Journal of Global Optimization (2005) 31: 635–672
DOI 10.1007/s10898-004-9972-2

A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems

M. MONTAZ ALI[1], CHAROENCHAI KHOMPATRAPORN[2] and ZELDA B. ZABINSKY[2]
[1]*School of Computational and Applied Mathematics, Witwatersrand University, Private Bag-3, Wits-2050, Johannesburg, South Africa (e-mail: mali@cam.wits.ac.za)*
[2]*Industrial Engineering, University of Washington, Box 352650, Seattle, WA 98195-2650, USA (e-mails: ckhom@u.washington.edu, zelda@u.washington.edu)*

Abstract. There is a need for a methodology to fairly compare and present evaluation study results of stochastic global optimization algorithms. This need raises two important questions of (i) an appropriate set of benchmark test problems that the algorithms may be tested upon and (ii) a methodology to compactly and completely present the results. To address the first question, we compiled a collection of test problems, some are better known than others. Although the compilation is not exhaustive, it provides an easily accessible collection of standard test problems for continuous global optimization. Five different stochastic global optimization algorithms have been tested on these problems and a performance profile plot based on the improvement of objective function values is constructed to investigate the macroscopic behavior of the algorithms. The paper also investigates the microscopic behavior of the algorithms through quartile sequential plots, and contrasts the information gained from these two kinds of plots. The effect of the length of run is explored by using three maximum numbers of function evaluations and it is shown to significantly impact the behavior of the algorithms.

Key words: Controlled random search, Differential evolution, Empirical comparison of algorithms, Genetic algorithm, Global optimization, Hide-and-Seek, Improving Hit-and-Run, Performance profile and Test problems, Population set based global optimization, Simulated annealing

1. Introduction

Problems in business, medicine, engineering, and applied sciences nowadays are more and more complex. In mathematical terminology, these problems are no longer linear, quadratic, nor unimodal. The domains of the problems could be nonconvex and disconnected. Their objective functions are often multimodal with peaks, valleys, channels, and flat hyperplanes of different heights. Solving these types of problems, which are classified as global optimization problems, to optimality undoubtedly becomes a true challenge. In many cases, just estimating the total number

of local optima on this rugged objective function terrain is very difficult, while finding and enumerating all the local optima may be completely impractical. Stochastic global optimization techniques offer alternatives to address these difficult issues. Instead of finding and comparing all the local optima to identify the global optimal solution, stochastic global optimization algorithms sample and evaluate candidate solutions and approximate the global optimum (or at least one of the global optima). This means that the final solution obtained from stochastic global optimization methods converge to the global optimum in a probabilistic sense.

A variety of recently proposed stochastic global optimization algorithms show potential to solve complex problems but their relative merits are difficult to analyze systematically. Typically after an algorithm has been developed, it is tested against a set of problems. The set of test problems of one algorithm is often selected haphazardly and differs from that of another algorithm. The differences in test problem sets may induce bias toward particular algorithms when comparing their performances. Moreover, analyzing a large set of data collected in order to gain insights on how the algorithms behave is in itself another challenge.

In this paper, we propose a collection of test problems that can be used as benchmark problems. Although this collection is not exhaustive, it could be used as a starting point when one needs some test problems. Previous attempts to assemble global optimization test problems include the two collections by Floudas and Pardalos (1990) and by Floudas et al. (1999). Other on-line collections of test problems also exist. These include the CUTE test collection by Gould et al. (2001), the GLOBAL library at GAMS World (2002), and the COCONUT benchmark maintained by Neumaier (2003a). Notice though that the CUTE collection primarily consists of local optimization problems, and the COCONUT benchmark focuses on constrained global optimization problems. The problems in this paper, besides the box constraints, are mainly unconstrained multimodal optimization problems. Nonetheless, the two volumes together with the on-line collections and the problems in this paper should offer a sufficiently large set of test problems. Some of the problems in this paper are inspired by real world applications. For example, the problem (36) listed in Table 1 is called the Transistor Modelling problem. The function provides a least-squares approach to the solution of a set of nine simultaneous nonlinear equations, which arise in the context of Transistor Modelling (Price, 1983). The sinusoidal problem (48) in Table 1 was constructed to retain characteristics that arise in aircraft composite structural design problems (Zabinsky et al., 1992), yet properties of the constructed problem are known, e.g. the global optimum and the number of local optima. The sinusoidal problem has a few parameters that can be adjusted to change the relative heights of the local optima, the location of the global optimum, and the number of variables. Similarly, Storn's

Table 1. Fifty test problems

	Name of the problem	Dimension n
1	Ackley's Problem (ACK)	10
2	Aluffi-Pentini's Problem (AP)	2
3	Becker and Lago Problem (BL)	2
4	Bohachevsky 1 Problem (B1)	2
5	Bohachevsky 2 Problem (B2)	2
6	Branin Problem (BR)	2
7	Camel Back-3 Three Hump Problem (CB3)	2
8	Camel Back-6 Six Hump Problem (CB6)	2
9	Cosine Mixture Problem (CM)	2, 4
10	Dekkers and Aarts Problem (DA)	2
11	Easom Problem (EP)	2
12	Epistatic Michalewicz Problem (EM)	5
13	Exponential Problem (EXP)	10
14	Goldstein and Price Problem (GP)	2
15	Griewank Problem (GW)	10
16	Gulf Research Problem (GRP)	3
17	Hartman 3 Problem (H3)	3
18	Hartman 6 Problem (H6)	6
19	Helical Valley Problem (HV)	3
20	Hosaki Problem (HSK)	2
21	Kowalik Problem (KL)	4
22	Levy and Montalvo 1 Problem (LM1)	3
23	Levy and Montalvo 2 Problem (LM2)	5, 10
24	McCormick problem (MC)	2
25	Meyer and Roth Problem (MR)	3
26	Miele and Cantrell Problem (MCP)	4
27	Modified Langerman Problem (ML)	10
28	Modified Rosenbrock Problem (MRP)	2
29	Multi-Gaussian Problem (MGP)	2
30	Neumaier 2 Problem (NF2)	4
31	Neumaier 3 Problem (NF3)	10
32	Odd Square Problem (OSP)	10
33	Paviani's Problem (PP)	10
34	Periodic Problem (PRD)	2
35	Powell's Quadratic Problem (PQ)	4
36	Price's Transistor Modelling Problem (PTM)	9
37	Rastrigin Problem (RG)	10
38	Rosenbrock Problem (RB)	10
39	Salomon Problem (SAL)	5, 10
40	Schaffer 1 Problem (SF1)	2
41	Schaffer 2 Problem (SF2)	2
42	Shubert Problem (SBT)	2
43	Schwefel Problem (SWF)	10
44	Shekel 5 Problem (S5)	4
45	Shekel 7 Problem (S7)	4
46	Shekel 10 Problem (S10)	4
47	Shekel's Foxholes Problem (FX)	5, 10
48	Sinusoidal Problem (SIN)	10, 20
49	Storn's Tchebychev Problem (ST)	9, 17
50	Wood's Problem (WP)	4

Tchebychev problem (49) in Table 1 arises in the analog or digital filter design in electrical engineering. The global minimum of this problem is located at the origin but it is very difficult to locate the minimizer. Many of the other problems included in the table are well-known in the literature for testing genetic or population set based algorithms.

The global optimization problems in this paper follow the form:

$$\text{minimize } f(x) \text{ subject to } x \in \Omega$$

where x is a continuous variable vector with domain $\Omega \subset \mathbb{R}^n$, and $f(x) : \Omega \mapsto \mathbb{R}$ is a continuous real-valued function. The domain Ω is defined within upper and lower limits of each dimension. We notate the global optimal solution x^*, with its corresponding global optimal function value $f(x^*)$ or f^* for a short hand notation.

Five stochastic global optimization algorithms are used as demonstrative algorithms. They are either the simulated annealing (SA) type or the population set based type. These two types of algorithms are used extensively in practice. The first two algorithms are *Improving Hit-and-Run* (IHR) (Zabinsky, et al., 1993) and *Hide-and-Seek* (HNS) (Romeijn and Smith, 1994). These algorithms are of the simulated annealing type. The other three algorithms are *Controlled Random Search* (CRS), *Real Coded Genetic Algorithm* (GA), and *Differential Evolution* (DE), which are of the population set based type (Ali and Törn, 2004). More details on these algorithms can be found in Appendix A. Our goal is to gain insights on the behaviors and to provide a methodology to compare these two types of stochastic algorithms. All five algorithms are tested on the same set of problems which are detailed in Appendix B. Specific experiment and data collection procedures of these algorithms will be discussed in the next section.

To compactly and comprehensively represent the data collected from the five illustrative algorithms, we propose a modified version of the performance profile suggested by Dolan and Moré (2002). The modified performance profile is applicable to stochastic algorithms and uses the relative value of objective function as the comparison basis, instead of CPU time as in the original Dolan and Moré's performance profile. Different run lengths (maximum numbers of function evaluations) are used to construct different modified performance profiles. The varying run lengths allow a comparison between computation and accuracy of solutions. A discussion on the effect of the run lengths is presented later in the paper. Although the performance profile can capture a macroscopic behavior among the five demonstrative algorithms, it does not encapsulate a microscopic behavior of the algorithms. The microscopic behavior of the algorithms is investigated through quartile sequential plots. The quartile sequential plots of some example problems are included in the paper to illustrate the use of the plots.

The organization of the paper is as follows. The next section details the methodology procedures on the experiment used to obtain the performance data of the five demonstrative algorithms, and lists the benchmark problems in the test problem set. Section 3 summarizes the modified performance profile which is based on the relative improvement of objective function values, and presents the comparative study results. Section 4 concludes with final remarks. Appendix A describes the five stochastic global optimization algorithms used to illustrate the methodology in more details. And finally Appendix B provides the mathematical formulations for all the benchmark test problems.

2. Experimental Methodology and Test Problems

A major part of the experimental procedure of this comparative study is a result of a discussion among a group of participants at the Stochastic Global Optimization workshop held in New Zealand, June 2001. The authors thank all of the participants. This experimental procedure is intended to establish a common ground to fairly compare stochastic algorithms which shall be run on different platforms.

2.1. METHODOLOGY

As discussed in the companion paper (Khompatraporn et al., 2005), there are many measures of merit for algorithms. These measures can be grouped into four categories:

1. *General applicability*: Dimensionality, number of local optima.
2. *Efficiency*: Time complexity, space complexity, analytic complexity, algebraic complexity, theoretical complexity, order of convergence.
3. *Trustworthiness*: Accuracy, success ratio.
4. *Ease of use*: User friendliness.

In congruence to *general applicability*, the five demonstrative algorithms were tested on problems with different dimensions varying from 2 to 20 and different number of local optima. The number of local optima is not known in most of these problems, but is more than two in almost all cases.

Computation time, or *time complexity* of the algorithms, is an important measure. A fair comparison of computation time may be performed when the algorithms are coded using the same computer language and the experiment is conducted on the same computer platform. In addition, the total computation time also depends on the compiler, the coding skill of the programmer, and user interface routines which could be resource-intensive. Dixon and Szegö (1978) suggested the use of *standard unit time* as a comparative measure. This standard unit time would work well when the

difference is mainly the computer platform. The experiment conducted in this paper used different computer languages, platforms, and compilers, so CPU time was not considered in the comparative analysis. Instead, we used the number of function evaluations as a measure of time complexity. This is appropriate for the algorithms tested because they have similar computational overhead per function evaluation, and they do not calculate the gradient or Hessian. Thus, the function evaluation reflects the time complexity of the algorithms.

The *space complexity* of the simulated annealing type and population set based algorithms remains constant as the number of function evaluations increases, so it is not included in the performance evaluation. The *accuracy* used in the experiment yields the degree of solution accuracy up to the 5th decimal number so as to keep the experimental results realistic and comparable to the available information such as the optimal solution(s) pertaining to the test problems. The *algebraic complexity*, *analytic complexity*, and *theoretical complexity* for stochastic global optimization (SGO) algorithms as discussed in Khompatraporn et al. (2005) are not emphasized in the experiment since the focus is computational rather than theoretical. The SA type algorithms used here guarantee to converge to the optimal solution in probability. The *order of convergence* of IHR and HNS algorithms are discussed in Zabinsky et al. (1993) and Romeijn and Smith (1994), respectively, where the number function evaluations for IHR is shown to be $O(n^{5/2})$ for a class of elliptical problems, and HNS is shown to stochastically dominate pure adaptive search (PAS) under certain assumptions. Although no theoretical convergence results are known for the population set based methods considered here, empirical evidence suggests that the probability of finding the global minimum increases with the size of the population set (Ali and Törn, 2004).

One consideration in comparing algorithms is the trade-off between *efficiency* and *trustworthiness*. Törn and Žilinskas (1989) suggested a comparison by using *success ratio* which is the number of times that the algorithm successfully finds the optimum over the total number of times that the algorithm is applied, starting at random points, to the problem. But an advantage of applying the performance profile plot, which will be discussed in more detail in Section 3, is that the success ratio is implicitly embedded in the plot. The value of ρ in the performance profile (see Dolan and Moré, 2002) plot gives us a sense of how good the solutions obtained by the algorithms are relative to the best solutions found. The algorithms that yield a higher value of ρ are of course the ones that are more successful. Various performance profiles may be constructed at different levels of efficiency to explore overall behavior. We will examine three levels of efficiency which are $100n^2$, $10n^2$, and $10n$ maximum number of objective function evaluations to investigate the behavior of algorithms.

The *user friendliness* of these algorithms is not discussed because it depends on the implementation of the algorithms. However, all five algorithms were easily applied to the test problems.

Another issue that needs to be addressed and agreed upon is the term *iteration* because various algorithms associate different meanings and computational effort with iteration. For instance, a mechanism of a simple genetic algorithm is to progress in an epoch or generation base. During each epoch i, a number of chromosomes or individuals k_i, where k_i is an integer greater or equal to one, are evaluated. Each individual evaluation normally requires one objective function calculation. Hence, there are at least k_i function evaluations needed for that epoch. If there are some l_i mutations during the epoch i, then there are $k_i + l_i$ function evaluations required for that epoch. But the genetic algorithm needs multiple epochs to improve the estimate of the optimal solution. The number of epochs m is usually predetermined by the user. So the total number of function evaluations implemented by this genetic algorithm equals $\sum_{i=0}^{m}(k_i + l_i)$, where epoch zero refers to the algorithm's initialization epoch. If an epoch is equivalent to an iteration, then the number of iterations is not equal to the number of function evaluations in the genetic algorithm context. On the other hand, a simulated annealing type algorithm calls the function evaluation routine only once per iteration. The number of function evaluations is thus the same as the number of iterations in both IHR and HNS. To prevent any future nonequivalent comparison, the number of function evaluations shall be used as the basis to compare and to stop the algorithms.

To directly compare the performance of algorithms, we keep the effort constant and compare the quality of the result obtained. We investigate three lengths of runs, long, medium, and short with $100n^2$, $10n^2$, and $10n$ function evaluations respectively, and record only the improving objective function values. The factor n^2 or n in the maximum number of function evaluations reflect the effect of dimensionality on performance, because problems with higher dimension are expected to be more difficult to solve than those with lower dimension. The quadratic polynomial is used because we expect the performance to grow faster than linear, and it is somewhat consistent with the rate of convergence of IHR on elliptical programs. The coefficients are arbitrary but they are sufficient to demonstrate the effect of the maximum number of function evaluations on the performance of the algorithms.

Considering the various algorithmic comparative issues discussed above, the following experimental procedure was established at the end of the workshop in New Zealand:

1. Assemble a set of 50 test problems.
2. Test selected algorithms on all 50 test problems.
3. Perform 30 replications for each problem and each algorithm.

4. Stop each algorithm after a fixed number of function evaluations ($100n^2$, $10n^2$, or $10n$) for each replication, where n refers to the dimension of the problem.
5. Record only the function evaluations that obtain improving objective function values. Collect both the improving function evaluation numbers and the corresponding objective function values as data.
6. Perform graphical and statistical analysis on the data.

The number of problems in the test problem suite is based on a concept that there should not be too few of the problems that no conclusion can be drawn, yet the test suite should not be overwhelming that this study is unmanageable. The number of test problems was 50, with several variations for some problems as listed in Table 1, bringing the final number up to 56. To take into account the impact of the random number seeds we ran each algorithm on each problem 30 times, totaling $30 \times (50 + 6) = 1680$ replications per algorithm. Implementation of the three population set based algorithms requires setting of some parameter values. We set these parameter values according to suggestions in the respective papers, and the parameter values used in this study are included in Appendix A. Implementation of IHR and HNS is straightforward and does not depend on any parameter. Appendix A describes these algorithms in more detail.

The focus of the analysis is decided to be on the objective function values found by the algorithms, rather than the locations of the solutions in the domain space. This is because locations must distinguish between multiple optima, whereas using objective function values easily includes multiple optima. As a result, only the iterations with improving function values are recorded.

2.2. TEST PROBLEMS FOR COMPUTATIONAL EXPERIMENTS

A collection of 50 test problems and 6 variations have been compiled from the literature and various sources. These problems range from 2 to 20 in dimension and have a variety of inherent difficulty. All the problems have continuous variables. Table 1 lists these problems in alphabetical order with the corresponding number of variables of each problem. Table 2 tallies the problems according to their dimensions. We only have one problem in 20 dimensions, and we recommend that test sets in the future should include more multimodal problems in higher dimensions. A detailed description of each test problem can be found in Appendix B.

3. Comparative Assessment

In this section, we summarize the results obtained from the five algorithms briefly discussed in Section 1. A systematic assessment of global optimiza-

Table 2. Number of test problems by dimension

Dimension n	Number of problems
2	19
3	5
4	9
5	4
6	1
9	2
10	14
17	1
20	1

tion algorithms via a modified performance profile similar in nature to the one suggested by Dolan and Moré (2002) is used. The modified performance profile is applicable to stochastic algorithms in our experimental framework. Dolan and Moré proposed a performance profile of an algorithm based on three items: time $t_{(p,s)}$, performance ratio $r_{(p,s)}$, and fractional performance $\rho_s(\tau)$, where $p \in \mathcal{P}$, the set of all problems and $s \in \mathcal{S}$, the set of all solvers.

Computing time $t_{(p,s)}$ required to solve problem p by solver s was used by Dolan and Moré, but they also recommended that various measures could be used instead of computing time. We replace the computing time $t_{(p,s)}$ with a different measure on the relative improvement of the function values. In particular, we replace $t_{(p,s)}$ with $m_{(p,s)}$ for the m-fold improvement (scaled distance to the optimal function value f^*) found by solver s on problem p after a fixed number k of function evaluations. Precisely, we have:

$$m_{(p,s)} = \frac{\hat{f}_{(p,s)}(\text{after } k \text{ function evaluations}) - f^*}{(f_w - f^*)} \tag{1}$$

for problem p with solver s, where f_w denotes the worst (largest) function value found among the solvers on that particular problem p, and $\hat{f}_{(p,s)}$ (after k function evaluations) is the average estimate of the optimal function values after k function evaluations over 30 replications, found by solver s on problem p. Notice here that $m_{(p,s)}$ is calculated from a single value of $\hat{f}_{(p,s)}$ but there were 30 replications for each solver on each problem in the experiment. To collapse the results from 30 replications to a single number, we considered several approaches, including the average of the estimated optimal function value over all 30 replications, the median value over 30 replications, and the best value found over 30 replications. We decided to use the average as the estimate of the optimal function values. This implies that our focus is on the average performance, which seems to be a typical performance reported in the literature.

The second item, the performance ratio $r_{(p,s)}$, is defined similarly to Dolan and Moré's definition, but again substituting $m_{(p,s)}$ for $t_{(p,s)}$ to obtain:

$$r_{(p,s)} = \frac{m_{(p,s)}}{\min\{m_{(p,s)} \text{ for all } s \in \mathcal{S}\}} \tag{2}$$

for each problem p and solver s. Notice that the value of the performance ratio is always 1 for the solver s that performs the best on a specific problem p. If the performance ratio $r_{(p,s)} = 2$, then the m-fold improvement found by solver s on problem p was twice the best value found by another solver on the same problem p.

The final item is $\rho_s(\tau)$, the fraction of the total number of problems that solver s has a performance ratio $r_{(p,s)}$ within a factor τ:

$$\rho_s(\tau) = \frac{1}{\text{total number of problems}} \{\text{size}\{p \in \mathcal{P} : r_{(p,s)} \leqslant \tau\}\}, \tag{3}$$

where $\mathcal{P}$ is the set of all problems and $\tau \geqslant 1$. The "total number of problems" for our investigation is 56, where the extra six problems are due to variations of some test problems. The "size" is the number of problems such that the performance ratio $r_{(p,s)}$ is less than or equal to τ for solver s.

The performance profile plot compares how well the solvers can estimate the global optimum relative to one another. For instance, in Figure 1(a) at $\tau = 1$, the Genetic Algorithm found the best solutions (first place) in over 50% of the 56 total test problems. When $\tau = 10$, the Genetic Algorithm found a solution within ten times of the best found solution in approximately 75% of the 56 total test problems. The sum of these fractions for any given τ may be greater than one because more than one solver may find solutions within τ multiple of the best found solution on the same problem. The curve of each solver is nondecreasing but its final value on the vertical axis depends on the last value of τ. If the value of the curve is less than one at the last value of τ, it means that on some problems that particular solver got stuck at some local optima and its best solutions found were worse than the best solutions found by some other solver(s) in multiples of τ beyond the horizontal scale. As τ approaches infinity, the final value of each curve will approach one.

Figure 1 also illustrates the effect that a stopping criterion based on the maximum number of function evaluations has on the performance of the algorithms by examining three different levels of the maximum number of function evaluations. Figure 1(a)–(c) show the performance profiles of the the algorithms when the maximum numbers of function evaluations are $100n^2$, $10n^2$, and $10n$, respectively. The three subfigures in Figure 1 show that the performance of the algorithms highly depends on the maximum number of function evaluations allowed. From Figure 1(a) with a maximum of $100n^2$ function evaluations, GA and CRS illustrate promising results by dominating the other algorithms throughout the range of τ. The

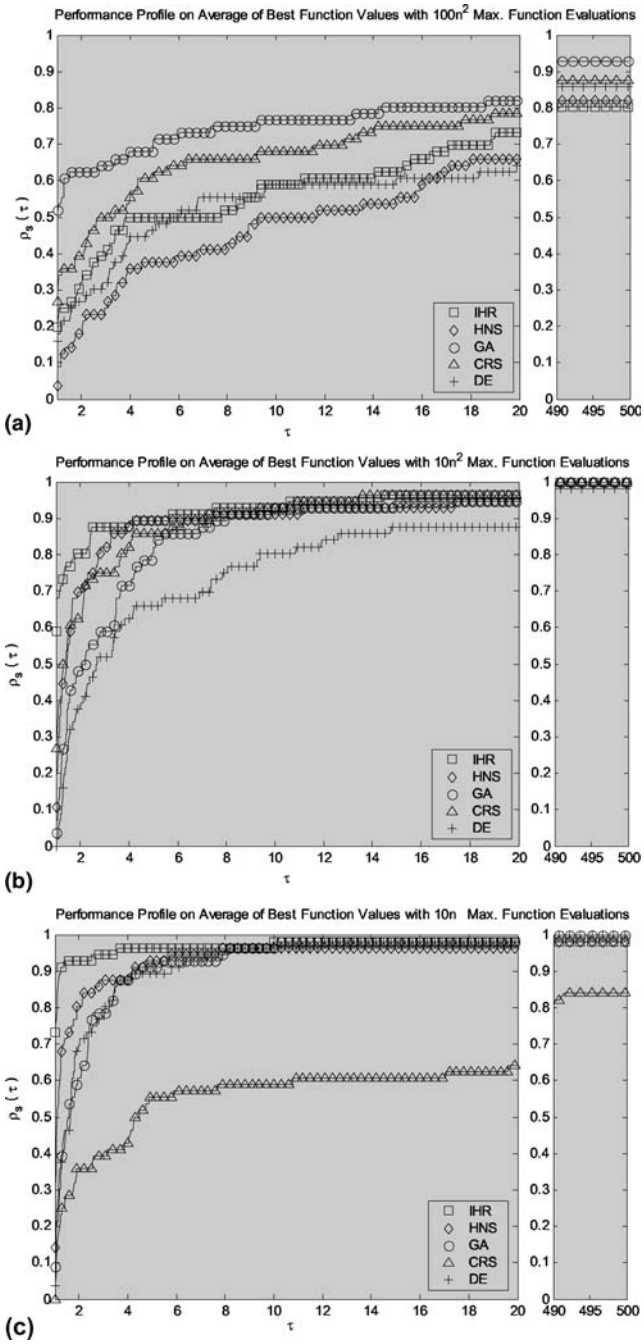

Figure 1. Performance profile plot based on the average of best objective function values.

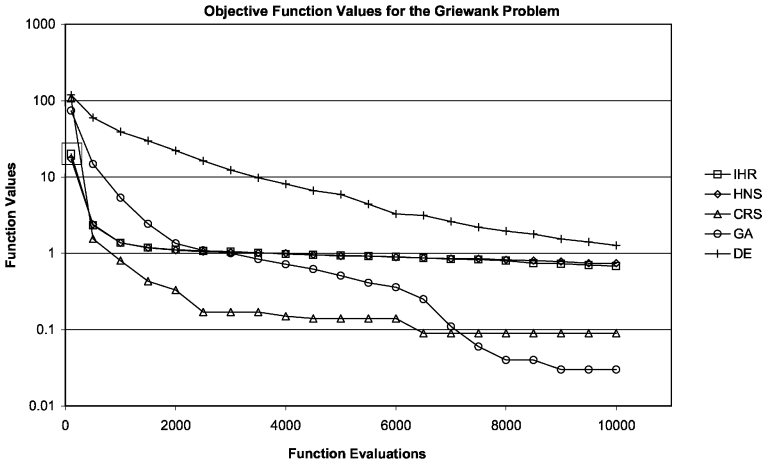

Figure 2. The objective function values for Griewank problem.

same subfigure also shows that IHR and DE perform moderately, whereas HNS is outperformed by the other algorithms. However, when the maximum numbers of function evaluations are $10n^2$ and $10n$ as in Figures 1(b) and (c), HNS performs relatively well among the five algorithms and IHR dominates the other algorithms over most of the τ values. On the other hand, CRS which was the second best algorithm in Figure 1(a) performs poorly when the maximum number of function evaluations is limited to $10n$ as shown in Figure 1(c). This implies that if a practitioner was willing to use $100n^2$ function evaluations, then GA or CRS would be preferred,

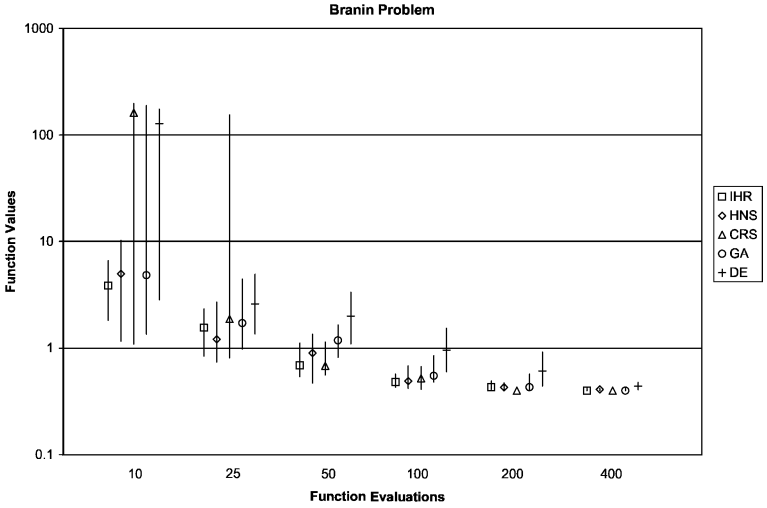

Figure 3. Quartile sequential plot of Branin problem.

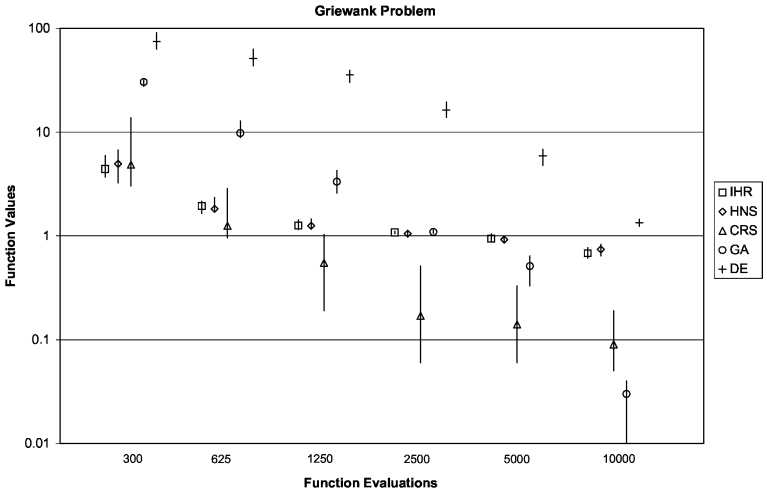

Figure 4. Quartile sequential plot of Griewank problem.

whereas if only $10n$ function evaluations were allowed then IHR or HNS should be implemented.

This effect of the maximum number of function evaluations on algorithm performance is not surprising when we examine the data for individual problems. For example, the data ($m_{(p,s)}$ values) for the Griewank problem as shown in Figure 2 reveals that as the number of function evaluations increases the best algorithm for this problem changes. At the very beginning IHR and HNS perform relatively well compared to the other algorithms. Then at about the 500th function evaluation, CRS outperforms the other algorithms and remains the best algorithm until approximately the 6,500th function evaluation. After this point, GA takes over as the best algorithm for the problem and remains so until the last function evaluation. The figure also shows that although DE is dominated by the other four algorithms throughout the 10,000 function evaluations, it consistently improves its estimate of the global optimum.

Furthermore, we construct individual quartile sequential plots, as shown in Figures 3–6, to capture the progression of the algorithms in terms of the improvement of the objective function values and the variation of the best function values found in each replication as the number of function evaluations increases. We note that the scale of the horizontal axis in the quartile sequential plots illustrated are intentionally not evenly spaced. Algorithms often find a lot of improving function values at the very beginning of replication runs. This behavior will be lost with an evenly scaled axis scheme. Each vertical bar at the k-th function evaluation in the quartile sequential plots represents a range of function values between the 25th and the 75th percentiles. The symbol indicates the 50th percentile at that particular kth function evaluation.

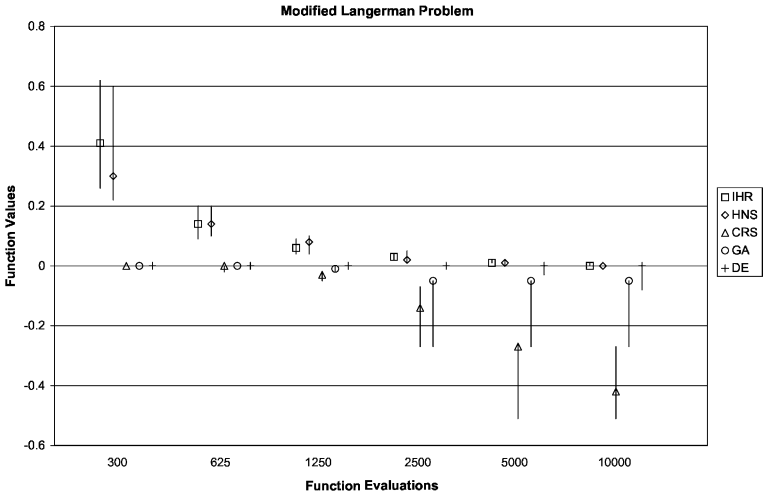

Figure 5. Quartile sequential plot of modified Langerman problem.

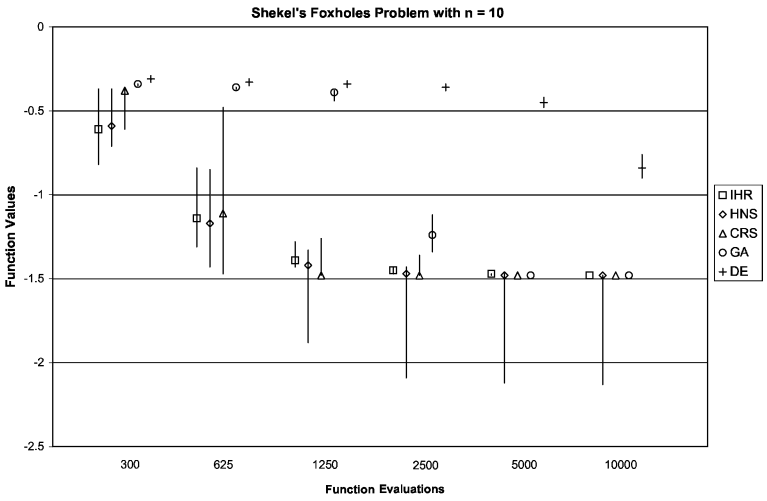

Figure 6. Quartile Sequential Plot of Shekel's Foxholes Problem with $n = 10$.

The Branin Problem in Figure 3 represents a case when all algorithms give similar results for both the average and the variance of all the replications at the last function evaluation. Notice the variance of all algorithms, especially CRS, GA, and DE algorithms, is relatively large at the very first function evaluations. Then all algorithms settle at about the same function value, each with an extremely tight 25–75th quartile range at the last function evaluation. This result suggests that when the variation is high, different approaches (e.g. average, median, or best) to estimate the optimal function values may lead to different conclusions using the performance profile.

In a similar plot of the Griewank Problem shown in Figure 4, the percentile ranges for IHR, HNS, and DE are tighter than those of CRS and GA. As in Figure 2, CRS algorithm performs relatively well compared to the other algorithms but is overtaken by GA at the very last function evaluations. Comparing Figures 2 and 4 illustrates the extra information a quartile sequential plot provides.

The quartile sequential plot for the Modified Langerman Problem in Figure 5 indicates that IHR, HNS, GA, and DE algorithms seem to be trapped at local optima, i.e. the function values are not improving, throughout or after some point during the replication runs. Only the CRS algorithm shows improvement in the function values after getting stuck at some local optima at the very beginning of the runs.

And lastly the quartile sequential plot of the Shekel's Foxholes Problem with $n = 10$ in Figure 6 shows a circumstance where the IHR, CRS, and GA algorithms are clearly trapped at some local optima after 5000 function evaluations with a tight 25–75th percentile range. The HNS algorithm dominates the other four algorithms after 1250 function evaluations. The DE algorithm is again clearly dominated throughout the entire replication run in this plot.

As one can imagine, creating the quartile sequential plots for all 56 problems can become a tedious task. But the detailed information on how the algorithms progress will be completely lost without these plots. While Figure 2 illustrates that CRS outperforms the other algorithms between 500 and 7000 function evaluations, Figure 4 shows the high variation of CRS in this range. The 25–75th percentile range for CRS at 625 function evaluations overlaps the ranges for IHR and HNS, but at 2500 function evaluations the entire quartile range for CRS is below the other algorithms. At 10,000 function evaluations, however, GA overtakes CRS in both average value and quartile range. Without Figure 3, we would not notice the large variations and may be tempted to draw false conclusions based on small differences of the means. And without Figure 6 we would not know that HNS outperformed the other algorithms, as the percentiles were not symmetrical.

While it is clear that the maximum number of function evaluations affects the performance of each algorithm, the choice of the maximum number of function evaluations depends on the user's goal and limitation. Figure 1 indicates that the conclusion of which algorithm performs best depends on the number of function evaluations the user is willing to spend. A better stopping criterion than the maximum number of function evaluations is perhaps one that yields some statistical information and balances the user's cost of performing function evaluations with the benefit of having a high probability of obtaining a solution close to the global optimum. A topic for future research is to establish a statistical stopping criterion for global optimization algorithms.

4. Concluding Remarks

This paper presents a comparative study of stochastic global optimization algorithms using a modified performance profile plot together with quartile sequential plots. The modified performance profile plot is suitable when a macroscopic behavior of the algorithms is to be monitored. But the microscopic behavior of the algorithms can only be revealed with more individual problem plots such as the quartile sequential plots. We demonstrate the effect that the maximum number of function evaluations as the stopping criterion has on the performance of the algorithms. One algorithm may be preferred if a small number of function evaluations is allowed but a different algorithm may be favored if a large number of function evaluations is permitted. We also compiled 50 benchmark test problems that could be readily used and we hope that some of these problems will eventually become a part of the standard test problem set. And lastly we suggest a research direction on developing a statistical stopping criterion for global optimization algorithms.

Acknowledgement

The authors would like to thank János Pintér for his organizational assistance and valuable comments in the early stage of this paper. We also thank János Pintér, Don Kulasiri, and Robert Sherlock for their participation and input at the 2001 Stochastic Global Optimization workshop. Our gratitude goes to Graham Wood who organized the workshop and made this work possible, as well as support from the Marsden Fund administered by the Royal Society of New Zealand. The work of Zelda B. Zabinsky and Charoenchai Khompatraporn has been partially supported by NSF Grant No.s DMI-9820878 and DMI-0244286.

Appendix A. Algorithms

In this section, we present the five algorithms mentioned in the paper. Only a brief description of the algorithms will be given. For more details, we refer to the respective individual references. We begin with the simulated annealing type algorithms.

A.1. HIT-AND-RUN-METHODS

Two Hit-and-Run methods for global optimization are used in the computational experiments performed in this paper. A survey of these methods, as applied to global optimization, can be found in Zabinsky and Wood

(2002). Hit-and-Run was originally developed as a means to generate asymptotically uniformly distributed points (Smith, 1984), and has been applied to global optimization by using it to generate candidate points within a simulated annealing algorithm. The two specific implementations summarized here have been named Improving Hit-and-Run (Zabinsky et al., 1993), and Hide-and-Seek (Romeijn and Smith, 1994).

Both Hit-and-Run methods described here use the same underlying procedure. Given a random point x_k which may depend upon the previous point or several previous points, the algorithm generates a random direction d_k and a random step size s_k along the random direction to determine a candidate point $x_{k+1} = x_k + s_k d_k$. In Improving Hit-and-Run (IHR), the new point x_{k+1} is accepted only if its objective function value is better than that of the previous point, i.e. $f(x_k + s_k d_k) < f(x_k)$ for a minimization problem. In Hide-and-Seek, the candidate point is accepted with a probability that corresponds to the Metropolis acceptance criterion.

Both algorithms are motivated by theoretical results from pure adaptive search and adaptive search, and will converge with probability one to the global optimum for a broad class of problems (summarized in Wood and Zabinsky (2002)). It has also been shown that the expected number of function evaluations of Improving Hit-and- Run on quadratic problems is of order $O(n^{5/2})$ (Zabinsky, et al., 1993). Other variations of these two algorithms have been developed, but will not be discussed here. The basic procedure of the two algorithms is summarized below.

ALGORITHM 1. Improving Hit-and-Run (IHR) Algorithm

Step 1. *Initialize* the iteration counter $k = 0$, and let $x_0 \in \Omega$ and $f_0 = f(x_0)$.

Step 2. *Generate a random direction vector* d_k from a uniform distribution on the surface of a unit n-dimensional hypersphere.

Step 3. *Generate a step size* s_k by sampling uniformly in Ω along the direction vector d_k from the current point x_k. The candidate point is $w_{k+1} = x_k + s_k d_k$.

Step 4. *Update the current point* by accepting or rejecting the candidate point according to the following scheme:

$$x_{k+1} = \begin{cases} x_k + s_k d_k & \text{if } f(x_k + s_k d_k) < f_k \\ x_k & \text{otherwise} \end{cases}$$

Set $f_{k+1} = f(x_{k+1})$.

Step 5. *Stop if the stopping criterion is satisfied.* Otherwise increase k by one and return to Step 2.

ALGORITHM 2. Hide-and-Seek (HNS) Algorithm

Step 1. *Initialize* k, x_0, and f_0 as in Algorithm 1. Also initialize $T_0 \in [0, \infty)$.

Step 2. *Generate a random direction vector* as in Algorithm 1.

Step 3. Generate a step size as in Algorithm 1.

Step 4. *Update the current point* by accepting or rejecting the candidate point according to the following scheme:

$$x_{k+1} = \begin{cases} x_k + s_k d_k & \text{with probability } P_{T_k}(x_k, w_{k+1}) \\ x_k & \text{otherwise} \end{cases}$$

where $P_{T_k}(x_k, w_{k+1}) = \min\{\exp((f(x_k) - f(w_{k+1}))/T_k), 1\}$ and update the temperature according to a cooling schedule, $T_{k+1} = \tau(T_k)$.
Set $f_{k+1} = f(x_{k+1})$.

Step 5. *Stop if the stopping criterion is satisfied.* Otherwise increase k by one and return to Step 2.

In Step 4 of Hide-and-Seek, the probability of accepting the candidate point w_{k+1}, given the current iteration point x_k and the current temperature T_k is expressed as P_{T_k}. This is characteristic of simulated annealing algorithms in general, where, in addition to a generator, there is a probability of accepting nonimproving points. For Hide-and-Seek, the acceptance probability is

$$P_T(x, w) = \begin{cases} 1 & \text{if improving, i.e. } f(w) < f(x) \\ \exp\left\{\frac{f(x) - f(w)}{T}\right\} & \text{otherwise} \end{cases}$$

which is also known as the Metropolis criterion (Metropolis, et al., 1953). Notice that Improving Hit-and-Run only accepts improving points, so may be viewed as the limiting case occurring as T_k tends to zero. Romeijn and Smith (1994) showed that virtually any cooling schedule for Hide-and-Seek could be chosen, as long as the temperature converges to zero in probability. They developed a cooling schedule based on the best objective function values sampled thus far, coupled with an estimate of the unknown global minimum f^*. This adaptive cooling schedule of Hide-and-Seek was used in the computational experiments, and

$$T_k = \frac{2(f(x_k) - \hat{f}(x))}{\chi^2_{1-\alpha}(n)}$$

where $\chi^2_{1-\alpha}(n)$ is the $100(1-\alpha)$ percentile point of the χ-squared distribution with n degrees of freedom. This is motivated by a second-order Taylor series approximation of the objective function, and uses an estimator of the optimum, $\hat{f}(x)$, which is updated when an improving point is obtained. The estimator of the global optimum after k iterations is given by

$$\hat{f}(x_0, \ldots, x_k) = f_{(0)}^k - \frac{f_{(1)}^k - f_{(0)}^k}{(1-q)^{-n/2} - 1}$$

where $f_{(0)}^k = \min_{j=0,1,\ldots,k} f(x_j)$ is the smallest objective function value found after k iterations, $f_{(1)}^k$ is the second smallest, and $q \in (0,1)$ so that $\hat{f}$ is the lower endpoint of a $100(1-q)\%$ confidence interval for f^*.

A.2. POPULATION SET BASED GLOBAL OPTIMIZATION METHODS

Population set based global optimization methods have been developed since the 1970's, including Genetic Algorithms. A fundamental concept of population set based global optimization methods is to successively transform the population set S into points that concentrate at the global optimum, where S is a subset of the domain Ω. Ali and Törn (2002) have experimentally shown that some modifications of these methods are efficient and robust. We present here a few modified versions of the population set based algorithms, namely the Controlled Random Search algorithm, a real coded Genetic Algorithm, and the Differential Evolution algorithm.

A.2.1. *Controlled Random Search (CRS)*

Controlled Random Search (CRS) algorithm was first developed by Price (1977, 1983). The algorithm generates alternative solutions to replace the worst solution in a population set S through simplices. Several versions of the CRS algorithm have been proposed. We will refer to the original CRS algorithm as CRS1 (Price, 1977). The first two improvements, namely CRS2 (Price, 1983) and CRS3 (Price, 1987), were suggested by Price himself. Subsequently, Ali and Storey (1994) suggested CRS4 and CRS5, and the experiments proved CRS4 to be superior than CRS5 (Ali et al., 1997). More details of CRS4 algorithm can be found in Ali and Storey (1994) and Ali and Törn 2004. The experiment conducted in this study used the CRS4 version described below.

ALGORITHM 3. Controlled Random Search 4 (CRS4) Algorithm

Step 1. *Determine the initial set* $S = \{x_1, x_2, \ldots, x_N\}$ where the points $x_i, i = 1, 2, \ldots, N$ are sampled uniformly in Ω. Evaluate $f(x_i)$ at each point $x_i, i = 1, 2, \ldots, N$. Take $N = 10n$, where n is the dimension of x. Set generation counter $k = 0$.

Step 2. *Determine the best and the worst points in S.* Determine the points $x_{\max}$, $x_{\min}$ and their function values $f_{\max}$, $f_{\min}$ such that $f_{\max} = \max_{x \in S} f(x)$ and $f_{\min} = \min_{x \in S} f(x)$. If the stopping condition (e.g. $f_{\max} - f_{\min} < \epsilon$) is satisfied, then stop.

Step 3. *Generate a new point to replace a point in S.* Choose uniformly n distinct points $x_{p(1)}, x_{p(2)}, \ldots, x_{p(n)}$ from S (selection) and compute $x_{\text{new}} = 2G - x_{p(n)}$ where the centroid G is given by

$$G = \frac{1}{n}\left(x_{\min} + \sum_{j=1}^{n-1} x_{p(j)}\right). \tag{A1}$$

If $x_{\text{new}} \neq \Omega$ go to Step 3; otherwise compute $f(x_{\text{new}})$. If $f(x_{\text{new}}) \geqslant f_{\max}$ then go to Step 3.

Step 4. *Update S.* Update S by $S = S \cup \{x_{\text{new}}\} - \{x_{\max}\}$. If x_{new} becomes the new best point in S, then evaluate f in r points (e.g. $r = 4$) from the β-distribution using the current $x_{\min}$ as its mean and the distance between $x_{\min}$ and $x_{\max}$ as standard deviation. Update S for each point if necessary. Increase k by one and go to Step 2.

A.2.2. *Genetic Algorithm (GA)*

Another popular population set based method is the Genetic Algorithm (GA) (Goldberg, 1989). In this method a set $S \subset \Omega$ is initialized. Then S is evolved from one generation to the next by replacing subsets of S successively. As the algorithm proceeds the set of solutions in S tends towards the global optimum. In GA, every solution x is often coded using a binary string representation, but this study used a real coded GA to represent continuous variables. Three concepts involving the evolution of the set S in the basic GA are:

- *Evaluation*: The function $f(x)$ is evaluated every time a new point (or solution) is chosen to be in the current set S.
- *Stochastic selection*: From the current set S, randomly select m points in S to be parents, with bias towards better points.
- *Reproduction*: The selected points (parents) above are used to produce a set of new points called children using two genetic operations: crossover and mutation. The crossover operation is achieved by taking two points, cutting the strings at a random index and exchanging parts by arithmetic crossovers described in Equations A2 and A3. Mutation is achieved by simply flipping the bit at some random index.

This cycle of evaluation, selection and reproduction terminates when a stopping criterion is met. We present here a real coded genetic algorithm (RGA) for continuous variables as presented in (Ali and Törn, 2004). In this implementation of GA two children points are calculated at a time. The crossover operator has two parts. In the first part a child (point) is calculated from randomly selected $n + 2$ parents, $x_{p(1)}, x_{p(2)}, \ldots, x_{p(n+2)}$, from S, where n is the dimension of the problem. The selected $n + 2$ points are used to calculate the centroid G as in Equation A1 of the n points

remaining after excluding the two worst points, say $x_{p(n+1)}$ and $x_{p(n+2)}$. The first child is then taken as the best of $\{\hat{x}_1, \hat{x}_2\}$, where

$$\hat{x}_1 = 2G - x_{p(n+1)} \quad \text{and} \quad \hat{x}_2 = 2G - x_{p(n+2)}. \tag{A2}$$

If the j-th point ($j = 1, 2$), $\hat{x}_j \notin \Omega$ then it is calculated as $\hat{x}_j = \frac{1}{2}(G + x_{p(n+j)})$.

The second child is found from the best of $\{\hat{x}_3, \hat{x}_4\}$, where $\hat{x}_3$ and $\hat{x}_4$ are obtained using the second part of the crossover rule which will now be defined. The second part of crossover is done arithmetically, i.e. given the parent vectors $x = (x^1, \ldots, x^i, \ldots, x^n)$ and $y = (y^1, \ldots, y^i, \ldots, y^n)$ one calculates the i-th component

$$\hat{x}_3^i = \alpha_i x^i + (1 - \alpha_i) y^i, \quad \hat{x}_4^i = \alpha_i y^i + (1 - \alpha_i) x^i, \quad i = 1, 2, \ldots, n \tag{A3}$$

where α_i are uniform random numbers in $[-0.5, 1.5]$.

This crossover is carried out between two parents (e.g. x and y) selected randomly from the $n + 2$ points, again excluding the worst points $x_{p(n+1)}$ and $x_{p(n+2)}$. If the trial points fall outside Ω, random selection of $\alpha^i \in [-0.5, 1, 5]$ continues until $\{\hat{x}_3, \hat{x}_4\} \in \Omega$. For the next pair of children $n + 2$ points are again selected randomly from S and the above process continued. The m children are therefore generated two at a time. Mutation for a point is carried out by setting

$$\hat{x}^i = \hat{x}^i + \gamma(U^i - L^i) \tag{A4}$$

for a randomly selected index i. Here, U^i and L^i are the upper and lower bound of the element x^i and γ is a random number in $[-0.01, 0.01]$. The mutation probability is taken to be 0.001. The description of RGA is presented below.

ALGORITHM 4. Real Coded Genetic Algorithm (RGA)

Step 1. *Determine the initial set S.* Same as in Algorithm 3.

Step 2. *Determine the best and the worst points in S.* Same as in Algorithm 3.

Step 3. *Generate m new points to replace m points in S.*

- Selection: select $n + 2$ points randomly from S as parents.
- Create two points (children) using crossover (parts I and II) and mutation.

  Crossover: use first part of the crossover rule in Equation A2 to create the first point (child) and the second part in Equation A3 for the second point.

  Mutation: mutate a component of each point with probability 0.001 using Equation (7). Mutation is repeated if the mutated solution is infeasible.

Repeat Step 3 until m points are created.

Step 4. *Update S.* Replace m worse points in S with the m new points generated. Increase k by one and go to Step 2.

A.2.3. *Differential Evolution (DE)*

Storn and Price (1997) recently proposed a new population set based method, the Differential Evolution (DE) Method. Unlike CRS which attempts to replace a single point in S per generation, and GA which replaces m points (parents) of S by the new m points (children) per generation, DE attempts to replace all points in S by new points at each generation. We present here a modification of the differential evolution algorithm, namely the differential evolution algorithm using pre-calculated differentials (DEPD) designed by Ali and Törn (2002). The overall structure of DEPD resembles that of CRS and GA. Like the other population set based global optimization methods DEPD also attempts to guide an initial set $S = \{x_1, x_2, \ldots, x_N\}$ of points in Ω to the vicinity of the global minimum through repeated cycles of selection, reproduction (mutation and crossover), and acceptance. In its mutation phase DEPD randomly selects three distinct points x_{r1}, x_{r2} and x_{r3} from the current set S. None of these points should coincide with the current target point x_i. The weighted difference of any two points is then added to the third point which can be mathematically described as

$$\hat{x}_i = x_{r1} + F(x_{r2} - x_{r3}) \tag{A5}$$

where F is a scaling factor which can be calculated automatically using a formula, see Ali and Törn (2002, 2004). All the differential vectors $(x_{r2} - x_{r3})$ are stored in an array A before A is updated with new differentials after every R generation. The trial point y_i is found from its parents x_i, the target point, and $\{\hat{x}_i$ using the following crossover rule

$$y_i^j = \begin{cases} \hat{x}_i^j & \text{if } r^j \leqslant 0.5 \text{ or } j = I_i \\ x_i^j & \text{if } r^j > 0.5 \text{ and } j \neq I_i \end{cases} \tag{A6}$$

where I_i is a randomly chosen integer in the set I, i.e., $I_i \in I = \{1, 2, \ldots, n\}$; the superscript j represents the j-th component of respective vectors; $r^j \in (0, 1)$ drawn randomly for each j. In the acceptance phase the function value at the trial point, $f(y_i)$, is compared to $f(x_i)$, the value at the target point. If $f(y_i) < f(x_i)$ then y_i replaces x_i in S, otherwise S retains the original x_i. The process continues until all members of S are targeted. The DEPD can be summarized as follows.

ALGORITHM 5. DEPD Algorithm

Step 1. *Determine the initial set S.* Same as in Algorithm 3.

Step 2. *Determine the best and the worst points in S.* Same as in Algorithm 3.

Step 3. *Generate N new points to replace all points in S.* For each $x_i \in S$, determine y_i by the following two reproduction operations:

- *Mutation:* If $(k = 0)$ or $k \equiv 0 \pmod{R}$ execute mode 1 else mode 2.
  - Mode 1: Randomly select three points x_{r1}, x_{r2}, and x_{r3} from S except x_i, the running target and find the point $\hat{x}_i$ by the mutation rule in equation A5 using the differential vector $(x_{r2} - x_{r3})$. If a component $\hat{x}_i^j$ falls outside Ω then it is found randomly in between the j-th lower and upper limits. Update the i-th element of the array A with this differential.
  - Mode 2: Randomly select a point x_{r1} from S and a differential vector from A and find the point $\hat{x}_i$ by the mutation rule in equation A5. If a component $\hat{x}_i^j$ falls outside Ω then it is found randomly in-between the j-th lower and upper limits.
- **Crossover:** Calculate the trial vector y_i corresponding to the target x_i from x_i and $\hat{x}_i$ using the crossover rule in Equation A6.

Step 4. *Update S.* Select each trial vector y_i for the $(k+1)$-th generation using the acceptance criterion: replace $x_i \in S$ with y_i if $f(y_i) < f(x_i)$, otherwise retain x_i. Increase k by one and go to Step 2.

Appendix B. A Collection of Benchmark Global Optimization Test Problems

In this appendix we present 50 popular test problems which are often used by GO researchers. Some of these problems can be found in textbooks, in individual research articles, or at web sites, but it is difficult to find a single source for all of them. Therefore we produce these models in full detail below. Please note that in several cases the optimal solution vector and corresponding global solution are known only as a numerical approximation.

1. *Ackley's Problem (ACK)* (Storn and Price, 1997)

$$\min_x f(x) = -20\exp\left(-0.02\sqrt{n^{-1}\sum_{i=1}^{n} x_i{}^2}\right) - \exp\left(n^{-1}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$$

subject to $-30 \leqslant x_i \leqslant 30$, $i \in \{1, 2, \ldots, n\}$. The number of local minima is not known. The global minimum is located at the origin with $f(x^*) = 0$. Tests were performed for $n = 10$.

2. *Aluffi-Pentini's Problem (AP)* (Aluffi-Pentini et al., 1985)

$$\min_x f(x) = 0.25{x_1}^4 - 0.5{x_1}^2 + 0.1x_1 + 0.5{x_2}^2$$

subject to $-10 \leqslant x_1, x_2 \leqslant 10$. The function has two local minima, one of them is global with $f(x^*) \approx -0.3523$ located at $(-1.0465, 0)$.

3. *Becker and Lago Problem (BL)* (Price, 1977)

$$\min_x f(x) = (|x_1| - 5)^2 + (|x_2| - 5)^2$$

subject to $-10 \leqslant x_1, x_2 \leqslant 10$. The function has four minima located at $(\pm 5, \pm 5)$, all with $f(x^*) = 0$.

4. *Bohachevsky 1 Problem (BF1)* (Bohachevsky et al., 1986)

$$\min_x f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$$

subject to $-50 \leqslant x_1, x_2 \leqslant 50$. The number of local minima is unknown but the global minimizer is located at $x^* = (0, 0)$ with $f(x^*) = 0$.

5. *Bohachevsky 2 Problem (BF2)* (Bohachevsky et al., 1986)

$$\min_x f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2) + 0.3$$

subject to $-50 \leqslant x_1, x_2 \leqslant 50$. The number of local minima is unknown but the global minimizer is located at $x^* = (0, 0)$ with $f(x^*) = 0$.

6. *Branin Problem (BP)* (Dixon and Szegö, 1978)

$$\min_x f(x) = a(x_2 - b{x_1}^2 + cx_1 - d)^2 + g(1 - h)\cos(x_1) + g$$

where $a = 1$, $b = 5.1/(4\pi^2)$, $c = 5/\pi$, $d = 6$, $g = 10$, $h = 1/(8\pi)$. The region of interest is given by $-5 \leqslant x_1 \leqslant 10$ and $0 \leqslant x_2 \leqslant 15$. There are three minima, all global, in this region. The minimizers are $x^* \approx (-\pi, 12.275), (\pi, 2.275), (3\pi, 2.475)$ with $f(x^*) = 5/(4\pi)$.

7. *Camel Back – 3 Three Hump Problem (CB3)* (Dixon and Szegö, 1975)

$$\min_x f(x) = 2x_1^2 - 1.05x_1^4 + \frac{1}{6}x_1^6 + x_1x_2 + x_2^2$$

subject to $-5 \leqslant x_1,\ x_2 \leqslant 5$. The function has three local minima, one of them is global located at $x^* = (0, 0)$ with $f(x^*) = 0$.

8. *Camel Back – 6 Six Hump Problem (CB6)* (Dixon and Szegö, 1978; Michalewicz, 1996)

$$\min_x f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$$

subject to $-5 \leqslant x_1,\ x_2 \leqslant 5$. This function is symmetric about the origin and has three conjugate pairs of local minima with values $f \approx -1.0316$, -0.2154, 2.1042. The function has two global minima at $x^* \approx (0.089842, -0.712656)$ and $(-0.089842, 0.712656)$ with $f(x^*) \approx -1.0316$.

9. *Cosine Mixture Problem (CM)* (Breiman and Cutler, 1993)

$$\max_x f(x) = 0.1\sum_{i=1}^{n}\cos(5\pi x_i) - \sum_{i=1}^{n} x_i^2$$

subject to $-1 \leqslant x_i \leqslant 1$, $i \in \{1, 2, \ldots, n\}$. The global maxima are located at the origin with the function values 0.20 and 0.40 for $n = 2$ and $n = 4$, respectively.

10. *Dekkers and Aarts Problem (DA)* (Dekkers and Aarts, 1991)

$$\min_x f(x) = 10^5 x_1^2 + x_2^2 - (x_1^2 + x_2^2)^2 + 10^{-5}(x_1^2 + x_2^2)^4$$

subject to $-20 \leqslant x_1, x_2 \leqslant 20$. The origin is a local minimizer, but there are two global minimizers located at $x^* = (0, 15)$ and $(0,-15)$ with $f(x^*) = -24777$.

11. *Easom Problem (EP)* (Michalewicz, 1996)

$$\min_x f(x) = -\cos(x_1)\cos(x_2)\exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$$

subject to $-10 \leqslant x_1,\ x_2 \leqslant 10$. The minimum value is located at (π, π) with $f(x^*) = -1$. The function value rapidly approaches zero, when away from (π, π).

12. *Epistatic Michalewicz Problem (EM)* (second ICEO)

$$\min_x f(x) = -\sum_{i=1}^{n} \sin(y_i)(\sin(iy_i^2/\pi))^{2m}$$

$$y_i = \begin{cases} x_i \cos(\theta) - x_{i+1}\sin(\theta), & i = 1, 3, 5, \ldots, \leqslant \text{n} \\ x_i \sin(\theta) + x_{i+1}\cos(\theta), & i = 2, 4, 6, \ldots, \leqslant \text{n} \\ x_i, & i = n \end{cases}$$

subject to $0 \leqslant x_i \leqslant \pi,\ i \in \{1, 2, \ldots, n\},\ \theta = \pi/6$ and $m = 10$. The number of local minima is not known but the global minimizer is presented in Table 3.

13. *Exponential Problem (EXP)* (Breiman and Cutler, 1993)

$$\max_x f(x) = \exp\left(-0.5\sum_{i=1}^{n} x_i^2\right)$$

subject to $-1 \leqslant x_i \leqslant 1,\ i \in \{1, 2, \ldots, n\}$. The optimal value $f(x^*) = 1$ is located at the origin. Our tests were performed with $n = 10$.

14. *Goldstein and Price (GP)* (Dixon and Szegö, 1978)

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

subject to $-2 \leqslant x_1,\ x_2 \leqslant 2$. There are four local minima and the global minima is located at $x^* = (0, -1)$, with $f(x^*) = 3$.

15. *Griewank Problem (GW)* (Griewank, 1981; Jansson and Knüppel, 1995)

$$\min_x f(x) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right)$$

Table 3. Epistatic Michalewicz's global optimizers

n	*f*(*x**)	*x**
5	−4.687658	(2.693, 0.259, 2.074, 1.023, 1.720)
10	−9.660152	(2.693, 0.259, 2.074, 1.023, 2.275, 0.500, 2.138, 0.794, 2.219, 0.533)

subject to $-600 \leqslant x_i \leqslant 600$, $i \in \{1, 2, \ldots, n\}$. The function has a global minimum located at $x^* = (0, 0, \ldots, 0)$ with $f(x^*) = 0$. Number of local minima for arbitrary n is unknown, but in the two dimensional case there are some 500 local minima. Tests were performed for $n = 10$.

16. *Gulf Research Problem (GRP)* (Moré et al., 1981; Himmelblau, 1972)

$$\min_x f(x) = \sum_{i=1}^{99} \left[\exp\left(-\frac{(u_i - x_2)^{x_3}}{x_1} \right) - 0.01 \times i \right]^2$$

where $u_i = 25 + [-50 \ln(0.01 \times i)]^{1/1.5}$ subject to $0.1 \leqslant x_1 \leqslant 100$, $0 \leqslant x_2 \leqslant 25.6$, and $0 \leqslant x_3 \leqslant 5$. This problem has a global minimizer at (50,25,1.5) with $f(x^*) = 0$.

17. *Hartman 3 Problem (H3)* (Dixon and Szegö, 1978)

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

subject to $0 \leqslant x_j \leqslant 1$, $j \in \{1, 2, 3\}$ with constants a_{ij}, p_{ij} and c_i given in Table 4. There are four local minima, $x_{\text{local}} \approx (p_{i1}, p_{i2}, p_{i3})$ with $f(x_{\text{local}}) \approx -c_i$. The global minimum is located at $x^* \approx (0.114614, 0.555649, 0.852547)$ with $f(x^*) \approx -3.862782$.

18. *Hartman 6 Problem (H6)* (Dixon and Szegö, 1978)

$$\min_x f(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2 \right]$$

subject to $-0 \leqslant x_j \leqslant 1$, $j \in \{1, \ldots, 6\}$, with constants a_{ij} and c_i given in Table 5 and constants p_{ij} in Table 6. There are four local minima,

Table 4. Data for Hartman 3 Problem

i	c_i	a_{ij}			p_{ij}		
		$j = 1$	2	3	$j = 1$	2	3
1	1	3	10	30	0.3689	0.117	0.2673
2	1.2	0.1	10	35	0.4699	0.4387	0.747
3	3	3	10	30	0.1091	0.8732	0.5547
4	3.2	0.1	10	35	0.03815	0.5743	0.8828

Table 5. Data for Hartman 6 Problem

i	c_i	a_{ij}					
		$j = 1$	2	3	4	5	6
1	1	10	3	17	3.5	1.7	8
2	1.2	0.05	10	17	0.1	8	14
3	3	3	3.5	1.7	10	17	8
4	3.2	17	8	0.05	10	0.1	14

Table 6. Data for Hartman 6 Problem

i	p_{ij}					
	$j=1$	2	3	4	5	6
1	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	0.2348	0.1451	0.3522	0.2883	0.3047	0.665
4	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

$x_{\text{local}} \approx (p_{i1}, \ldots, p_{i6})$ with $f(x_{\text{local}}) \approx -c_i$. The global minimum is located at $x^* \approx (0.201690, 0.150011, 0.476874, 0.275332, 0.311652, 0.657301)$ with $f(x^*) \approx -3.322368$.

19. *Helical Valley Problem (HV)* (Wolfe, 1978)

$$\min_x f(x) = 100[(x_2 - 10\theta)^2 + (\sqrt{(x_1^2 + x_2^2)} - 1)^2] + x_3^2$$

where

$$\theta = \begin{cases} \frac{1}{2\pi}\tan^{-1}\frac{x_2}{x_1}, & \text{if } x_1 \geqslant 0 \\ \frac{1}{2\pi}\tan^{-1}\frac{x_2}{x_1} + \frac{1}{2}, & \text{if } x_1 < 0 \end{cases}$$

subject to $-10 \leqslant x_1, x_2, x_3 \leqslant 10$. This is a steep-sided valley which follows a helical path. The minimum is located at $x^* = (1, 0, 0)$ with $f(x^*) = 0$.

20. *Hosaki Problem (HSK)* (Benke and Skinner, 1991)

$$\min_x f(x_1, x_2) = (1 - 8x_1 + 7x_1^2 - 7/3x_1^3 + 1/4x_1^4)x_2^2 \exp(-x_2)$$

subject to $0 \leqslant x_1 \leqslant 5$, $0 \leqslant x_2 \leqslant 6$. There are two minima of which the global minimum is $f(x^*) \approx -2.3458$ with $x^* = (4, 2)$.

21. *Kowalik Problem (KL)* (Jansson and Knüppel, 1995)

$$\min_x f(x) = \sum_{i=1}^{11} \left(a_i - \frac{x_1(1 + x_2 b_i)}{(1 + x_3 b_i + x_4 b_i^2)} \right)^2$$

subject to $0 \leqslant x_i \leqslant 0.42$, $i \in \{1, 2, 3, 4\}$. The values for a_i and b_i are given in Table 7:

This is a least squares problem with a global optimal value $f(x^*) \approx 3.0748 \times 10^{-4}$ located at $x^* \approx (0.192, 0.190, 0.123, 0.135)$.

Table 7. Data for Kowalik Problem

i	1	2	3	4	5	6	7	8	9	10	11
a_i	0.1957	0.1947	0.1735	0.16	0.0844	0.0627	0.0456	0.0342	0.0323	0.0235	0.0246
b_i	0.25	0.50	1.0	2.0	4.0	6.0	8.0	10.0	12.0	14.0	16.0

22. *Levy and Montalvo 1 Problem (LM1)* (Levy and Montalvo, 1985)

$$\min_x f(x) = (\pi/n)\Big(10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\Big)$$

where $y_i = 1 + \frac{1}{4}(x_i + 1)$ for $-10 \leqslant x_i \leqslant 10$, $i \in \{1, 2, \ldots, n\}$. There are approximately 5^n local minima and the global minimum is known to be $f(x^*) = 0$ with $x^* = (-1, -1, \ldots, -1)$. Our tests were performed with $n = 3$.

23. *Levy and Montalvo 2 Problem (LM2)* (Levy and Montalvo, 1985; Dekkers and Aarts, 1991)

$$\min_x f(x) = 0.1\Big(\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\Big)$$

subject to $-5 \leqslant x_i \leqslant 5$, $i \in \{1, 2, \ldots, n\}$. There are approximately 15^n minima and the global minimizer is known to be $x^* = (1, 1, \ldots, 1)$ with $f(x^*) = 0$. Our tests were performed with $n = 5, 10$.

24. *McCormick Problem (MC)* (McCormick, 1982)

$$\min_x f(x) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - (3/2)x_1 + (5/2)x_2 + 1$$

subject to $-1.5 \leqslant x_1 \leqslant 4$ and $-3 \leqslant x_2 \leqslant 3$. This problem has a local minimum at (2.59, 1.59) and a global minimum at (−0.547, −1.547) with $f(x^*) \approx -1.9133$.

25. *Meyer and Roth Problem (MR)* (Wolfe, 1978)

$$\min_x f(x) = \sum_{i=1}^{5}\left(\frac{x_1 x_3 t_i}{(1 + x_1 t_i + x_2 v_i)} - y_i\right)^2$$

subject to $-10 \leqslant x_i \leqslant 10$, $i \in \{1, 2, 3\}$. This is a least squares problem with minimum value $f(x^*) \approx 0.4 \times 10^{-4}$ located at $x^* \approx (3.13, 15.16, 0.78)$. Table 8 lists the parameter values of this problem.

26. *Miele and Cantrell Problem (MCP)* (Wolfe, 1978)

$$\min_x f(x) = (\exp(x_1) - x_2)^4 + 100(x_2 - x_3)^6 + (\tan(x_3 - x_4))^4 + x_1^8$$

Table 8. Data for Meyer and Roth Problem

i	t_i	v_i	y_i
1	1.0	1.0	0.126
2	2.0	1.0	0.219
3	1.0	2.0	0.076
4	2.0	2.0	0.126
5	0.1	0.0	0.186

Table 9. Data for Modified Langerman Problem

j	c_j	a_{ji}									
		$i = 1$	2	3	4	5	6	7	8	9	10
1	0.806	9.681	0.667	4.783	9.095	3.517	9.325	6.544	0.211	5.122	2.020
2	0.517	9.400	2.041	3.788	7.931	2.882	2.672	3.568	1.284	7.033	7.374
3	0.100	8.025	9.152	5.114	7.621	4.564	4.711	2.996	6.126	0.734	4.982
4	0.908	2.196	0.415	5.649	6.979	9.510	9.166	6.304	6.054	9.377	1.426
5	0.965	8.074	8.777	3.467	1.867	6.708	6.349	4.534	0.276	7.633	1.567

subject to $-1 \leqslant x_i \leqslant 1$, $i \in \{1, 2, 3, 4\}$. The number of local minima is unknown but the global minimizer is located at $x^* = (0, 1, 1, 1)$ with $f(x^*) = 0$.

27. *Modified Langerman Problem (ML)* (second ICEO)

$$\min_x f(x) = -\sum_{j=1}^{5} c_j \cos(d_j/\pi) \exp(-\pi d_j),$$

where $d_j = \sum_{i=1}^{n} (x_i - a_{ji})^2$ and $0 \leq x_i \leqslant 10$, $i \in \{1, 2, \ldots, n\}$. The test used $n = 10$. The constants c_j and a_{ji} are given in Table 9. The number of local minima is not known, but the global minima are shown in Table 10.

28. *Modified Rosenbrock Problem (MRP)* (Price, 1977)

$$\min_x f(x) = 100(x_2 - {x_1}^2)^2 + \left[6.4(x_2 - 0.5)^2 - x_1 - 0.6\right]^2$$

subject to $-5 \leqslant x_1, x_2 \leqslant 5$. This function has two global minima each with $f(x^*) = 0$ (corresponding to the intersection of two parabolas) and a local minimum (where the parabolas approach without intersection). The global minima are located at $x^* \approx (0.3412, 0.1164)$, (1,1).

29. *Multi-Gaussian Problem (MGP)* (Benke and Skinner, 1991)

$$\max_x f(x) = \sum_{i=1}^{5} a_i \exp(-((x_1 - b_i)^2 + (x_2 - c_i)^2)/{d_i}^2)$$

subject to $-2 \leqslant x_1, x_2 \leqslant 2$. The function has one global maximum at $x^* \approx (-0.01356, -0.01356)$ with $f(x^*) \approx 1.29695$. There are also 4 other local maxima and a saddle point. Values for the parameters a_i, b_i, c_i, and d_i are given in Table 11.

Table 10. Global optimizers for Modified Langerman Problem

n	$f(x^*)$	x^*
5	−0.965	(8.074, 8.777, 3.467, 1.867, 6.708)
10	−0.965	(8.074, 8.777, 3.467, 1.867, 6.708, 6.349, 4.534, 0.276, 7.633, 1.567)

Table 11. Data for Multi-Gaussian Problem

i	a_i	b_i	c_i	d_i
1	0.5	0.0	0.0	0.1
2	1.2	1.0	0.0	0.5
3	1.0	0.0	−0.5	0.5
4	1.0	−0.5	0.0	0.5
5	1.2	0.0	1.0	0.5

30. *Neumaier 2 Problem (NF2)* (Neumaier, 2003b)

$$\min_x f(x) = \sum_{k=1}^{n} \left(b_k - \sum_{i=1}^{n} x_i^k \right)^2$$

subject to $0 \leqslant x_i \leqslant n,\ i \in \{1, 2, \ldots, n\}$. We consider a case when $n = 4$ and $b = (8, 18, 44, 114)$. The global minimum is $f(1, 2, 2, 3) = 0$.

31. *Neumaier 3 Problem (NF3)* (Neumaier, 2003b)

$$\min_x f(x) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$$

subject to $-n^2 \leqslant x_i \leqslant n^2,\ i \in \{1, 2, \ldots, n\}$. The case considered here is $n = 10$. The number of local minima is not known, but the global minima can be expressed as:

$$f(x^*) = -\frac{n(n+4)(n-1)}{6}, \quad x_i^* = i(n+1-i).$$

The global minima for some values of n are presented in Table 12.

32. *Odd Square Problem (OSP)* (Second ICEO)

$$\min_x f(x) = -(1.0 + 0.2d/(D + 0.1)) \cos(D\pi) e^{-D/2\pi}$$

where

$$d = \sqrt{\sum_{i=1}^{n} (x_i - b_i)^2}, \quad D = \sqrt{n}(\max |x_i - b_i|),$$

$-15 \leqslant x_i \leqslant 15,\ i \in \{1, 2, \ldots, 20\}$ and

$$\underline{b} = (1, 1.3, 0.8, -0.4, -1.3, 1.6, -2, -6, 0.5, 1.4, 1, 1.3, 0.8, -4, -1.3, 1.6, -0.2, -0.6, 0.5, 1.4).$$

The number of local minima for a given n is not known but the global minimum is known to be $f(x^*) \approx -1.143833$, $x^* \cong \underline{b}$ (many solutions near $\underline{b}$). We used $n = 10$ in our experiment.

Table 12. Global minima for Neumaier 3 Problem

n	10	15	20	25	30
$f(x^*)$	−210	−665	−1520	−2900	−4930

33. *Paviani Problem (PP)* (Himmelblau, 1972)

$$\min_x f(x) = \sum_{i=1}^{10}[(\ln(x_i - 2))^2 + (\ln(10 - x_i))^2] - \left(\prod_{i=1}^{10} x_i\right)^{0.2}$$

subject to $2 \leqslant x_i \leqslant 10,\ i \in \{1, 2, \ldots, 10\}$. This function has a global minimizer at $x_i^* \approx 9.351$ for all i, with $f(x^*) \approx -45.778$.

34. *Periodic Problem (PRD)* (Price, 1977)

$$\min_x f(x) = 1 + \sin^2 x_1 + \sin^2 x_2 - 0.1 \exp(-{x_1}^2 - {x_2}^2)$$

subject to $-10 \leqslant x_1, x_2 \leqslant 10$. There are 49 local minima all with minimum values 1 and global minimum located at $x^* = (0, 0)$ with $f(x^*) = 0.9$.

35. *Powell's Quadratic Problem (PWQ)* (Wolfe, 1978)

$$\min_x f(x) = (x_1 + 10x_1)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$$

subject to $-10 \leqslant x_i \leqslant 10,\ i \in \{1, 2, 3, 4\}$. This is a unimodal function with $f(x^*) = 0,\ x^* = (0, 0, 0, 0)$. The minimizer is difficult to obtain with accuracy as the Hessian matrix at the optimum is singular.

36. *Price's Transistor Modelling Problem (PTM)* (Price, 1977, 1983)

$$\min_x f(x_1, x_2, \ldots, x_9) = \gamma^2 + \sum_{k=1}^{4}({\alpha_k}^2 + {\beta_k}^2)$$

where

$$\alpha_k = (1 - x_1 x_2)x_3\{\exp[x_5(g_{1k} - g_{3k}x_7 \times 10^{-3} - g_{5k}x_8 \times 10^{-3})] - 1\} - g_{5k} + g_{4k}x_2,$$

$$\beta_k = (1 - x_1 x_2)x_4\{\exp[x_6(g_{1k} - g_{2k} - g_{3k}x_7 \times 10^{-3} + g_{4k}x_9 \times 10^{-3})] - 1\} - g_{5k}x_1 + g_{4k},$$

$$\gamma = x_1 x_3 - x_2 x_4,$$

and $-10 \leqslant x_i \leqslant 10,\ i \in \{1, 2, \ldots, 9\}$. The values of g_{ik} are given in Table 13.

The global minimum occurs very close to (0.9,0.45,1,2,8,8,5,1,2) with $f(x^*) = 0$. The number of local minima is unknown.

37. *Rastrigin Problem (RG)* (Storn and Price, 1997; Törn and Žilinskas, 1989)

Table 13. Data for Price's Transistor Modelling Problem

i	g_{ik}			
	$k = 1$	2	3	4
1	0.485	0.752	0.869	0.982
2	0.369	1.254	0.703	1.455
3	5.2095	10.0677	22.9274	20.2153
4	23.3037	101.779	111.461	191.267
5	28.5132	111.8467	134.3884	211.4823

$$\min_x f(x) = 10n + \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)]$$

subject to $-5.12 \leqslant x_i \leqslant 5.12$, $i \in \{1, 2, \ldots, n\}$. The total number of minima for this function is not exactly known but the global minimizer is located at $x^* = (0, 0, \ldots, 0)$ with $f(x^*) = 0$. For $n = 2$, there are about 50 local minimizers arranged in a lattice like configuration. Our tests were performed with $n = 10$.

38. *Rosenbrock Problem (RB)* (Schwefel, 1995; Moré, et al., 1981)

$$\min_x f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$$

subject to $-30 \leqslant x_i \leqslant 30$, $i \in \{1, 2, \ldots, n\}$. Our tests were performed with $n = 10$. This function is known as the extended Rosenbrock function. It is unimodal, yet due to a saddle point it is very difficult to locate the minimizer $x^* = (1, 1, \ldots, 1)$ with $f(x^*) = 0$.

39. *Salomon Problem (SAL)* (Salomon, 1995)

$$\min_x f(x) = 1 - \cos(2\pi\|x\|) + 0.1\|x\|$$

where $\|x\| = \sqrt{\sum_{i=1}^{n} x_i^2}$ and $-100 \leqslant x_i \leqslant 100$. The number of local minima (as a function of n) is not known, but the global minimizer is located at $x^* = (0, 0, 0, \ldots, 0)$ with $f(x^*) = 0$. Our tests were performed with $n = 5, 10$.

40. *Schaffer 1 Problem (SF1)* (Michalewicz, 1996)

$$\min_x f(x) = 0.5 + \frac{\left(\sin\sqrt{x_1^2 + x_2^2}\right)^2 - 0.5}{\left(1 + 0.001(x_1^2 + x_2^2)\right)^2}$$

subject to $-100 \leqslant x_1, x_2 \leqslant 100$. The number of local minima is not known, but the global minimum is located at $x^* = (0, 0)$ with $f(x^*) = 0$.

41. *Schaffer 2 Problem (SF2)* (Michalewicz, 1996)

$$\min_x f(x) = (x_1^2 + x_2^2)^{0.25}(\sin^2(50(x_1^2 + x_2^2)^{0.1}) + 1)$$

subject to $-100 \leqslant x_1, x_2 \leqslant 100$. The number of local minima is not known, but the global minimum is located at $x^* = (0, 0)$ with $f(x^*) = 0$.

42. *Shubert Problem (SBT)* (Levy and Montalvo, 1985)

$$\min_x f(x) = \prod_{i=1}^{n} \left(\sum_{j=1}^{5} j\cos((j+1)x_i + j) \right)$$

subject to $-10 \leqslant x_i \leqslant 10$, $i \in \{1, 2, \ldots, n\}$. Our tests were performed with $n = 2$. The number of local minima for this problem (given n) is not known but for $n = 2$, the function has 760 local minima, 18 of which are global with $f(x^*) \approx -186.7309$. All two dimensional global minimizers are listed in Table 14:

Table 14. Global optimizers for Shubert Problem

x^*				
(−7.0835, 4.8580),	(−7.0835, −7.7083),	(−1.4251, −7.0835),	(5.4828, 4.8580),	(−1.4251, −0.8003),
(4.8580, 5.4828),	(−7.7083, −7.0835),	(−7.0835, −1.4251),	(−7.7083, −0.8003),	(−7.7083, 5.4828),
(−0.8003, −7.7083),	(−0.8003, −1.4251),	(−0.8003, 4.8580),	(−1.4251, 5.4828),	(5.4828, −7.7083),
(4.8580, −7.0835),	(5.4828, −1.4251),	(4.8580, −0.8003)		

43. *Schwefel Problem (SWF)* (Muhlenbein et al., 1991)

$$\min_x f(x) = -\sum_{i=1}^{n} x_i \sin(\sqrt{|x_i|})$$

subject to $-500 \leqslant x_i \leqslant 500$, $i \in \{1, 2, \ldots, n\}$. The number of local minima for a given n is not known, but the global minimum value $f(x^*) \approx -418.9829n$ is located at $x^* = (s, s, \ldots, s)$, $s \approx 420.97$. Our tests were performed with $n = 10$.

44. *Shekel 5 Problem (S5)* (Dixon and Szegö, 1978)

$$\min_x f(x) = -\sum_{i=1}^{5} \frac{1}{\sum_{j=1}^{4} (x_j - a_{ij})^2 + c_i}$$

subject to $0 \leqslant x_j \leqslant 10$, $j \in \{1, 2, 3, 4\}$ with constants a_{ij} and c_j given in Table 15 below. There are five local minima and the global minimizer is located at $x^* = (4.00, 4.00, 4.00, 4.00)$ with $f(x^*) \approx -10.1499$.

45. *Shekel 7 Problem (S7)* (Dixon and Szegö, 1978)

$$\min_x f(x) = -\sum_{j=1}^{7} \frac{1}{\sum_{i=1}^{4} (x_j - a_{ij})^2 + c_i}$$

subject to $0 \leqslant x_j \leqslant 10$, $j \in \{1, 2, 3, 4\}$ with constants a_{ij} and c_j given in Table 15. There are seven local minima and the global minimizer is located at $x^* = (4.00, 4.00, 4.00, 4.00)$ with $f(x^*) \approx -10.3999$.

Table 15. Data for Shekel Problem Family

	i	a_{ij}				c_i
		$j = 1$	2	3	4	
S5	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
S7	6	2	9	2	9	0.6
	7	5	5	3	3	0.3
S10	8	8	1	8	1	0.7
	9	6	2	6	2	0.5
	10	7	3.6	7	3.6	0.5

46. *Shekel 10 Problem (S10)* (Dixon and Szegö, 1978)

$$\min_x f(x) = -\sum_{j=1}^{10} \frac{1}{\sum_{i=1}^{4}(x_j - a_{ij})^2 + c_i}$$

subject to $0 \leqslant x_j \leqslant 10$, $j \in \{1, 2, 3, 4\}$ with constants a_{ij} and c_j given in Table 15. There are 10 local minima and the global minimizer is located at $x^* = (4.00, 4.00, 4.00, 4.00)$ with $f(x^*) \approx -10.5319$.

47. *Shekel's Foxholes (FX)* (Bersini et al., 1996)

$$\min_x f(x) = -\sum_{j=1}^{30} \frac{1}{c_j + \sum_{i=1}^{n}(x_i - a_{ji})^2}$$

subject to $0 \leqslant x_i \leqslant 10$, $i \in \{1, 2, \ldots, 10\}$. Our tests were performed with $n = 5$ and 10. The constants c_j and a_{ji} are given in Table 16. The number of local minima is not known, but the global minima are presented in Table 17.

Table 16. Data for Shekel's Foxholes Problem

j	c_j	a_{ji}									
		$i = 1$	2	3	4	5	6	7	8	9	10
1	0.806	9.681	0.667	4.783	9.095	3.517	9.325	6.544	0.211	5.122	2.020
2	0.517	9.400	2.041	3.788	7.931	2.882	2.672	3.568	1.284	7.033	7.374
3	0.100	8.025	9.152	5.114	7.621	4.564	4.711	2.996	6.126	0.734	4.982
4	0.908	2.196	0.415	5.649	6.979	9.510	9.166	6.304	6.054	9.377	1.426
5	0.965	8.074	8.777	3.467	1.863	6.708	6.349	4.534	0.276	7.633	1.567
6	0.669	7.650	5.658	0.720	2.764	3.278	5.283	7.474	6.274	1.409	8.208
7	0.524	1.256	3.605	8.623	6.905	4.584	8.133	6.071	6.888	4.187	5.448
8	0.902	8.314	2.261	4.224	1.781	4.124	0.932	8.129	8.658	1.208	5.762
9	0.531	0.226	8.858	1.420	0.945	1.622	4.698	6.228	9.096	0.972	7.637
10	0.876	7.305	2.228	1.242	5.928	9.133	1.826	4.060	5.204	8.713	8.247
11	0.462	0.652	7.027	0.508	4.876	8.807	4.632	5.808	6.937	3.291	7.016
12	0.491	2.699	3.516	5.874	4.119	4.461	7.496	8.817	0.690	6.593	9.789
13	0.463	8.327	3.897	2.017	9.570	9.825	1.150	1.395	3.885	6.354	0.109
14	0.714	2.132	7.006	7.136	2.641	1.882	5.943	7.273	7.691	2.880	0.564
15	0.352	4.707	5.579	4.080	0.581	9.698	8.542	8.077	8.515	9.231	4.670
16	0.869	8.304	7.559	8.567	0.322	7.128	8.392	1.472	8.524	2.277	7.826
17	0.813	8.632	4.409	4.832	5.768	7.050	6.715	1.711	4.323	4.405	4.591
18	0.811	4.887	9.112	0.170	8.967	9.693	9.867	7.508	7.770	8.382	6.740
19	0.828	2.440	6.686	4.299	1.007	7.008	1.427	9.398	8.480	9.950	1.675
20	0.964	6.306	8.583	6.084	1.138	4.350	3.134	7.853	6.061	7.457	2.258
21	0.789	0.652	2.343	1.370	0.821	1.310	1.063	0.689	8.819	8.833	9.070
22	0.360	5.558	1.272	5.756	9.857	2.279	2.764	1.284	1.677	1.244	1.234
23	0.369	3.352	7.549	9.817	9.437	8.687	4.167	2.570	6.540	0.228	0.027
24	0.992	8.798	0.880	2.370	0.168	1.701	3.680	1.231	2.390	2.499	0.064
25	0.332	1.460	8.057	1.336	7.217	7.914	3.615	9.981	9.198	5.292	1.224
26	0.817	0.432	8.645	8.774	0.249	8.081	7.461	4.416	0.652	4.002	4.644
27	0.632	0.679	2.800	5.523	3.049	2.968	7.225	6.730	4.199	9.614	9.229
28	0.883	4.263	1.074	7.286	5.599	8.291	5.200	9.214	8.272	4.398	4.506
29	0.608	9.496	4.830	3.150	8.270	5.079	1.231	5.731	9.494	1.883	9.732
30	0.326	4.138	2.562	2.532	9.661	5.611	5.500	6.886	2.341	9.699	6.500

Table 17. Global optimizers for Shekel's Foxholes Problem

n	$f(x^*)$	x^*
5	−10.4056	(8.025, 9.152, 5.114, 7.621, 4.564)
10	−10.2088	(8.025, 9.152, 5.114, 7.621, 4.564, 4.771, 2.996, 6.126, 0.734, 4.982)

48. *Sinusoidal Problem (SIN)* (Zabinsky et al., 1992)

$$\min_x f(x) = -\left[A\prod_{i=1}^{n}\sin(x_i - z) + \prod_{i=1}^{n}\sin(B(x_i - z))\right]$$

subject to $0 \leqslant x_i \leqslant 180$, $i \in \{1, 2, \ldots, n\}$. The variable x is in degrees. Parameter A affects the amplitude of the global optimum; B affects the periodicity and hence the number of local minima; z shifts the location of the global minimum; and n indicates the dimension. Our tests were performed with $A = 2.5, B = 5, z = 30$, and $n = 10$, and 20. The location of the global solution is at $x^* = (90 + z, 90 + z, \ldots, 90 + z)$ with the global optimum value of $f(x^*) = -(A + 1)$. The number of local minima increases dramatically in dimension, and when $B = 5$ the number of local minima is equal to:

$$\sum_{i=0}^{\lfloor n/2 \rfloor}\left(\frac{n!}{(n-2i)!(2i)!}3^{n-2i}2^{2i}\right).$$

49. *Storn's Tchebychev Problem (ST)* (Price, 2002)

$$\min_x f(x) = p_1 + p_2 + p_3$$

$$p_1 = \begin{cases} (u-d)^2 & \text{if } u < d \\ 0 & \text{if } u \geq d \end{cases} \qquad u = \textstyle\sum_{i=1}^{n}(1.2)^{n-i}x_i$$

$$p_2 = \begin{cases} (v-d)^2 & \text{if } v < d \\ 0 & \text{if } v \geq d \end{cases} \qquad v = \textstyle\sum_{i=1}^{n}(-1.2)^{n-i}x_i$$

$$p_3 = \textstyle\sum_{j=0}^{m} p_j', p_j' = \begin{cases} (w_j-1)^2 & \text{if } w_j > 1 \\ (w_j+1)^2 & \text{if } w_j < -1 \\ 0 & \text{if } -1 \leqslant w_j \leqslant 1 \end{cases} \qquad w_j = \textstyle\sum_{i=1}^{n}\left(\frac{2j}{m} - 1\right)^{n-i}x_i,$$

for $n = 9$: $x_i \in [-128, 128]^n$, $d = 72.661$, and $m = 60$

for $n = 17$: $x_i \in [-32768, 32768]^n$, $d = 10558.145$, and $m = 100$.

The number of local minima is not known but the global minimum is known to be as shown in Table 18.

Table 18. Global optimizers for Storn's Tchebychev Problem

n	$f(x^*)$	x^*
9	0	(128, 0, −256, 0, 160, 0, −32, 0, 1)
17	0	(32768, 0, −1331072, 0, 21299, 0, −180224, 84480, 0, −2154, 0, 2688, 0, −128, 0, 1)

50. *Wood's Function (WF)* (Michalewicz, 1996; Wolfe, 1978)

$$\min_x f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + 10.1\left[(x_2 - 1)^2 + (x_4 - 1)^2\right] + 19.8(x_2 - 1)(x_4 - 1)$$

subject to $-10 \leqslant x_i \leqslant 10,\ i \in \{1, 2, 3, 4\}$. The function has a saddle near $(1, 1, 1, 1)$. The only minimum is located at $x^* = (1, 1, 1, 1)$ with $f(x^*) = 0$.

References

Ali, M.M. and Storey, C. (1994), Modified controlled random search algorithms. *International Journal of Computer Mathematics* 54, 229–235.

Ali, M.M., Storey, C. and Törn, A. (1997), Application of some stochastic global optimization algorithms to practical problems. *Journal of Optimization Theory and Applications* 95, 545–563.

Ali, M.M. and Törn, A. (2004), Population set based global optimization algorithms: Some modifications and numerical studies. *Computers and Operations Research* 31,(10), 1703–1725.

Ali, M.M. and Törn, A. (2002), Topographical differential evolution Algorithm using pre-calculated differentials. In: Zilinskas, A., Dzemyda, G. and Saltenis, V. (eds.), *Stochastic and Global Optimization*, pp. 1–17. Kluwer Academic Publishers, Dordrecht/Boston/London.

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

Benke, K.K. and Skinner, D.R. (1991), A direct search algorithm for global optimization of multivariate functions. *The Australian Computer Journal* 23, 73–85.

Bersini, H., Dorigo, M., Langerman, S., Seront, G. and Gambardella, L. (1996), Results of the first International Contest on Evolutionary Optimization (1st ICEO). In: *Proceedings of the IEEE International Conference on Evolutionary Computation* (*ICEC*), pp. 611–615 IEEE Press, New York.

Bohachevsky, M.E., Johnson, M.E. and Stein, M.L. (1986), Generalized simulated annealing for function optimization. *Technometrics* 28, 209–217.

Breiman, L. and Cutler, A. (1993), A deterministic algorithm for global optimization, *Mathematical Programming* 58, 179–199.

Dekkers, A. and Aarts, E. (1991), Global optimization and simulated annealing. *Mathematical Programming* 50, 367–393.

Dixon, L. and Szegö, G. (1975), *Towards Global Optimization*, North Holland, New York.

Dixon, L. and Szegö, G. (1978), *Towards Global Optimization*, Vol. 2, North Holland, New York.

Dolan, D. and Moré, J. (2002), Benchmarking Optimization Software with Performance Profiles. *Mathematical Programming* Series A. 91, 201–213.

Floudas, C.A. and Pardalos, P.M. (1990), *A Collection of Test Problems for Constrained Global Optimization Algorithms*, Lecture Notes in Computer Science, Vol. 455, Springer-Verlag, Berlin/Heidelberg/New York.

Floudas, C.A., Pardalos, P.M., Adjiman, C.S., Esposito, W.R., Gümüs, Z.H., Harding, S.T., Klepeis, J.L., Meyer, C.A. and Schweiger, C.A. (1999), *Handbook of Test Problems in Local and Global Optimization*, Kluwer Academic Publishers, Dordrecht/Boston/London.

GAMS World (2002), GLOBAL Library, WWW-document, http://www.gamsworld.org/global/globallib.htm.

Goldberg, D. (1989), *Genetic Algorithm in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Company, Reading.

Gould, N.I.M, Orban, D. and Toint, P.L. (2001), CUTEr, a constrained and unconstrained testing environment, revisited, WWW-document, http://cuter.rl.ac.uk/cuter-www/problems.html.

Griewank, A.O. (1981), Generalized Descend for Global Optimization. *Journal of Optimization Theory and Applications* 34, 11–39.

Himmelblau, D.M. (1972), *Applied Nonlinear Programming*, McGraw-Hill, New York.

Jansson, C. and Knüppel, O. (1995), A Branch and Bound Algorithm for Bound Constrained Optimization Problems without Derivatives. *Journal of Global Optimization* 7, 297–331.

Khompatraporn, C., Pintér, J.D. and Zabinsky, Z.B. (2005), Comparative Assessment of Algorithms and Software for Global Optimization. *Journal of Global Optimization* 31, 613–633.

Levy, A.V. and Montalvo, A. (1985), The tunneling algorithm for the global minimization of functions. *Society for Industrial and Applied Mathematics* 6, 15–29.

McCormick, G.P. (1982), *Applied Nonlinear Programming*, *Theory*, *Algorithms and Applications*, John Wiley and Sons, New York.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H. and Teller, E. (1953), Equation of state Calculations by fast computing machines. *Journal of Chemical Physics* 21, 1087–1092.

Michalewicz, Z. (1996), *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, Berlin/Heidelberg/New York.

Moré, J., Garbow, B. and Hillstrom, K. (1981), Testing Unconstrained Optimization Software, *ACM Transaction on Mathematical Software* 7, 17–41.

Muhlenbein, H., Schomisch, S. and Born, J. (1991), The parallel genetic algorithm as function optimizer, In: Belew R. and Booker, L. (eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufman pp. 271–278.

Neumaier, A. (2003a), COCONUT benchmark, WWW-document, http://www.mat.univie.ac.at/~neum/glopt/coconut/benchmark.html

Neumaier, A. (2003b), Global and Local Optimization, WWW-document, http://solon.cma.univie.ac.at/~neum/glopt.html.

Price, W.L. (1977), Global optimization by controlled random search. *Computer Journal* 20, 367–370.

Price, W.L. (1983), Global optimization by controlled random search. *Journal of Optimization Theory and Applications* 40, 333–348.

Price, W.L. (1987), Global optimization algorithms for a CAD workstation. *Journal of Optimization Theory and Applications* 55, 133–146.

Price, K.V. (2002), *Private Communication*, 836 Owl Circle, Vacaville, CA 95687.

Romeijn, H.E. and Smith, R.L. (1994), Simulated annealing for constrained global optimization. *Journal of Global Optimization* 5, 101–126.

Salomon, R. (1995), Reevaluating genetic algorithms performance under coordinate rotation of benchmark functions, *BioSystems* 39(3): 263–278.

Schwefel, H.P. (1995), *Evolution and Optimum Seeking*, John Wiley and Sons, New York.

Second (2nd) ICEO: Second International Contest on Evolutionary Optimization, WWW-document, http://iridia.ulb.ac.be/langerman/2ndICEO.html.

Smith, R.L. (1984), Efficient Monte Carlo procedures for generating random feasible points uniformly over bounded regions. *Operations Research* 32, 1296–1308.

Storn, R. and Price, K. (1997), Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* 11, 341–359.

Törn, A. and Žilinskas, A. (1989), *Global Optimization*, Lecture Notes in Computer Science, Vol. 350, Springer-Verlag, Berlin/Heidelberg/New York.

Wolfe, M.A. (1978), *Numerical Methods for Unconstrained Optimization*, Van Nostrand Reinhold Company, New York.

Wood, G.R. and Zabinsky, Z.B. (2002), Stochastic adaptive search. In: Pardalos, P. and Romeijn, E. (eds.), *Handbook of Global Optimization*, pp. 231–249 Kluwer Academic Publishers, Dordrecht/Boston/London.

Zabinsky, Z.B. Graesser, D.L., Tuttle, M.E. and Kim, G.I. (1992), Global optimization of composite laminates using improving hit and run, In: Floudas C. and Pardalos P. (eds.), pp. 343–368. *Recent Advances in Global Optimization*, Princeton University Press.

Zabinsky, Z.B., Smith, R.L., McDonald, J.F., Romeijn, H.E. and Kaufman, D.E. (1993), Improving hit-and-run for global optimization. *Journal of Global Optimization* 3, 171–192.

Zabinsky, Z.B. and Wood, G.R. (2002), Implementation of stochastic adaptive search with hit-and-run for a generator, In: Pardolos, P. and Romeijn, E. (eds.), *Handbook of Global Optimization*, pp. 251–273. Kluwer Academic Publishers, Dordrecht/Boston/London: