ELSEVIER

European Journal of Operational Research 103 (1997) 198-208

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

Theory and Methodology

A tabu search Hooke and Jeeves algorithm for unconstrained optimization

K.S. Al-Sultan *, M.A. Al-Fawzan

Department of Systems Engineering, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia

Received 26 June 1995; accepted 13 August 1996

Abstract

This paper addresses the problem of finding the global minimum of a nonconvex function. A new hybrid algorithm for this problem based on tabu search is developed. It is hybrid in the sense that search directions are generated using tabu search strategy and then they are used in an optimization algorithm. The algorithm is tested on some standard test functions and its performance is compared with existing algorithms. Computational results show that the proposed algorithm is very efficient and robust. © 1997 Elsevier Science B.V.

Keywords: Global optimization; Tabu search; Random search directions

1. Introduction

Global optimization problems arise in many practical engineering problems. These optimization problems have the feature that the objective function may not necessarily be convex and therefore may possess many local minima in the region of interest. For applications of global optimization, the reader is referred to Törn and Žilinskas (1989).

A standard global optimization problem can be defined as follows:

Given a continuous function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$, *find a point* $x^* \in \mathbb{R}^n$ *satisfying* $f(x^*) \leq f(x)$ *for all* $x \in \mathbb{R}^n$.

Classical nonlinear programming algorithms including derivative-based (e.g., the steepest descent method, Newton's method, the method of conjugate gradients) and derivative-free methods (e.g., Rosenbrock's method, Hooke and Jeeves' method, Nelder and Mead's method) have not been successful in solving these problems. For more details on these methods, see Bazaraa et al. (1993) and Rekalitis et al. (1983). In general, these methods converge to a stationary point for which there is no guarantee of even local optimality.

There have been attempts by several researchers to develop global algorithms for nonconvex function minimization problems. For example, Corana et al. (1987) have developed a simulated annealing-based algorithm

* E-mail: alsultan@ccse.kfupm.edu.sa.

PII S0377-2217(96)00282-2

which handles multimodal functions, and provides the global minimum irrespective of the initial point in most of the cases. However, this algorithm is very costly in terms of the number of function evaluations needed to obtain the solution even though it performs well in terms of the quality of the solution. Aluffi-Pentini et al. (1985) presented a simulated annealing algorithm which follows the paths of a system of stochastic differential equations. This method found the global minimum for all test functions that were used. However, both methods require a large number of function evaluations. Recently, Dekkers and Aarts (1991) proposed a simulated annealing algorithm which outperformed the one of Aluffi-Pentini et al. (1985).

Genetic algorithms have also been proposed in the literature for global optimization. For example, see Hajela (1990) and Pham and Yang (1993). These algorithms use genetic techniques on the solution space of the problem. This is done by defining an injective mapping from a pre-defined set to the solution space in $\mathbb{R}^n$. The pre-defined set is a collection of binary vectors, each one representing a point in the solution space, and the genetic algorithm is used to go from one point to another in the solution space. A complete representation of the solution space is not possible using these algorithms, as all real variables cannot be represented completely by a pre-defined set of binary vectors. Hussien and Al-Sultan (1994) have solved this problem, and proposed generating search directions (rather than solution points) using genetic algorithms. These directions are used in Hooke and Jeeves' algorithm. Their algorithm is very efficient.

There are very few algorithms that use tabu search. Hu (1992) developed a tabu search algorithm particularly suited for optimal engineering design. His algorithm assumes that each variable of the problem is bounded by a known closed interval. A set H of different steps of each interval is computed. Then the algorithm will generate random neighbors from the current point which are contained in a ball of radius $h_i \in H$. The best of these neighbors is selected and its corresponding h_i stored in the tabu list. In his experiments, Hu tested his algorithm with one- and two-dimensional problems only. He compared it with random search and with the genetic algorithm of Hajela (1990). Hu's algorithm requires 2^n possible random moves to be examined for the general problems of n variables.

In this paper, we present a new tabu search based algorithm for solving the above discussed problem which is more efficient than the most competitive algorithms in the literature, i.e., it requires less function evaluations. This algorithm has two features. First, the algorithm resembles Hooke and Jeeves' algorithm (Bazaraa et al., 1993) in the sense that it goes through an exploratory search and pattern search. Second, the algorithm generates random search directions and performs a line search on each direction and the direction of the best point is stored in the tabu list. That is why we call it a hybrid tabu search algorithm. The algorithm shares some spirit with that of Hussien and Al-Sultan (1994) in the sense that it generates directions and uses them in an optimization algorithm. However, we use tabu search rather than genetic algorithm.

The paper is organized as follows: in Section 2, we give a brief introduction to tabu search. We present and state the proposed algorithm in Section 3. Computational results and discussion are presented in Section 4. Finally, conclusions are given in Section 5.

2. Tabu search

Tabu search is a metaheuristic that guides local heuristic search procedures to explore the solution space beyond local optimality. It was introduced by Glover (1986, 1989, 1990) specifically for combinatorial problems. Its basic ideas have also been also proposed by Hansen (1986) and Hansen and Jaumard (1987) with another name "steepest ascent mildest descent". Since then, tabu search has been applied to a wide range of problem settings in which it has consistently found better solutions than methods previously applied to these problems. For example, tabu search has been applied to flow shop scheduling (Widmer and Hertz, 1989; Taillard, 1990), architectural design (Bland and Dawson, 1991), time tabling problem (Hertz, 1991), among others.

The tabu search starts at some initial point and then moves successively among neighboring points. At each iteration, a move is made to the best point in the neighborhood of the current point which may not be an

improving solution. The method forbids (makes tabu) points with certain attributes with the goals of preventing cycling and guiding the search towards unexplored regions of the solution space. This is done using an important feature of the tabu search method called *tabu list*. A tabu list consists of the latest moves made so that recently visited points are not generated again. The size of the tabu list can be either fixed or variable. In its simplest form, tabu search requires the following ingredients:

- Initial point.
- Mechanism for generating some neighborhood of the current point.
- Tabu list.
- Aspiration criterion.
- Stopping criterion.

For more complete description of this method, the interested reader can refer to the papers of Glover (1989, 1990).

3. The proposed algorithm

As explained in Section 1, our algorithm uses the optimization technique of Hooke and Jeeves where the directions are generated randomly and a line search is performed along each generated direction to determine the optimal step length. Then the best nontabu improving point (or best nontabu if no improving point is found) is selected and its associated direction is stored in the tabu list. This procedure is repeated and controlled by tabu search.

Specifically, the algorithm starts at some point, say x_1, and it goes through several iterations. At each iteration k, the algorithm goes through m exploratory searches (cycles) and one pattern search. The point x_k of each iteration becomes the starting point z_1 for the m cycles resulting in the points $z_2, z_3, \ldots, z_{m+1}$. In each cycle, r directions are generated randomly, and a line search is performed along each direction. The direction that gives the minimum functional value is selected provided that it is nontabu or it is tabu and it improves on the best solution found so far. Then, this direction is stored in the *tabu list*. A direction is said to be tabu if it is the negative of any direction in the *tabu list*, otherwise it is said to be nontabu. After m cycles, the algorithm performs a line search along the direction $z_{m+1} - z_1$ to generate the next point x_{k+1} and this constitutes the pattern step. If $k = MAXITER$, where $MAXITER$ is the maximum allowed number of the iterations, the algorithm stops; otherwise the algorithm goes through iteration $k + 1$ starting from the point x_{k+1}. Next, we formally present the proposed algorithm.

Statement of the algorithm

Initialization step

Choose r (the number of random search directions to be used in each cycle). Choose m (the number of cycles to be performed in each iteration). Choose a suitable size for the *tabu list*, TL. Choose $MAXITER$ (the maximum number of iterations). Choose ϵ (the desired accuracy of the percentage of improvement between two consecutive iterations). Choose a starting point x_1. Let $z_1 = x_1$.

Let $TL = \phi$, $BFV = f(x_1)$.

Let $k = j = 1$, and go to the main step.

Main step

(1) *Perform m cycles*

(1.1) Generate r different random directions, $d_1, d_2, \ldots, d_r$ (see Section 3.1).

Let λ^* and d^* be such that

$$f(z_j + \lambda^* d^*) = \min_{1 \le i \le r} f(z_j + \lambda_i d_i)$$

(or d^* is the best direction in this cycle with respect to the generated directions, and λ^* is the corresponding optimal step length as discussed in Section 3.2).

(1.2) *Check tabu status*

(1.2.1) $l = 1$

(1.2.2) If ($d^* \notin TL$) or ($d^* \in TL$ and $f(z_j + \lambda^* d^*) < BFV$) then go to step (1.3); otherwise, replace l by $l + 1$. Let the lth best direction (step length) be d^* (λ^*) (i.e. this is the best direction among all generated directions in this cycle excluding those considered earlier in this step) and repeat this step.

(1.3) *Update current point*

Let $z_{j+1} = z_j + \lambda^* d^*$.

Store d^* in TL and update it accordingly (see Section 3.3).

If $f(z_j + \lambda^* d^*) < BFV$ then $BFV = f(z_j + \lambda^* d^*)$.

(1.4) If $j = m$ go to step (2); otherwise, replace j by $j + 1$ and go to step (1.1).

(2) *Perform pattern search*

Let $d = z_{m+1} - z_1$. Let $\bar{\lambda}$ be an optimal solution to the problem

$$\min_{\lambda \in R} f(z_{m+1} + \lambda d)$$

(see Section 3.2).

Let $x_{k+1} = z_{m+1} + \bar{\lambda} d$, and go to step (3).

(3) *Check stopping criterion*

If either $f(x_k)$ or $f(x_{k+1})$ is equal to zero, let $improv = 1$. If both are equal to zero, let $improv = 0$.

If neither $f(x_k)$ nor $f(x_{k+1})$ is equal to zero then let $improv = |(f(x_{k-1}) - f(x_k))/f(x_k)|$.

If $k = MAXITER$ or $improv \leq \epsilon$, stop; otherwise, let $j = 1$, and $z_1 = x_{k+1}$, and replace k by $k + 1$, and go to step (1).

3.1. Generation of random search directions

In our experiments, we used the following scheme for generating the random search directions. Let d_i be the ith component of the direction d, where $1 \leq i \leq n$ and n is the dimension of the problem. To generate a direction, perform the following steps:

(1) $i = 1$

(2) Let $rand \sim U[0, 1]$. *rand* is a random number uniformly distributed between 0 and 1.

(3) $$d_i = \begin{cases} -1 & \text{if } 0 \leq rand \leq 1/3, \\ 0 & \text{if } 1/3 < rand \leq 2/3, \\ 1 & \text{if } 2/3 < rand \leq 1. \end{cases}$$

(4) If $i = n$, stop; otherwise let $i = i + 1$ and go to step (2).

3.2. The line search scheme

Many line search schemes exist in the literature. However, all of them assume that the function is unimodal which is not usually encountered in global optimization problems. We have tried many of these, but unfortunately they failed. Therefore, we propose to use an exhaustive line search scheme which decides first on the general location of the minimum along the direction and then do a more fine search in the vicinity of the minimum.

We used the following line search scheme in our experiments. Let $[a, b]$ be the interval of uncertainty on which the line search is to be performed. Let x and d be the current point and direction respectively. Given the parameters of the scheme δ, k, s, and δ_f, perform the following steps:

(1) $\lambda^* = \lambda = a$, $min = f(x + \lambda d)$.

(2) Let $\lambda = \lambda + k$. If $\lambda > b$, go to step (4); otherwise go to step (3).
(3) If $f(x + \lambda d) < min$, then $min = f(x + \lambda d)$, and $\lambda^* = \lambda$. Go to step (2).
(4) $a = \lambda^* - \delta$, $b = \lambda^* + \delta$, $min = f(x + \lambda^* d)$, $\lambda^* = \lambda = a$, $k = k/s$.
(5) Let $\lambda = \lambda + k$. If $\lambda > b$, go to step (7); otherwise go to step (6).
(6) If $f(x + \lambda d) < min$, then $min = f(x + \lambda d)$, and $\lambda^* = \lambda$. Go to step (5).
(7) Let $\delta = \delta/s$. If $\delta \leq \delta_f$, stop; otherwise go to step (4).

λ^* is the optimal step length. However, one has to qualify this statement by the fact that the accuracy and reliability of the line search scheme are controlled by the constants δ, k, s, and δ_f. Both reliability and accuracy can be enhanced by finding the best values of these parameters by parametric study which could be at the expense of more function evaluations. Hence, a balance between reliability and accuracy on one hand and the number of function evaluations on the other hand is sought.

3.3. Storing in the tabu list

In our algorithm, the chosen direction has to be stored in the tabu list, *TL*, in step (1.3). *TL* has a fixed size. Each time a new element needs to be stored in *TL*, the oldest element is removed form *TL* and the new element is appended at the end of *TL*. We store in *TL* the negative of the chosen direction.

4. Computational results and discussion

In this section, we discuss various implementation details and results of our computational experiments.

4.1. Parameter setting

Tabu search is a parameter sensitive technique similar to simulated annealing and genetic algorithms. We have conducted an extensive parametric study for the proposed algorithm on the test problems in Section 4.2. The results of this study show that the following two sets of parameters give the best performance:

- *Set 1*: A compromise between solution quality and number of function evaluations.
  $r = 2n$;
  $m = 4$;
  tabu list size = 20;
  MAXITER = 2;
  $\epsilon = 10^{-4}$.
- *Set 2*: A very good solution quality with probably more function evaluations and more computation time.
  $r = n^2$;
  $m = 10$;
  tabu list size = 25;
  MAXITER = 5;
  $\epsilon = 10^{-10}$.

In our experiments we used *Set 1*.

We have also conducted an extensive parametric study for the line search scheme on the test problems in Section 4.2 and we have found that $k = 1$, $\delta = 0.5$, $s = 10$, and $\delta_f = 0.05$ work very well for the test problems given in Section 4.1.

Table 1
Methods used for the first experiment

Method	Name	Reference
A	Multistart	Rinnooy Kan and Timmer (1984)
B	Controlled Random Search	Price (1978)
C	Density Clustering	Törn (1978)
D	Clustering with distribution function	De Biase and Frontini (1978)
E	Multi Level Single Linkage	Rinnooy Kan and Timmer (1987)
F	Simulated Annealing	Dekkers and Aarts (1991)
G	Simulated Annealing based on stochastic differential equations	Aluffi-Pentini et al. (1985)
TS	Tabu Search	This paper

Table 2
Methods used for the second experiment

Method	Name	Reference
Simplex	Simplex method	Nelder and Mead (1964)
ARS	Adaptive Random Search	Masri et al. (1980)
SA	Simulated Annealing	Corana et al. (1987)
GA	Genetic Algorithm	Hussien and Al-Sultan (1994)

4.2. Test problems

We test our algorithm using a set of test functions known from the literature. These test functions are taken from Dixon and Szegö (1978), see Appendix A. We perform two experiments. In the first experiment, we compare our algorithm with the methods shown in Table 1.

In the second experiment, we compare it with the methods shown in Table 2 using the Rosenbrock test function in 2 and 4 dimensions (see Appendix B).

Since all the algorithms shown in Table 1 are tested on different machines, we use the standard unit of time as shown in Dixon and Szegö (1978) which make comparisons machine-independent. One unit of standard time is equivalent to the running time needed for 1000 evaluations of the Shekel 5 function using the point (4,4,4,4).

4.3. Results and discussion

Table 3 shows the number of function evaluations for each method. In Table 4, the computation times in units of standard time for each method are given. For methods A–G, numbers of function evaluations and computation times are taken from Dekkers and Aarts (1991). Table 4 shows no results for method G, since the running time available is only in absolute computer time.

Note that the number of function evaluations and running time used in generating the initial random sample are not counted in many methods. For example, for the initial random sample, the Multi Level Single Linkage method (Rinnooy Kan and Timmer, 1987) uses 1000 function evaluations and the Simulated Annealing method (Dekkers and Aarts, 1991) uses $10n$ function evaluations.

Tables 3 and 4 show that the results of our algorithm are very encouraging. Note that our algorithm never failed in finding the global minimum in all the tested problems, while methods E and F failed in problems S7 and S5, respectively, and our algorithm is still competitive in terms of efficiency.

Table 3
Number of function evaluations of the first experiment

Function Method	GP	BR	H3	H6	S5	S7	S10
A	4400	1600	2500	6000	6500	9300	11000
B	2500	1800	2400	7600	3800	4900	4400
C	2499	1558	2584	3447	3649	3606	3874
D	378	597	732	807	620	788	1160
E	148	206	197	487	404	432[a]	564
F[c]	563	505	1459	4648	365[a]	558	797
G[c]	5349	2700	3416	3975	2446	4759	4741
TS[b]	281	398	578	2125	753	755	1203

[a] The global minimum was not found in one of four runs.
[b] The average number of function evaluations of four runs.
[c] The number of function evaluations of the initial sampling are not included.

Table 4
Running time in units of standard time of the first experiment

Function Method	GP	BR	H3	H6	S5	S7	S10
A	4.5	2	7	22	13	21	32
B	3	4	8	46	14	20	20
C	4	4	8	16	10	13	15
D	15	14	16	21	23	20	30
E[c]	0.15	0.25	0.5	2	1	1[a]	2
F[c]	0.9	0.9	5	20	0.8[a]	1.5	2.7
TS[b]	0.22	0.3	1.02	7.2	1.47	1.58	3.34

[a] The global minimum was not found in one of four runs.
[b] The average running time of four runs.
[c] The running time of the initial sampling was not counted.

We observe that the quality of the solution, number of function evaluations, and running time are very much dependent on the initial starting point. Thus, the comparisons provided above are not entirely fair since the different methods may start at a different initial point. In addition, the type of language used to code each method, the data structure used, and the machine on which each method is tested, are also influencing factors for execution time.

Another important point is that the first and second derivatives of each of the tested functions can be easily obtained. Hence, methods which utilize these tools have an advantage over other methods. Unlike other methods, our algorithm does not require the derivatives of the function. Moreover, it can handle functions that are not differentiable and functions that are not even explicit.

For the second experiment, Tables 5 and 6 show comparisons of our algorithm with the algorithms shown in Table 2 using the Rosenbrock function in 2 and 4 dimensions. Results are taken from Corana et al. (1987) and Hussien and Al-Sultan (1994). Computation times are not reported in Corana et al. (1987), and we relied on function evaluations to measure efficiency. Our algorithm outperformed previous methods in terms of number of function evaluations except the Simplex method which fails to get the global solution in some cases. Table 6 shows that our algorithm is reliable.

Table 5
Number of function evaluations of the second experiment

Method Starting Point	Simplex	ARS	SA	GA	TS
2 dimensions:					
1001,1001	993	3411	500001	2389	1954
1001,-999	276	131841	508001	2214	1762
-999,-999	730	15141	524001	3254	2483
-999,1001	907	3802	484001	3412	2671
1443,1	907	181280	492001	2115	1616
1,1443	924	2629	512001	5781	4121
1.2,1	161	6630	488001	1548	1195
4 dimensions:					
101,101,101,101	1869	519632	1288001	228534	16528
101,101,101,-99	784	194720	1328001	213422	27940
101,101,-99,-99	973	183608	1264001	264521	13995
101,-99,-99,-99	1079	195902	1296001	299321	11443
-99,-99,-99,-99	859	190737	1304001	44567	14007
-99,101,-99,101	967	4172290	1280001	234512	16572
101,-99,101,-99	870	53878	1272001	193134	11471
201,0,0,0	1419	209415	1288001	182131	25413
1,201,1,1	1077	215116	1304001	283946	8884
1,1,1,201	1265	29069006	1272001	214312	34083

Table 6
Final functional value of the second experiment

Method Starting Point	Simplex	ARS	SA	GA	TS
2 dimensions:					
1001,1001	4.9E-10	1586.4	1.8E-10	1.2E-12	7.1E-11
1001,-999	7.4E-10	8.6E-9	2.6E-9	2.3E-10	1.5E-9
-999,-999	2.7E-10	1.2E-8	1.2E-9	4.4E-12	4.2E-9
-999,1001	9.2E-10	583.2	4.2E-8	3.4E-10	6.1E-9
1443,1	5.4E-11	4.7E-10	1.5E-8	3.3E-10	7.2E-9
1,1443	2.2E-10	1468.9	1.6E-9	1.2E-10	3.4E-9
1.2,1	2.4E-10	5.5E-7	2.0E-8	2.2E-18	2.6E-10
4 dimensions:					
101,101,101,101	3.70	1.9E-6	5.0E-7	4.8E-9	5.1E-8
101,101,101,-99	5.46E-17	1.7E-6	1.8E-7	2.1E-9	1.6E-7
101,101,-99,-99	9.8E-18	3.8E-6	5.9E-7	2.2E-8	8.3E-8
101,-99,-99,-99	3.4E-17	2.3E-6	7.4E-8	3.0E-9	1.4E-7
-99,-99,-99,-99	8.3E-18	2.7E-6	3.3E-7	5.7E-9	6.3E-7
-99,101,-99,101	1.2E-17	2.6E-6	2.8E-7	3.9E-8	9.5E-7
101,-99,101,-99	6.0E-18	3.7	2.3E-7	4.1E-8	4.5E-7
201,0,0,0	3.70	1.1E-6	7.5E-7	3.0E-8	5.7E-7
1,201,1,1	9.4E-18	1.2E-6	4.6E-7	5.8E-8	7.2E-7
1,1,1,201	3.9E-17	2.2E-6	5.2E-7	4.3E-8	6.1E-7

Table A.1
H3 ($n = 3$ and $q = 4$)

i	a_{ij}			c_i	p_{ij}		
1	3	10	30	1	0.3689	0.1170	0.2673
2	0.1	10	35	1.2	0.4699	0.4387	0.7470
3	3	10	30	3	0.1091	0.8732	0.5547
4	0.1	10	35	3.2	0.03815	0.5743	0.8828

5. Conclusions

In this paper, a new algorithm for global optimization has been presented. The algorithm uses tabu search for generating search directions and then uses these directions in Hooke and Jeeves' technique. Computational results showed that our algorithm is very promising. One can extend the work by investigating other schemes for generating the random search directions and by using other elements of tabu search such as long-term memory, strategic oscillation, path relinking, and others.

Appendix A

The following test functions are taken from Dixon and Szegö (1978).

GP (Goldstein and Price)

$$f(x_1, x_2) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$$
$$\times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)],$$
$$S = \{x \in \mathbb{R}^2 \mid -2 \le x_i \le 2, i = 1, 2\},$$
$$x_{\min} = (0, -1),$$
$$f(x_{\min}) = 3.$$

There are 4 local minima.

BR (Branin)

$$f(x_1, x_2) = a(x_2 - bx_1^2 + cx_1 - d)^2 + e(1 - f)\cos(x_1) + e,$$

where $a = 1, b = 5.1/(4\pi^2), c = 5/\pi, d = 6, e = 10, f = 1/(8\pi)$. $S = \{x \in \mathbb{R}^2 \mid -5 \le x_1 \le 10$ and $0 \le x_2 \le 15\}$, $x_{\min} = (-\pi, 12.275); (\pi, 2.275); (3\pi, 2.475)$, $f(x_{\min}) = 5/(4\pi)$. There are no more minima.

H3 and H6 (Hartmann's family)

$$f(x) = -\sum_{i=1}^{q} c_i \exp\left(-\sum_{j=1}^{n} a_{ij}(x_j - p_{ij})^2\right),$$
$$S = \{x \in \mathbb{R}^n \mid 0 \le x_j \le 1, 1 \le j \le n\}.$$

These functions both have four local minima, $x_{\text{loc}} \approx (p_{i1}, \ldots, p_{in})$, $f(x_{\text{loc}}) \approx -c_i$.

Table A.2
H6 ($n = 6$ and $q = 4$)

i	a_{ij}						c_i	p_{ij}					
1	10	3	17	3.5	1.7	8	1	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.05	10	17	0.1	8	14	1.2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	3	3.5	1.7	10	17	8	3	0.2348	0.1451	0.3522	0.2883	0.3047	0.6550
4	17	8	0.05	10	0.1	14	3.2	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

Table A.3
S5, S7, S10

i	a_i				c_i
1	4	4	4	4	0.1
2	1	1	1	1	0.2
3	8	8	8	8	0.2
4	6	6	6	6	0.4
5	3	7	3	7	0.4
6	2	9	2	9	0.6
7	5	5	3	3	0.3
8	8	1	8	1	0.7
9	6	2	6	2	0.5
10	7	3.6	7	3.6	0.5

S5, S7 and S10 (Shekel's family)

$$f(x) = -\sum_{i=1}^{q}((x - a_i)^{\mathrm{T}}(x - a_i) + c_i)^{-1}$$

with $n = 4$, $q = 5, 7, 10$ for S5, S7, S10, respectively, $x = (x_1, \ldots, x_n)^{\mathrm{T}}$ and $a_i = (a_{i1}, \ldots, a_{in})^{\mathrm{T}}$.

$$S = \{x \in \mathbb{R}^4 \mid 0 \le x_i \le 1, 1 \le i \le 4\}.$$

These functions have 5, 7 and 10 local minima for S5 and S7 and S10, respectively, $x_{\mathrm{loc}} \approx (1/c_i, \ldots, 1/c_q)$.

Appendix B

The following Rosenbrock test function in 2 and 4 dimensions is taken from Rosenbrock (1960) and Corana et al. (1987):

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 - (1 - x_1)^2,$$

$$f(x_1, x_2, x_3, x_4) = \sum_{i=1}^{3} 100(x_{i+1} - x_i^2)^2 - (1 - x_i)^2.$$

References

[1] Aluffi-Pentini, F., Parisi, V. and Zirilli, F. (1985), "Global optimization and stochastic differential equations", *Journal of Optimization Theory and Applications* 47, 1–16.

[2] Bazaraa, M.S., Sherali, H. and Shetty, C.M. (1993), *Nonlinear Programming: Theory and Algorithms*, John Wiley, Second Edition, New York.

[3] Bland, J.A., and Dawson, G.P. (1991), "Tabu search and design optimization", *Computer-aided Design* 23/3, 195–201.

[4] Corana, A., Marchesi, M., Martini, C. and Ridella, S. (1987), "Minimizing multimodal functions of continuous variables with the Simulated Annealing algorithm", *ACM Transactions Mathematical Software* 13/3, 262–280.

[5] De Biase, L. and Frontini, F. (1978), "A stochastic method for global optimization: its structure and numerical performance", in: L.C.W. Dixon, and G.P. Szegö, eds., *Towards Global Optimization* 2, North-Holland, Amsterdam, 85–102.

[6] Dekkers, A., and Aarts, E. (1991), "Global Optimization by Simulated Annealing", *Mathematical Programming* 50, 367–393.

[7] Dixon, L.C.W. and Szegö, G. P. (1978), "The global optimization problem: an introduction", in: L.C.W. Dixon and G.P. Szegö, eds., *Towards Global Optimization* 2, North-Holland, Amsterdam, 1–15.

[8] Glover, F. (1986), "Future paths for integer programming and linkage to artificial intelligence", *Computers & Operations Research* 13, 533–549.

[9] Glover, F. (1989), "Tabu Search-Part I", *ORSA Journal on Computing* 1, 190–206.

[10] Glover, F. (1990), "Tabu Search-Part II", *ORSA Journal on Computing* 2, 4–32.

[11] Hansen, P. (1986), "The steepest ascent mildest descent heuristic for combinatorial programming", *Congress on Numerical Methods in Combinatorial Optimization*, Capri, Italy.

[12] Hansen, P. and Jaumard, B. (1987), "Algorithms for the maximum satisfiability problem", RUTCOR Research Report RR#43-87, Rutgers, New Brunswick, NJ.

[13] Hajela, P. (1990), "Genetic Search-An approach to the nonconvex optimization problem", *American Institute of Aeronautics and Astronautics Journal* 28/7, 1205–1210.

[14] Hertz, A. (1991), "Tabu search for large scale time tabling problems", *European Journal of Operational Research* 51, 39–47.

[15] Hu, N. (1992), "Tabu Search Method with Random Moves for Globally Optimal Design", *International Journal for Numerical Methods in Engineering* 35, 1055–1070.

[16] Hussien, M.F., and Al-Sultan, K.S. (1994), "A Global Algorithm for Nonconvex Function Minimization", submitted.

[17] Masri, S.F., Bekey, G.A. and Safford, F.B. (1980), "A Global Optimization Algorithm using Adaptive Random Search", *Applied Mathematics and Computation* 7, 353–375.

[18] Nelder, J.A., and Mead, R. (1964), "A Simplex Method for Function Minimization", *Computer Journal* 7, 308–313.

[19] Pham, D.T., and Yang, Y. (1993), "Optimization of multi-modal discrete functions using genetic algorithms", *Proceedings of the Institution of Mechanical Engineers* 207, 53–59.

[20] Price, W.L. (1978), "A controlled random search procedure for global optimization", in: L.C.W. Dixon and G.P. Szegö, eds., *Towards Global Optimization* 2, North-Holland, Amsterdam, 71–84.

[21] Rekalitis, G.V., Ravindran, A. and Ragsdell, K.M. (1983), *Engineering Optimization: Methods and Applications*, John Wiley, New York.

[22] Rinnooy Kan, A.H.G., and Timmer, G.T. (1984), "Stochastic methods for global optimization", *American Journal of Mathematical and Management Sciences* 4, 7–40.

[23] Rinnooy Kan, A.H.G., and Timmer, G.T. (1987), "Stochastic Global Optimization Methods Part II: Multi Level Methods", *Mathematical Programming* 39, 57–78.

[24] Rosenbrock, H. (1960), "An automatic method for finding the greatest or least value of a function", *Computer Journal* 3, 175–184.

[25] Taillard, E. (1990), "Some efficient heuristic methods for the flow shop sequencing problem", *European Journal of Operational Research* 47, 65–74.

[26] Törn, A. (1978), "A search-clustering approach to global optimization", in: L.C.W. Dixon and G.P. Szegö eds., *Towards Global Optimization* 2, North-Holland, Amsterdam, 49–62.

[27] Törn, A. and Žilinskas, A. (1989), "Global Optimization", Lecture Notes in Computer Science 350, Springer.

[28] Widmer, M., and Hertz, A. (1989), "A new heuristic method for the flow shop sequencing problem", *European Journal of Operational Research* 41, 186–193.