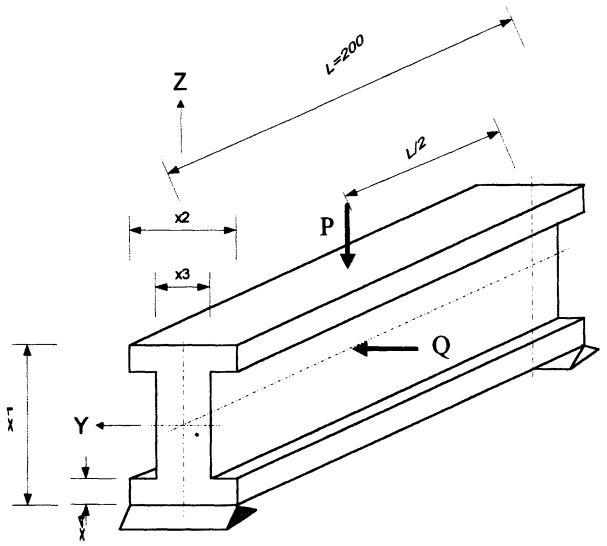

Figure 6 The simply supported I-beam of test problem 3.

Example 3 (design of a simply supported I-beam design)

This test is a design optimisation problem for a simply supported I-beam, which is shown in Figure 6. Osyczka[23] originally modelled this problem. Coello and Christiansen[24] remodelled the problem as a MOO problem and solved the model using their genetic algorithm, which is known as MOSES. They also made an extensive comparison of their results with some other MOO techniques with respect to the best results obtained for each objective function. In this test study, their model is presented as a GP model and solved by MOTS. The model is given in Equation (11). More details about the model can be obtained from Coello and Christiansen.[24]

$$\min\langle(2x_2x_4 + x_3(x_1 - 2x_4)) - 127.46\rangle$$

$$\min\left\langle\left(\frac{60\,000}{x_3(x_1 - 2x_4)^3 + 2x_2x_4(4x_4^2 + 3x_1(x_1 - 2x_4)}\right) - 0.0059\right\rangle$$

subject to

$$16 - \frac{180\,000x_1}{x_3(x_1 - 2x_4)^3 + 2x_2x_4(4x_4^2 + 3x_1(x_1 - 2x_4))} - \frac{15\,000x_2}{(x_1 - 2x_4)x_3^3 + 2x_4x_2^3} \geqslant 0 \qquad (11)$$

$$10 \leqslant x_1 \leqslant 80$$
$$10 \leqslant x_2 \leqslant 50$$
$$0.9 \leqslant x_3 \leqslant 5$$
$$0.9 \leqslant x_4 \leqslant 5$$
$$x_1, x_2, x_3, x_4 \leqslant 0 \quad \text{(continuous)}.$$

For solving the I-beam design problem MOTS parameters are set as follows; neighbourhood size = 10, tabu list size = 20, step sizes for the first, second, third and fourth variables are taken as 5, 5, 2, 2 and maximum number of iterations is set to 1000. Using this parameter set MOTS found 92 solutions after 178 iterations in 8 s. All obtained solutions are shown in Figure 7. The MOTS is also run with different parameter sets and no significant difference is observed on the solution quality. The comparison of the best values of objectives obtained from MOTS and other techniques reported in Coello and Christiansen[24] shows that solutions obtained from the MOTS are not dominated.

Example 4 (design of a machine tool spindle)

This test is another design optimisation problem for a machine tool spindle, which is shown in Figure 8. The problem is originally modelled by Eschenauer *et al.*[25] Coello[26] remodelled this problem as a MOO problem and solved it using their genetic algorithm, which is known as MOSES. They also compared their results with four other MOO techniques with respect to be best results obtained for each objective function. In this test study, their model is presented as a GP model and solved by MOTS. The model is given in Equation (12). More details about the model can be obtained from Coello.[26]

$$\min\left\langle\left(\frac{\pi}{4}[a(d_a^2 - d_o^2) + l(d_b^2 - d_o^2)]\right) - 450\,000\right\rangle$$

$$\min\left\langle\left(\frac{Fa^3}{3El_a}\left(1 + \frac{l}{a}\frac{l_a}{l_b}\right) + \frac{F}{c_a}\left[\left(1 + \frac{a}{l}\right)^2 + \frac{c_a a^2}{c_b l^2}\right]\right) - 0.011\right\rangle$$

$$l_a = 0.049(d_a^4 - d_0^4),\ l_b = 0.049(d_b^4 - d_0^4),$$
$$c_a = 35\,400|\delta_{ra}|^{1/9} d_a^{10/9},$$
$$c_b = 35\,400|\delta_{rb}|^{1/9}\, d_b^{10/9}$$

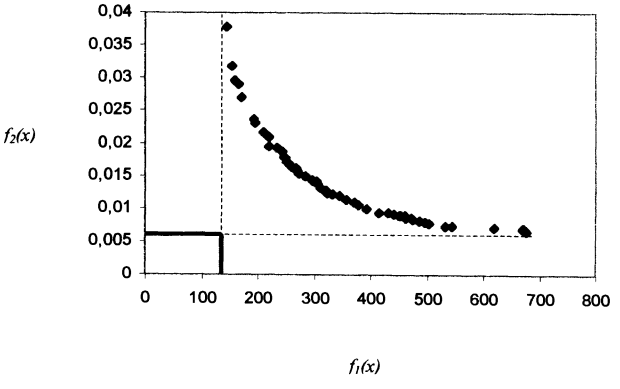

Figure 7 MOTS solutions for test problem 3.

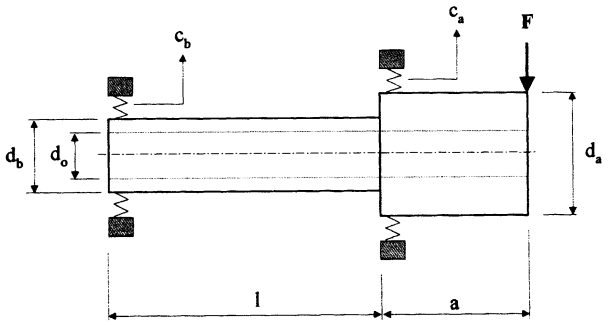

Figure 8 Sketch of the machine tool spindle of test problem 4.

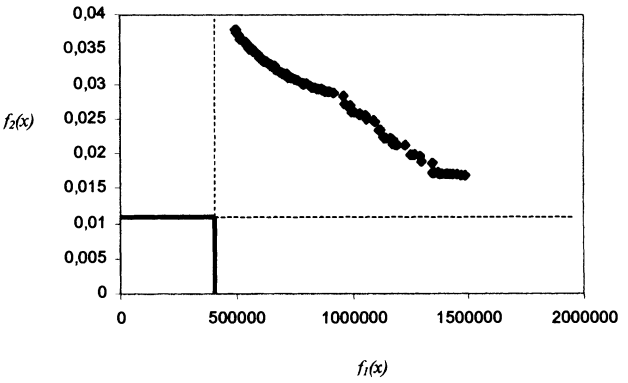

Figure 9 MOTS solutions for test problem 4.

subject to

$$
\begin{aligned}
&l - l_g \leqslant 0\\
&l_k - l \leqslant 0\\
&d_{a1} - d_a \leqslant 0\\
&d_a - d_{a2} \leqslant 0\\
&d_{b1} - d_b \leqslant 0\\
&d_b - d_{b2} \leqslant 0\\
&d_{om} - d_o \leqslant 0\\
&p_1 d_o - d_b \leqslant 0\\
&p_2 d_b - d_a \leqslant 0\\
&\left|\Delta_a + (\Delta_a - \Delta_b)\frac{a}{l}\right| - \Delta \leqslant 0
\end{aligned}
\qquad (12)
$$

$d_o, l \geqslant 0$, continuous and d_a, d_b discrete.

In the above model d_a and d_b are discrete variables and d_a should be selected from the following set {80, 85, 90, 95}, and d_b from the set {75, 80, 85, 90}. For solving the spindle design problem, MOTS parameters are set as follows: neighbourhood size = 10, tabu list size = 20, step sizes for the first, second, third and fourth variables are taken as 5, 5, 4, 4 and maximum number of iterations is set to 1000. Using this parameter set MOTS found 128 solutions after 348 iterations in 9 s. All obtained solutions are shown in Figure 9. The MOTS is also run with different parameter sets and no significant differences observed on the solution quality. The comparison of the best values of objectives obtained from MOTS and other techniques reported in Coello[26] shows that solutions of MOTS are not dominated. These comparisons are also shown in Table 2.

Conclusions

Any optimisation technique, which works with more than one solution vector in its inherent solution mechanism, such as TS, can be effectively used for solving MOO models. TS is a heuristic neighbourhood search algorithm that works with a set of potential solutions known as neighbourhood solutions. This gives TS an advantage in solving MOO problems directly without requiring an additional method. Based on this observation, Baykasoglu *et al*[17] recently developed the multiple objective version of TS, which is known as MOTS.

In this study an application of MOTS to solve the GP models is presented. The proposed approach can solve any type of GP models without requiring any weighting factor for goals. The proposed method can produce multiple solutions to a given GP. The decision maker can select

Table 2 Comparison of the best results for each objective of the test problem 4 (for each method the best results for $f_1(x)$ and $f_2(x)$ are shown in bold)

Techniques	d_o	l	d_a	d_b	$f_1(x)$	$f_2(x)$
Monte Carlo 1	59.08	189.17	90	75	**606 765.47**	0.032 463
Monte Carlo 2	26.26	193.29	90	85	1457 748.36	**0.019 242**
GA (binary)	60.00	200.00	80	75	**494 015.44**	0.038 087
GA (binary)	25.00	190.09	95	90	1643 777.68	**0.016 613**
GA (floating point)	56.16	194.49	95	90	**1124 409.37**	0.017 951
GA (floating point)	25.35	189.58	95	90	1637 052.38	**0.016 615**
Literature	63.89	183.29	85	80	**531 183.70**	0.030 215
Literature	66.45	183.36	95	85	694 200.03	**0.023 101**
MOTS	59.84	199.26	80	75	**497 644.1**	0.037 839
MOTS	39.02	199.62	85	80	1485 169	**0.016 894**

one of these solutions for implementation. The efficiency of the proposed technique is presented with four test studies, including two engineering design problems. The results are encouraging. It can finally be concluded that MOTS is a viable candidate for solving GP models.

References

1 Charnes A and Cooper WW (1961). *Management Models and Industrial Applications of Linear Programming.* John Wiley & Sons: New York.

2 Sang ML (1972). *Goal Programming for Decision Analysis.* Averbach: Philadelphia.

3 Ignizo JP (1982). *Linear Programming in Single & Multiple Objective Systems.* Prentice Hall: Englewood cliffs, NJ.

4 Myint S and Tabucanon MT (1994). A multiple criteria approach to machine selection for flexible manufacturing systems. *Int. J Prod Econ* **33**: 121–131.

5 Gokcen H and Erel E (1997). A goal programming approach to mixed-model assembly line balancing problem. *Int J Prod Econ* **48**: 177–185.

6 Clayton E, Weber W and Taylor III B (1982). A goal programming approach to the optimisation of multiresponse simulation models. *IIE Trans* **14**: 282–287.

7 Dyer J (1972). Interactive goal programming. *Mngt Sci* **19**: 62–70.

8 Dhingra AK and Lee BH (1994). A genetic algorithm approach to single and multiobjective structural optimization with discrete-continuous variables. *Int J Num Meth Engng* **37**: 4059–4080.

9 Wilhelm MR and Ward TL (1987). Solving quadratic assignment problems by simulated annealing. *IIE Trans* **3**: 107–119.

10 Siarry P and Berthiau G (1997). Fitting of Tabu search to optimise functions of continuous variables. *Int J Num Meth Engng* **40**: 2449–2457.

11 Malasri S, Martin JR and Medina RA (1996). Solving mathematical programming problems using genetic algorithms. *Comp Civil Engng* **1**: 233–239.

12 Sonmez AI, Baykasoglu A, Dereli T and Filiz IH (1999). Dynamic optimization of multipass milling operations via geometric programming. *Int J Machine Tools Manufac* **39**: 297–320.

13 Chipperfield A and Fleming P (1996). Multiobjective gas turbine engine controller design using genetic algorithms. *IEEE Trans Indust Electron* **43**: 583–587.

14 Holland JH (1992). Genetic algorithms. *Sci Am* **7**: 44–50.

15 Glover F (1990). Tabu search: a tutorial. *Interfaces* **20**(1): 74–94.

16 Kirkpatrick S, Gelatt Jr CD and Vecchi MP (1983). Optimisation by simulated annealing. *Science* **220**: 671–680.

17 Baykasoglu A, Owen S and Gindy N (1999). A taboo search based approach to find the Pareto optimal set in multiple objective optimisation. *Engng Optm* **31**: 731–748.

18 Baykasoglu A, Owen S and Gindy N (1999). Solution of goal programming models using a basic taboo search algorithm. *J Opl Res Soc* **50**: 960–973.

19 Baykasoglu A and Gindy NNZ (2000). MOCACEF 1.0: Capability based approach to form part-machine groups for cellular manufacturing applications. *Int J Prod Res.* **38**: 1133–1161.

20 Baykasoglu A and Gindy NNZ (1999). Loading flexible cell production systems: A tabu search based multiple objective simulation optimisation approach. In: Hillery MT and Lewis HJ (eds). *15th International Conference on Production Research, 9–13 August, 1999* vol. 2. University of Limerick, Ireland. Taylor and Francis, pp 1441–1444.

21 Deb K (1999). Non-linear goal programming using multi-objective genetic algorithms. *Techical Report* (CI-60/98), University of Dortmund, Germany.

22 Romero C (1991). *Handbook of Critical Issues in Goal Programming*. Oxford: Pergamon.

23 Osyczka A (1985). Design Optimization. In: Gero JS (ed). *Multicriteria Optimization for Engineering Design*. Academic: New York, pp 193–227.

24 Coello CCA and Christiansen AD (1999). MOSES: A multiple objective optimization tool for engineering design. *Engng Optm* **31**: 337–368.

25 Eschenauer H, Koski J and Osyczka A (1990). *Multicriteria Design Optimization*. Springer-Verlag: Berlin.

26 Coello CCA (1996). An Empirical study of evolutionary techniques for multiobjective optimization in engineering design. PhD thesis, Department of Computer Science, Tulane University, New Orleans, LA.

Received July 2000;
accepted June 2001 after one revision