INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING, VOL. 40, 2449–2457 (1997)

FITTING OF TABU SEARCH TO OPTIMIZE FUNCTIONS OF CONTINUOUS VARIABLES

P. SIARRY

Ecole Centrale de Paris, Laboratoire d'Electronique et de Physique Appliquée, Grande Voie des Vignes, 92295 Châtenay-Malabry, France

G. BERTHIAU

CEA, Centre d'Etudes Nucléaires de Saclay, 91191 Gif-sur-Yvette, France

SUMMARY

Tabu Search (TS) is a stochastic global optimization procedure which proved efficient to solve various combinatorial optimization problems. However, very few works deal with its application to global minimization of functions depending on continuous variables. The aim of this paper is to propose an adaptation of TS to the optimization of continuous functions, and to study the influence of the main algorithm parameters on the convergence towards the optimum. In particular, the application of TS to function optimization involves the definition of the current solution neighbourhood and the management of the tabu list. The efficiency of TS applied to continuous global optimization has been tested in detail by using classical multimodal functions for which minima are known. © 1997 by John Wiley & Sons, Ltd.

KEY WORDS: tabu search; global optimization; continuous variables

1. INTRODUCTION

The underlying principles of Tabu Search (TS) were first exposed by Glover.[1,2] More recently overviews about TS and its implementation in various fields have been proposed, for instance by Hertz *et al.*[3] and Glover *et al.*[4] We recall that TS is a heuristic optimization technique developed specifically for combinatorial problems, such as graph coloring,[5] quadratic assignment,[6] electronic circuit design[7] and scheduling.[8]

TS is an iterative search method which operates roughly in the following way. It starts from an initial solution s, randomly selected. From this 'current solution' s, a set of 'neighbours', called s', is generated by applying to s each of the perturbations, or 'moves', which have been defined beforehand. If the set of neighbours of s is too large, only a subset is generated, composed of a predetermined k number of neighbours, randomly selected. The objective function to be minimized is evaluated at each generated solution s', and the best s' becomes the new current solution, even if it is worse than s: hence it is possible to escape from the local minima of the objective function. Then a new 'iteration' is performed: the previous procedure is repeated by starting from the new current point, until some given stopping condition is reached. Typically, the algorithm is stopped after M iterations without any improvement of the objective function. However the algorithm, as described above, may easily recycle, if a point just already visited is generated again. To avoid this danger, the m last retained current points are stored in a list, called the 'tabu list'. The neighbours of the current solution which belong to the tabu list are

CCC 0029–5981/97/132449–09$17.50

Received 26 September 1995
Revised 3 September 1996

systematically eliminated, so that the algorithm is forced to select at each iteration a point not recently selected before.

The general flow chart of this algorithm, called 'simple Tabu Search', is shown in Figure 1. The main stages of simple TS appear clearly: initial solution, generation of neighbours and selection. There exists more sophisticated versions of TS: in particular, a solution belonging to the tabu list can loose its tabu status if its 'aspiration level' is high enough.[2] We do not describe here these refinements, because this paper is devoted to the adaptation of simple Tabu Search to the optimization of continuous objective functions. However, we think that most of the refinements proposed in the literature can be easily incorporated in our algorithm, which is close to a combinatorial simple TS.

Until now, we are aware of only one paper by Hu[9] dedicated to the adaptation of TS to continuous optimization. But the algorithm proposed by Hu seems to us rather far from original TS. Conversely our purpose is to keep as close as possible to original simple TS.

The paper is organized as follows. In Section 2, we deal with our adaptation of TS to continuous function optimization. In Section 3, the new algorithm's efficiency is tested through several functions of which local and global minima are known. The sensitivity of the main parameters of the method is studied and a comparison is made with pure Random Search and Simulated Annealing. Some words of conclusion constitute the Section 4.

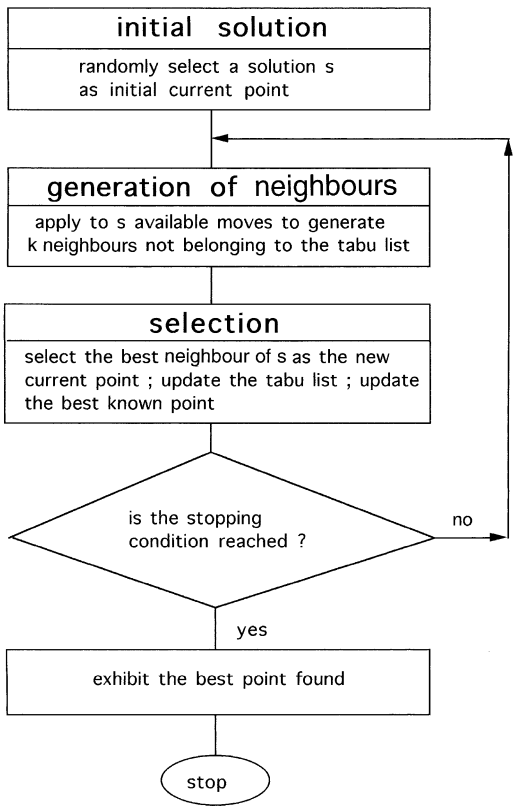

Figure 1. General flow chart of simple tabu search

2. FITTING OF TABU SEARCH TO CONTINUOUS OPTIMIZATION

To discretize the algorithm sketched on Figure 1, two issues must be examined: the generation of current solution neighbours and the elaboration of the tabu list.

Generation of current solution neighbours

To define a neighbourhood of the current solution, we used, as in Reference 9, the notion of a 'ball'. Let $B(s, r)$ be the ball centered on s with the radius r; it contains the set of points s' such as: $\|s' - s\| \leqslant r$.

A first implementation of continuous TS could consist in selecting the neighbours of the current solution s randomly inside a ball $B(s, h)$. However, the number of generated neighbours being generally small (typically between 4 and 20), this procedure leads to an inhomogeneous exploration of the space around s. Therefore we performed a partition of the space around the current solution s, considering a set of concentric balls with radii $h_0, h_1, \ldots, h_k$. This partition is shown in Figure 2, in the case of a 2-variables function $f(x_1, x_2)$, for $k = 4$.

Hence the space is partitioned into concentric 'crowns' $C_i(s, h_i, h_{i-1})$ such as

$$C_i(s, h_i, h_{i-1}) = \{s' | h_{i-1} \leqslant \|s' - s\| < h_i\}$$

The k neighbours of s are obtained by selecting one point at random inside each crown C_i, for i varying from 1 to k. One checks that each selected point is not inside any of the tabu list balls defined further, else one randomly selects another point in the considered crown. The ball $B(s, h_0)$, that is the immediate neighbourhood of s, is excluded, to force a significant moving away of the current solution.

The radii h_i, for $i = 1$ to k, were fixed according to one of the following methods:

Geometrical partitioning: the radii h_i are calculated according to a geometrical progression of ratio 2, to privilege internal regions:

$$h_{k-i+1} = h_k/2^{i-1}, \quad i = 1, \ldots, k$$

The radius h_0 is an independent parameter. Hence the partitioning of space around s depends on three parameters: the number k of neighbours, the radius h_k of the external ball and the radius h_0 of the internal ball.

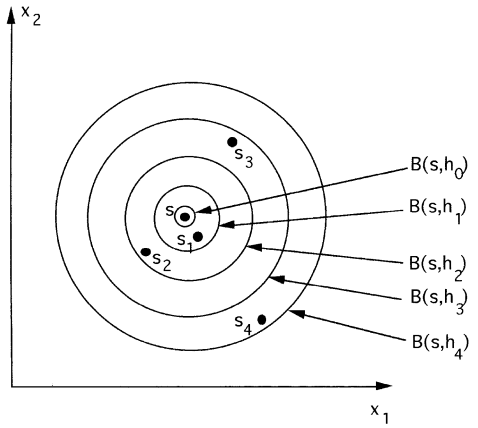

Figure 2. Partition of the current solution neighbourhood

Linear partitioning:

$$h_{k-i+1} = h_k \cdot (k - i + 1)/k, \quad i = 1, \ldots, k$$

Isovolume partitioning:

$$h_{i+1} = (h_i^n - h_i^n/k)^{1/n}, \quad i = 0, \ldots, k-1$$

where n is the number of objective function variables.

This method intends to define domains having the same volume, so that neighbours far from the current solution can be considered with a higher probability.

As in the classical combinatorial TS algorithm, the best of the k neighbours of s, called s^*, becomes the new current solution, even if it is worse than s.

We point out that the above generation of current solution neighbours intentionally does not exploit any information related to the gradient of the objective function: in fact, in most of the practical applications, this gradient is not available or its computation is time expensive. In that sense our algorithm can be classified as a 'direct' optimization algorithm. However, when handling objective functions of which the analytic expression is explicitly known (such as functions we have used to test our algorithm), a potentially more efficient variant of our algorithm could easily be designed by selecting the current solution neighbours along the gradient direction.

Elaboration of the tabu list

Before iterating the procedure described above around the new current solution s^*, one puts into the tabu list the ball $B(s, \varepsilon)$, which constitutes the immediate neighbourhood of the solution s retained just before s^*. Thus one temporarily prohibits coming back too close to s. ε was set equal to h_0. The tabu list contains m balls corresponding to the m last retained solutions: the last ball which enters the list eliminates the oldest one.

3. EXPERIMENTAL RESULTS

The TS algorithm adapted to the optimization of continuous functions was tested through classical multiminima functions for which minima are known.[10,11] For each test function, we adopted the hyperrectangular search domain associated with its definition, as given in the usual literature.

The functions used were the following ones:

Goldstein-Price 2 *variables*:

$$G(\mathbf{x}) = [1 + (x_1 + x_2 + 1)^2 * (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$$
$$\times [30 + (2x_1 - 3x_2)^2 * (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$$

search domain: $-2 \leqslant x_j \leqslant 2, j = 1, 2$;
4 local minima; lowest minimum: $\mathbf{x}^* = (0, -1)$; $G(\mathbf{x}^*) = 3$.

Hartmann 3 *variables*:

$$H(\mathbf{x}) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right]$$

i	a_{ij}			c_i	p_{ij}		
1	3·0	10·0	30·0	1·0	0·3689	0·1170	0·2673
2	0·1	10·0	35·0	1·2	0·4699	0·4387	0·7470
3	3·0	10·0	30·0	3·0	0·1091	0·8732	0·5547
4	0·1	10·0	35·0	3·2	0·03815	0·5743	0·8828

search domain: $0 \leqslant x_j \leqslant 1, j = 1, 3$;
4 local minima; $\mathbf{p}_i = (p_{i1}, p_{i2}, p_{i3})$ is the ith local minimum approximation; $H(\mathbf{p}_i) \# -c_i$.

Rosenbrock n variables ($n = 2$, 5 and 10):

$$R_n(\mathbf{x}) = \sum_{j=1}^{n-1} [100(x_j^2 - x_{j+1})^2 + (x_j - 1)^2]$$

search domain: $-5 \leqslant x_j \leqslant 10, j = 1, n$;
lowest minimum: $\mathbf{x}^* = (1, \ldots, 1)$; $R_n(\mathbf{x}^*) = 0$.

Six parameters characterizing the adapted TS have an important influence on its convergence, namely the number k of points generated around the current solution, the cardinal m of the tabu list, the maximal number M of iterations without any improvement of the objective function, the radius h_k of the external ball of space partitioning around s, the radius h_0 of the internal ball of space partitioning around s, and the radius ε of balls belonging to the tabu list.

We have also tested the three types of current solution neighbourhood partitioning: geometrical, linear and isovolume partitionings.

The section is divided into two parts. In the first one, typical continuous TS results are presented and compared to results from a pure random search method and from simulated annealing. In the second part, the sensitivity of continuous TS main parameters is discussed.

Typical continuous TS results

The results presented were obtained using the following parameter default values:

$$k = m = 5, \quad M = 400, \quad h_k = 1, \quad h_0 = \varepsilon = 0{\cdot}01$$

and a geometrical partitioning of current solution neighbourhood.

To avoid any hazardous interpretation of optimization results, related to the choice of particular starting points, we performed each test 100 times, starting from different points randomly selected in the search domain. Indeed TS being partly a stochastic algorithm, results yielded by one single execution cannot be considered as trustworthy. Therefore, the TS tests were systematically performed 100 times, with different initial seeds for the pseudo-random numbers generator, and hence different starting points. The results of TS tests performed on five functions are reported in Table I. To appraise the program efficiency, we retained the following criteria summarizing results from 100 minimizations, for each function: rate of successful minimizations (yielding points very 'close to'—in a sense defined further—the global optimum of the objective function, called FOBJ), standard deviation of final 'errors', defined as FOBJ gaps between best successful points found and the known global optimum, average of FOBJ evaluations numbers. We considered one minimization to be 'successful' as soon as the FOBJ gap between the best point found and the known global optimum fell under a specified bound. This bound was taken as a small fraction (typically equal to 1·0E − 05) of the FOBJ initial value average. Then

Table I. Average results of continuous Tabu Search

Function	Rate of successful minimizations	Standard deviation of final 'errors'	Average of FOBJ evaluations numbers
G	100	0·008	1636
H	100	0·013	528
R_2	100	0·010	1616
R_5	83	0·005	52 733
R_{10}	70	0·003	263 299

Table II. Comparison of continuous TS with pure random search and simulated annealing with respect to FOBJ evaluations numbers

Function	Continuous tabu search TS	Pure random search RS	Ratio TS/RS	Simulated annealing
G	1636	42401	1/30	783
H	528	17567	1/30	698
R_2	1616	43113	1/30	796
R_5	52 733	$> 2{\cdot}10^6$	$< 1/40$	5364
R_{10}	263 299	—	$< 1/50$	12 316

a comparison is proposed in Table II with pure Random Search (RS) and Simulated Annealing (SA). RS was stopped as soon as the solution reached was approximately equal to that provided by TS. The continuous SA algorithm used for the comparison was that designed by the authors and described in detail in Reference 12.

We see that the rates of successful minimizations are high and the FOBJ evaluation numbers are reasonable for functions of 2 or 3 variables. The number of objective function evaluations is about 30 times lower for TS than for RS. But when the number of variables increases, the TS efficiency collapses in comparison with simulated annealing: for instance, the number of objective function evaluations is about 20 times higher for TS than for SA for the Rosenbrock 10-variables function.

Sensitivity of continuous TS main parameters

Sensitivity of the radius ε of tabu list balls. We tested various values of ε between 0·001 and 0·2. But for the functions used, the optimal value of this parameter is highly dependent on the function. Results observed for the functions G and R_2, in the case $k = m = 5$, are reported in Tables III and IV. Naturally, the number of moves inside the tabu list is strongly correlated to the value of ε. But the accuracy of the result and the speed of continuous TS convergence are weakly dependent on ε. Roughly speaking, an excessive $\varepsilon(\varepsilon > 0{\cdot}05)$ induces numerous moves rejected, which is costly in CPU-time. In the opposite case, an insufficient $\varepsilon(\varepsilon < 0{\cdot}005)$ implies an unused tabu list and a bad speed of convergence. For the functions under test, the best compromise seems to be reached around $\varepsilon = 0{\cdot}01$.

Table III. Sensitivity of the radius ε of tabu list balls for the Goldstein–Price function

Radius ε of tabu list balls	Standard deviation of final 'errors'	Average of FOBJ evaluations numbers	Average number of moves inside the tabu list
0·2	0·008	1646	1836
0·1	0·008	1664	546
0·01	0·008	1636	47
0·001	0·007	1676	0·6

Table IV. Sensitivity of the radius ε of tabu list balls for the 2-variables Rosenbrock function

Radius ε of of tabu list balls	Standard deviation of final 'errors'	Average of FOBJ evaluations numbers	Average number of moves inside the tabu list
0·2	0·017	1659	1384
0·1	0·012	1720	294
0·01	0·010	1616	2·8
0·001	0·031	1788	0·03

Table V. Sensitivity of the cardinal m of the tabu list for the Goldstein–Price function

Cardinal m of the tabu list	Standard deviation of final 'errors'	Average of FOBJ evaluations numbers	Average number of moves inside the tabu list
5	0·0076	1636	47
6	0·0074	1401	49
7	0·0066	1478	55
8	0·0068	1618	58
9	0·0060	1598	64
10	0·0062	1594	65
11	0·0064	1485	68
12	0·0071	1392	66

Sensitivity of the cardinal m of the tabu list. We let vary the cardinal m of the tabu list in the interval [5, 12] defined by Glover. Results obtained for functions G and R_2, in the case $k = 5$ and $\varepsilon = 0{\cdot}01$, are reported in Tables V and VI. Results do not vary a lot according to the value of m. The optimal value of m, which depends on the objective function, is about 5–7, because higher values are CPU time costly: so the magic number 7 advocated by Glover is again pointed out.

Sensitivity of the number k of neighbours generated round the current solution. We let k vary in the interval [1, 30]. A compromise between accuracy and CPU time is achieved for the value $k = 5$. For $k > 7$, the CPU time cost is prohibitive.

Table VI. Sensitivity of the cardinal m of the tabu list for the 2-variables Rosenbrock function

Cardinal m of the tabu list	Standard deviation of final 'errors'	Average of FOBJ evaluations numbers	Average number of moves inside the tabu list
5	0·010	1616	2·8
6	0·011	1526	3·0
7	0·011	1487	3·1
8	0·011	1525	3·3
9	0·011	1492	3·4
10	0·011	1450	3·5
11	0·012	1410	3·7
12	0·011	1451	3·9

Table VII. Influence of the type of partitioning for three test functions: GP: geometrical partitioning, LP: linear partitioning, IP: isovolume partitioning

Test function	Type of partitioning	Standard deviation of final 'errors'	Average of FOBJ evaluations numbers
G	GP	0·008	1636
	LP	0·062	1483
	IP	0·125	1429
H	GP	0·013	528
	LP	0·017	436
	IP	0·019	660
R_2	GP	0·010	1616
	LP	0·050	1405
	IP	0·081	1560

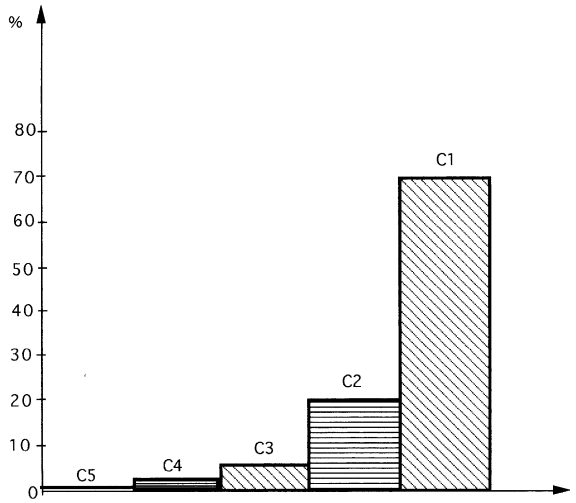

Figure 3. Frequency of position of the best neighbour in the different crowns round the current point (function R_2, $k = 5$, geometrical partitioning)

Influence of the type of partitioning. Results of tests performed for three functions and three types of partitionings are reported in Table VII. We notice that the geometrical and linear partitionings behave better than the isovolume partitioning. For the geometrical partitioning, we have shown in Figure 3 the histogram giving the frequency of position of the best neighbour in the different crowns around the current point, for the function R_2 in the case $k = 5$. Not surprisingly, the best neighbour is more often found in the most internal crowns. The histogram of Figure 3 suggests to adapt the probability of selection of the different crowns, according to their efficiency, estimated in the course of the optimization process.

4. CONCLUSION

We have proposed an implementation of Tabu Search adapted to the optimization of functions depending on continuous variables. First experimental results are encouraging for functions of two or three variables. A lot of work must still be performed to exhibit an optimal, problem-independent, tuning of the parameters of the method and to improve the efficiency in the case of high dimension problems. As indicated above, most of the classical refinements of TS can be incorporated in our algorithm: the design of a more sophisticated version of our continuous TS algorithm is in progress, including prospects of 'intensification', 'diversification', 'aspiration' and 'dynamic tabu list management' techniques.

REFERENCES

1. F. Glover, 'Heuristics for integer programming using surrogate constraints', *Dec. Sci.*, **8,** 156 –166, (1977).
2. F. Glover, 'Future paths for integer programming and links to Artificial Intelligence', *Comput. Oper. Res.*, **13,** N° 5, pp. 533–549, 1986.
3. A. Hertz, D. De Werra, 'Tabu Search techniques: a tutorial and an application to Neural Networks', *OR Spektrum*, **11,** 131–141 (1989).
4. F. Glover and M. Laguna, 'Tabu Search', in Colin R. Reeves (ed.), *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell Scientific Publications, Oxford, 1993, pp. 70–150.
5. N. Dubois and D. De Werra, 'Epcot: an efficient procedure for coloring optimally with Tabu Search', *Comput. Math. Appl.*, **25,** 35–45 (1993).
6. J. Skorin-Kapov, 'Tabu Search applied to the quadratic assignment problem', *ORSA J. Comput.*, **2,** 33–41 (1990).
7. A. Bland and G. P. Dawson, 'Tabu Search and design optimization', *IEEE Trans. C.A.D.*, **23,** 195–201 (1991).
8. S. Amellal and B. Kaminska, 'Scheduling algorithm in data path synthesis using the Tabu Search technique', *IEEE Proc. EDAC-EUROASIC'93 Conf.*, Paris, 1993, pp. 398–402.
9. N. Hu, 'Tabu Search method with random moves for globally optimal design', *Int. j. numer. methods eng.*, **35,** 1055–1070 (1992).
10. L. C. W. Dixon and G. P. Szegö (eds.), *Towards Global Optimization 2*, North-Holland, Amsterdam, 1978.
11. K. Schittkowski and W. Hock, 'More test examples for nonlinear programming codes', Lecture Notes in Economics and Mathematical Systems, Vol. 282, Springer, Berlin, 1987.
12. G. Berthiau and P. Siarry, 'An enhanced simulated annealing algorithm for extracting electronic component model parameters', *Adv. Eng. Software*, **18,** 171–176 (1993).

