Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 195 (2009) 31–45
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Continuous Optimization

Finding local optima of high-dimensional functions using direct search methods

Lars Magnus Hvattum [a,*], Fred Glover [b]

[a] Norwegian University of Science and Technology, Trondheim, Norway
[b] University of Colorado, Boulder, CO, USA

Received 11 June 2007; accepted 27 January 2008
Available online 5 February 2008

Abstract

This paper focuses on a subclass of box-constrained, non-linear optimization problems. We are particularly concerned with settings where gradient information is unreliable, or too costly to calculate, and the function evaluations themselves are very costly. This encourages the use of derivative free optimization methods, and especially a subclass of these referred to as direct search methods. The thrust of our investigation is twofold. First, we implement and evaluate a number of traditional direct search methods according to the premise that they should be suitable as local optimizers when used in a metaheuristic framework. Second, we introduce a new direct search method, based on Scatter Search, designed to remedy the lack of a good derivative free method for solving problems of high dimensions. Our new direct search method has convergence properties comparable to those of existing methods in addition to being able to solve larger problems more effectively.

Keywords: Non-linear optimization; Local minimum; Derivative free; Direct search; Scatter Search

1. Introduction

The important problem of minimizing non-linear functions can be handled in a large number of ways. Typically, techniques based on Newton's method can be applied successfully, by using gradient and Hessian information to calculate a good step and gradually move towards a (local) optimum of the function being minimized ([27] is a good introduction to these methods). However, sometimes Newton based approaches can be the wrong choice. This can be the case if: (1) the function evaluations are inaccurate, (2) the derivatives of the function are unavailable or unreliable, or (3) the function is not smooth [36,39]. In these cases, a better choice can be to rely on so-called derivative free methods, i.e., methods that do not explicitly use derivatives of the function being optimized.

Derivative free methods can be divided in two groups. One group includes methods that, instead of using the gradient directly, approximates the derivatives by building a model of the function based on function evaluations [6]. This requires that numerical function values are available, and the task of building a representative model can be difficult if the function evaluations are noisy. The second group of methods includes direct search methods, which are methods that do not try to use gradient information at all (i.e., they do not make any attempt at approximating the gradient) and that only require ordinal information about function values [22,36]. Direct search methods are hence deemed suitable for problems involving simulation-based optimization or optimizing non-numerical functions, as well as, in practice, problems involving non-smooth or discontinuous functions [19].

* Corresponding author. Tel.: +47 73593187.
E-mail addresses: lars.m.hvattum@iot.ntnu.no (L.M. Hvattum), fred.glover@colorado.edu (F. Glover).

0377-2217/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2008.01.039

In this paper, the focus is on the use of direct search methods to find local optima. Many direct search methods are simple in nature, and only undertake to find a local optimum (close to the starting point of the search). Thus, they are often used in hybrid methods for global optimization, e.g., as in [4,5,13–15,30]. However, quite often the choice of which direct search method to use is left unjustified, or, else only a few alternative methods are examined. At the same time, one of the most well-known and extensively used methods, the Nelder–Mead [26], can perform remarkably poorly both from a theoretical and practical point of view [12,22].

One issue that is often mentioned in discussions about direct search methods (as well as other derivative free methods), is their tendency to perform well only on problems with few dimensions and to falter when the number of dimensions grows [1,6,17,19,36]. This motivates the work presented here, in which different direct search methods (as well as two other derivative free methods) are evaluated based on their ability to quickly find local optima of high-dimensional functions.

All test problems used in the computational experiments presented later in this paper can be seen as instances of the following model:

$$\min_{x\in[\ell,u]\subseteq\Re^n} f(x), \tag{1}$$

where $f:\Re^n\to\Re$, and, $\ell,u\in\Re^n$. Strictly speaking, however, in accordance with the definition of direct search methods proposed in [22], several variations of this problem can be tackled. For example, the objective function need not be numerical (i.e., one could have $f(x):\Re^n\to \mathrm{R}$, seeking to determine x to minimize $|\{y: f(y)\leqslant f(x)\}|$, where $\leqslant$ defines an order on R) In addition, the functions that are optimized can contain noise. None of the solution methods examined in this work are specifically adapted to exploit box-constraints ($x\in[\ell,u]$), although we make reference to such constraints as a convenient foundation for generating an initial solution point. For the purpose of this paper, the formulation (1) will thus suffice, even though, unless otherwise stated, the methods discussed are applicable in the wider context mentioned above.

The remainder of this paper is as follows. In Section 2 we examine several traditional and some lesser known direct search methods (as well as two other derivative free methods – one mainly because it is claimed to be effective for problems with many variables). Then, in Section 3 we develop a direct search based on ideas from Scatter Search [9,20] and conduct computational tests that demonstrate the efficacy of our method in this setting. Finally, conclusions and suggestions for future research are discussed in Section 4.

2. Traditional direct search methods

Our goal in this section is to examine and evaluate eight different methods (see Table 1) according to their ability to find local optima of functions having many dimensions. The motivation for this is that these methods are frequently used as subsolvers in various global optimization methods, and their efficiency in finding local optima quickly is of interest to practitioners that implement such global methods. Table 2 gives some references for each method (their original source and/or other references where the method is used or discussed), and their classification within the realm of direct search methods. See [22] for one suggested taxonomy of direct search methods. The parameters defined for these methods are given the standard values found in the literature. Note that two of the methods, HPS and SPSA, are not labeled as direct search methods since they require a numerical objective function, and are thus less general than the other methods.

For some of these solution methods there exists a theory of convergence. Early work can be found in [38], where the author proves that the limit of the infimum of the norm of the gradient at the best vertex at iteration k converges to 0 as $k\to\infty$ for MDS, CS, and HJ, given that the function is continuously differentiable. Although this does not apply to the functions for which direct search methods are the preferred approach [19], the assumption of continuous differentiability sometimes holds in restricted areas of the function domain at hand. In addition, convergence analysis of these methods go much further, and a hierarchy of results based on the degree of smoothness can be found in [2]. At the other end of the scale some of the direct search methods have negative convergence results, such as NM, [12]. In this paper, the emphasis

Table 1
Direct search and other methods from the literature

Method	Full name
NM	Nelder–Mead
MDS	Multi-directional search
CS	Compass search
HJ	Hooke and Jeeves
ROS	Rosenbrock's algorithm (with improvements by Palmer)
SW	Solis and Wets' algorithm
HPS	Heuristic pattern search
SPSA	Simultaneous perturbation stochastic approximation

Table 2
Classification of the methods considered

Methods	References	Classifications
NM	[26,39]	Simplex search
MDS	[37,39]	Simplex search/pattern search
CS	[7,19]	Pattern search/generating set algorithm
HJ	[7,16]	Pattern search
ROS	[28,29]	Method with adaptive search directions
SW	[32,30]	Stochastic direct search
HPS	[13]	Derivative free method (not a direct search)
SPSA	[33,34]	Derivative free method (not a direct search)

is not on the theoretical convergence properties of the different methods, but rather on their performance as established by empirical evaluation.

To test the methods, a set of 12 well-known test functions has been selected, as well as two additional functions (given in Table 3). They have the property that all minima have the same value (the optimal value), which makes it easier to perform comparisons between the solution methods. In a situation where a method may converge to local optima with different values, it is necessary to decide whether it is preferable to converge quickly to a poor local optimum or slowly to a good local optimum. Function 44 in Table 3 has been modified (*) as described in Appendix A. Function 51 in Table 3 (**) has a modified box to avoid a second local optimum. The new box is $[-4, 1.5] \times [-2, 4]$ for the two-dimensional case. Functions 60 and 61 (***) are described in Appendix A.

The drawback of using the most well-known test functions, is that the functions usually lack certain properties that would make them interesting for derivative free methods (and direct search methods in particular). To be specific, they have known derivatives, and there is no noise, discontinuities, or other features that make the use of direct search methods pertinent. Still, these functions are among the ones that are almost always used when comparing even such methods as are implemented here. Notably, only functions 44, 60, and 61, out of which the latter two are formulated especially for this paper, have discontinuities or an undefined gradient. However, one would expect that the performance of the direct search methods does not depend too much on whether the functions are smooth, and have defined gradients, since this information is not used by the solution methods. Our subsequently reported computational findings support this expectation. On the other hand, the performance of Newtonian methods as well as other methods relying on derivative information might be expected to suffer, though we do not know of any tests of this proposition.

For the purpose of this work, some of the functions are modified to generalize into functions of higher dimensions. This is accomplished by introducing higher-dimensional terms that take the form of the lower-dimensional terms of the original function, by extending their definition to encompass a higher dimension. For instance, for the classical (two-dimensional) Rosenbrock function

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2, \tag{2}$$

the multi-dimensional generalization becomes

Table 3
List of functions

No.	Name	Source
1	Branin	[21]
7	Booth	[21]
8	Matyas	[21]
11	Rosenbrock	[21]
12	Zakharov	[21]
24	Trid	[21]
28	SumSquares	[21]
39	Sphere	[21]
41	Schwefel's Problem 1.2 (F2)	[35]
42	Rotated High Conditioned Elliptic Function (F3)	[35]
44	Modified Schwefel's Problem 2.6 (F5)	[35] (*)
51	McCormic	[24] (**)
60	Staircased Rosenbrock	(***)
61	Staircased LogAbs	(***)

$$f(x) = \sum_{i=1}^{n/2} 100(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1})^2, \tag{3}$$

where $n = 2k$ for k integer. Note that for the Rosenbrock function in particular, another generalization is sometimes proposed that does not retain the property of having only one local optimum [31].

For the ease of conducting computational experiments, all the functions are shifted so that the optimum has a value of 0, and one can thus use a stopping criterion in which the search is terminated whenever an x is found such that $f(x) < \epsilon$, where ϵ is taken, as is customary in the literature, to be 0.001. In addition, a limit on the number of function evaluations is set to 50,000, terminating the search when this number of evaluations is reached.

2.1. Computational results

We performed computational tests using the eight methods of Table 1 on the 14 functions in Table 3, with the number of dimensions ranging from 2 to 512. Each combination of method/function/size is repeated 10 times with different (random) starting points. All methods are initialized by first creating a simplex of points. For $\delta = \min_i(u_i - \ell_i)/s$, where s is a parameter, an initial x is created by letting $x_i = \ell_i + \delta\chi$, where χ is a random factor uniform on $[1, 2]$. Then, n additional points are created by setting $x_j^i = x_j + \delta$ for $i = j$ and $x_j^i = x_j$ otherwise. The value δ also coincides with the initial step-length used by each method, and in the computational tests, a value of $s = 4$ is used.

Results for two functions are presented in detail, Table 4 showing results on the Trid function, and Table 5 showing results on the Sphere function (further results are presented in Section 3). The results are presented as follows. Given 10 repetitions for each setting, the average number of iterations before convergence is reported. In the case where at least one repetition fails to produce a converged point, the ratio of successful repetitions is given in parentheses. One can observe that the methods have very different behavior on the two functions, but some trends can be extracted when looking at the full set of 14 test functions.

Table 6 summarizes the results on each function by giving the methods that yield the best results, as well as the methods that follow most closely. The categorization is approximate, and methods giving similar results are grouped together.

First, we can note that the results for the Nelder–Mead method are appalling. Only on rare occasions does NM succeed in locating local optima of functions that have more than four dimensions (the Sphere-function in Table 5 being an exception). Only for one function (the Rosenbrock) is NM close to being among the best methods (mainly due to the fact that almost all methods struggle on this particular function). The other simplex search in this test, MDS, performs much better, yet is still to be found among the worst methods on most functions. As a positive note about NM, it is indeed often the best method for the case of two-dimensional functions. This supports the findings in [15], in which NM is found to be better than SW for simple functions when used as an improvement strategy in a Scatter Search.

Two other methods are noticeably absent from Table 6, namely the two derivative free methods that fall outside the definition of direct search methods. The HPS, in general, performs moderately well. It is, however, always worse than at least two other methods in the test. The SPSA procedure, which is claimed to be suitable for high-dimensional problems, suffers from the problem that some of the parameters of the method are sufficiently esoteric to make practical exploitation difficult [34]. Although theoretical results about the method are good [33], in the tests conducted here, the method failed to perform well.

One more comment deserves to be made about Table 6, which reinforces the motivation for our development of a Scatter Search based method in Section 3. In all but one case, either CS or ROS is among the best performing methods (the notable exception being the Sphere function in Table 5, for which SW seems well suited). However, the two methods, CS and ROS, seem to perform well for different reasons, and typically, one of them is relatively weak when the other is strong, and vice versa.

Additional insight into the different methods can be gained by altering the starting conditions for each search. Table 7 shows the effect of changing the size of the initial simplex and the initial step length, δ, where increasing values of the

Table 4
Results for function 24, Trid, with the number of variables ranging from 2 to 64

n	NM	MDS	CS	HJ	ROS	SW	HPS	SPSA
2	36.0	72.6	40.3	53.5	39.0	54.0	88.9	3005.7
4	750.5	969.8	172.2	247.9	165.2	178.2	211.6	32171.9
8	(0.0)	14409.0	1230.4	1498.0	646.0	1802.1	5633.0	(0.0)
16	(0.0)	(0.0)	8811.5	8981.8	3041.8	16053.8	(0.6)	(0.0)
32	(0.0)	(0.0)	(0.0)	(0.0)	14499.4	(0.0)	(0.0)	(0.0)
64	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)

Table 5
Results for function 39, Sphere, with the number of variables ranging from 2 to 512

n	NM	MDS	CS	HJ	ROS	SW	HPS	SPSA
2	67.2	31.0	36.3	52.3	25.3	55.3	65.2	767.1
4	283.2	126.6	95.8	129.5	64.9	89.0	104.1	901.1
8	(0.9)	524.2	223.4	284.3	156.0	172.3	177.9	1092.6
16	(0.2)	2385.0	618.4	618.4	371.8	373.5	406.3	1277.3
32	(0.0)	10535.4	1107.2	1256.3	1082.8	784.8	1062.2	1652.4
64	(0.0)	46657.0	2440.1	2784.0	(0.9)	1602.2	3847.2	2536.7
128	(0.1)	(0.0)	5128.7	5642.5	(0.7)	3515.3	18226.9	5170.5
256	(0.0)	(0.0)	11113.2	12565.9	(0.3)	7401.3	(0.0)	(0.0)
512	(0.0)	(0.0)	23070.3	25148.6	(0.0)	15770.9	(0.0)	(0.0)

Table 6
Summary of the best methods for each function

Function	Winner	Runner-up
1	CS	HJ, SW
7	CS, SW	HJ
8	CS, SW	HJ
11	ROS	CS, SW
12	ROS	SW
24	ROS	CS, HJ
28	CS, HJ	ROS, SW
39	SW	CS, HJ
41	ROS	MDS
42	CS, HJ	MDS
44	CS	HJ
51	CS, HJ	SW
60	ROS	CS, HJ
61	CS	HJ

Table 7
Effect of changing the starting conditions, on the function Sphere, with $n = 4$

s	NM	MDS	CS	HJ	ROS	SW	HPS	SPSA
2	241.6	176.2	97.7	97.3	88.8	104.6	127.1	1561.3
4	283.2	126.6	95.8	129.5	64.9	89.0	104.1	901.1
6	227.4	125.8	79.7	120.3	64.7	77.0	87.8	791.0
8	107.1	152.6	81.9	127.3	65.7	93.4	98.3	1016.9
10	193.3	146.3	80.1	122.8	71.8	85.3	103.1	1120.7
20	314.2	183.3	75.1	105.1	79.3	96.1	130.9	1301.9
50	945.2	198.4	93.8	76.9	58.4	101.7	224.7	1381.1
100	312.4	140.4	152.9	129.9	98.9	112.2	419.6	1406.9
200	732.4	198.8	270.5	62.8	138.7	121.3	813.4	1430.2
500	1010.1	212.4	669.1	171.6	174.0	125.4	2002.8	1436.2
1000	1272.5	211.6	1334.1	203.0	140.1	140.0	4002.1	1440.4
2000	(0.8)	228.0	2662.9	187.8	195.8	145.7	7982.2	1441.3
5000	2141.8	185.0	6650.4	324.0	128.5	152.8	19949.8	1441.6
10000	1241.4	214.0	13296.8	409.0	95.6	157.8	39888.1	1441.1
20000	1846.2	219.0	26588.8	583.0	200.0	177.5	(0.0)	1432.5
50000	2250.6	230.0	(0.0)	914.0	101.0	180.9	(0.0)	1432.3

parameter s yield smaller step lengths (and thus, in a way, yield starting points that are further away from a local optimum). Clearly, some methods (such as MDS, HJ, ROS, SW, and SPSA) are better at adapting to the changes and at adjusting the step length. For the other methods, the initial guess of step length is more critical, and the user must make sure that a good initial value is found.

As a final note in this section, the effect of increasing the number of problem dimensions has a much larger impact than a similar change in the error tolerance, ϵ, used in the stopping criterion. One might suspect that doubling the number of dimensions produces an outcome similar to halving the tolerance for error, since most of the functions examined here are separable to a certain extent. However, increasing the number of dimensions poses fundamentally different challenges

Table 8
Effect of adjusting the error tolerance, on the function McCormic, with $n = 2$

ϵ	NM	MDS	CS	HJ	ROS	SW	HPS	SPSA
10^{0}	4.9	6.0	8.4	7.9	8.5	9.3	9.9	84.6
10^{-1}	9.0	14.0	15.3	17.6	12.7	14.6	22.9	238.8
10^{-2}	14.8	28.2	26.2	32.2	21.6	32.7	37.0	467.7
10^{-3}	22.7	36.6	33.8	41.1	31.7	44.9	55.8	807.0
10^{-4}	31.3	53.0	43.4	53.5	47.0	63.5	84.9	1257.9
10^{-5}	38.3	66.2	56.7	67.8	55.9	77.4	103.0	1820.4
10^{-6}	46.5	84.6	67.8	82.2	64.1	99.1	124.2	2527.2
10^{-7}	55.3	99.4	77.1	93.2	69.2	114.2	148.0	3362.4

to the direct search methods. Table 8 illustrates that the effect of reducing the error tolerance is not very severe compared to the effect of increasing the number of variables, as is done in Tables 4 and 5.

3. Scatter Search

The computational study of some well-known direct search methods in the previous section motivates the development of alternative methods. It was observed that both CS and ROS were able to produce good results compared to the other methods, but on different problem instances and under different conditions. The aim is now to take advantage of the ideas found in the metaheuristic called Scatter Search, by joining them with our findings from the computational tests on classical direct search methods, to create a method that can successfully find local optima of functions having a large number of dimensions. Scatter Search has already been tested as a global optimizer for box-constrained, non-linear functions in [21], but only on functions of up to 30-dimensions. Also, Scatter Search has been used in combination with various direct search methods employed as improvement strategies, e.g. in [15], but again only on functions of 30 or fewer dimensions. In contrast, our study presented here focuses on local optima only, and on functions of many variables. The variations of our new Scatter Search method combine convergence theory for direct search methods [38] with state-of-the-art knowledge from heuristic solution methods [8] in order to produce an efficient and robust method for handling high-dimensional functions.

A pseudo code description of our Scatter Search method, is given below as Procedure 1. As a prelude, we discuss some of the key features of the method.

The initialization of Procedure 1 uses the same values as supplied to every other method tested here, using the set of initial points consisting of the simplex described in Section 2 and the initial step length, δ, as specified in the creation of the simplex. The method has two main parts, one that corresponds to the traditional view of Scatter Search, in steps 3–13, and one that is mostly a variation of CS (steps 16–19), but specifically adapted to assist the Scatter Search portion.

Part 1 of the method proceeds as follows. A pool of solutions is maintained, containing every solution examined so far, with the option to discard solutions permanently if they are too far from the current best solution (where "too far" is defined as a multiple of δ). In step 4, a reference set is built as follows. Having isolated the best point in the pool, x^b, filter the remaining solutions in the pool, focusing only on a number, b_c, of solutions that are closest to x^b. Among these, first select the b_1 best solutions to include in the reference set. Then, among the remaining solutions, select the b_2 solutions that have recently been found to produce good solutions when used in the subset combination of step 7. This is achieved by storing, for each solution, x, a number of the most recent solutions created in step 7 when x was a member of the subset. Taking the average rank in the pool for the recent solutions created by using x, we obtain a measure of how well x can support the creation of new good solutions, and the b_2 best solutions based on this measure is included in the reference set.

Procedure 1. Scatter Search

1: Create a set of initial points.
2: **while** stopping criterion not met **do**
3: Isolate the best point found so far, x^b.
4: Build a reference set containing points that are close to x^b, and, that are either good points themselves or that have been used to create good points in step 7
5: Select a subset of the reference set, using either Procedure 2 or Procedure 3.
6: Let $\bar{x}$ denote the centroid of the subset selected in step 5.
7: Find a new point, x^1 that lies δ units from x^b, in a direction from $\bar{x}$ through x^b.
8: **if** x^1 is better than x^b **then**
9: Find a new point x^2 that lies 2δ units from x^b, in a direction from $\bar{x}$ through x^b.
10: In the case that x^2 is better than x^1, increase δ and go to step 14.

11: Find a new point x^3 that lies 0.5δ units from x^b, in a direction from $\bar{x}$ through x^b.
12: In the case that x^3 is better than x^1, decrease δ.
13: **end if**
14: If an improvement of x^b has been found, go to step 3, if a limit on the number of subsets to generate has not been exceeded, go to step 5.
15: **if** x^b is still the best point found **then**
16: **for** a number of steps $\leqslant 2n$ **do**
17: Extend x^b in a cartesian direction (iteratively cycling through each of $2n$ directions), for a length that depends on the direction and is limited by δ.
18: If an improvement of x^b is found in step 17, x^b is replaced. The direction-dependent length used in step 17 is adjusted based on whether or not an improvement was found.
19: **end for**
20: **end if**
21: If no improvement of x^b was found during steps 7–20, δ is reduced.
22: **end while**

In order to create new solutions, we must select a subset from the reference set. Two different ways of selecting subsets are outlined in the supporting Procedures 2 and 3, where Procedure 2 relies on randomization, and Procedure 3 is based on clustering the reference set using the k-clustering algorithm [23].

In the first approach, a subset is created in two steps: first, the size of the subset is decided, and second, solutions from the reference set are selected at random until the required subset size is obtained. The size of the subset is selected using biased randomization, where the probabilities of the different sizes are adjusted dynamically based on search history. Specifically, the most recent solutions created in step 7 are stored, differentiated by the size of the subsets used to produce them, and the different subset sizes are ranked according to the average rank of the solutions produced by subsets of the indicated size (i.e., using the same measure for ranking as when selecting the b_2 last solutions to enter the reference set).

The second approach creates subsets simply by dividing the points of the reference set into k clusters. The k-clustering algorithm we employ is a heuristic that partitions the reference set, such that the sum of distances between each point and the center of its cluster becomes small. The clusters are then ranked by the quality of the points that they contain, and the clusters are selected as subsets sequentially. The number of clusters must be at least as large as the number of subsets generated by each update of the reference set.

Note that for both subset generation approaches, the best solution, x^b, is implicitly included in the reference set and in every subset.

The combination of the subset elements to produce new solutions follows a strategy that contrasts with an idea used in NM. Instead of improving the worst solution in a set (in NM the set is a simplex), the aim here is to improve the best solution in the set. Therefore, in step 7, a ray is taken which intersects the centroid of the subset (x^b not included) and x^b. For a specific subset $\mathbf{S}$, the new point x^1 is

$$x^1 = x^b + \delta\left(x^b - \sum_{x\in\mathbf{S}} \frac{x}{|\mathbf{S}|}\right) \Bigg/ \left\| x^b - \sum_{x\in\mathbf{S}} \frac{x}{|\mathbf{S}|} \right\|. \tag{4}$$

Procedure 2. Randomized Subset Generation

1: Let the possible subset sizes be labeled from s_1 to s_M.
2: Sort the sizes according to previous performance (i.e., average rank of points generated in step 7 of Procedure 1).
3: **while** subset size has not been selected **do**
4: Cyclically select the next size from the sorted list of sizes, starting with the best ranked size.
5: Reject the current size with a given probability, p^{SS}
6: **end while**
7: Let s_i denote the selected subset size, and start with an empty subset.
8: **while** subset contains less than s_i points **do**
9: Add a random member of the reference set to the subset.
10: **end while**

Procedure 3. Clustering Based Subset Generation

1: **if** this subset generation procedure has not been executed since the last update of the reference set **then**
2: Execute the k-clustering algorithm in order to divide the points of the reference set in k clusters.
3: Sort the k clusters according to the quality of each cluster (i.e., the average rank of the points in the cluster).
4: **end if**
5: If this is the ith time that a subset has been requested since the last time that the reference set was updated, let the subset be equal to the ith best cluster, according to rank.

If the new point x^1 successfully improves on x^b, additional search along the same ray may be warranted. Hence, in steps 9 and 11, two solutions x^2 and x^3 are created in a similar fashion, further exploring the ray. Also, steps 10 and 12 adjust the step length δ by increasing δ if x^2 is better than x^1 or reducing δ if x^3 is better than x^1.

Unless x^1 successfully improves x^b, the search continues for a given number of steps by selecting different subsets, using the same reference set. However, if x^1 is successful, the reference set is rebuilt before continuing the search. This type of dynamic update of the reference set derives from the study of [25] where it was found to speed convergence towards a local optimum.

Now consider the case where the Scatter Search has not yet found an improvement of x^b after a predetermined number of subset combinations. At this point it is natural to decrease the step length, δ and continue. However, a second part of the search comes into play first. Part 2 closely resembles CS, in that it searches only in directions corresponding to the cartesian unit vectors, and in that it dynamically adjusts the step length for each of the directions considered ($2n$ directions are included: the positive and negative direction of each dimension). The main motivation behind this part, not considering the fact that CS itself is among the best direct search methods examined, is that the range of directions which can possibly be generated using the steps 5–7 may at some point become severely reduced. As a rule, several successful combinations are made consecutively along (approximately) the same direction. In consequence, any subset produced will then have a centroid that lies very close to this direction from x^b. In order to allow other search directions to arise, it accordingly becomes necessary to generate new solutions that are close to x^b but lie in a varied selection of directions from x^b.

Notice that the close resemblance to CS of Part 2 in the proposed method provides a way to exploit the convergence theory from [38]. Although steps 5–7 of Procedure 1 suggests using step lengths that depend on the direction, if this recommendation is ignored the analysis of [38] applies directly. Our empirical tests suggest though, that different step lengths for different directions are beneficial. It is perhaps interesting that the convergence analysis in [38] can be used to understand the behavior of some of the direct search methods when the number of dimensions increase. The theory shows that the bound on the angle between the search directions and the gradient may deteriorate as the number of dimensions of the problem increases. Part 1 of the Scatter Search can be seen as an attempt to counter this phenomenon by using metaheuristic, population-based ideas to generate alternative search directions. At the same time, Part 2 is designed both to feed Part 1 with new alternative directions and to secure some properties of convergence. Also note that Part 2 of the proposed method effectively takes the role of an improvement method, as customarily found within the Scatter Search paradigm. The novelty here though, is that this improvement method is applied very selectively, and only to the current best solution.

Another issue worth mentioning, is that there is always some danger of reproducing solutions (or creating solutions very close to previously examined solutions). Therefore, no solution is evaluated if it is closer than $\alpha\delta$ to a previously evaluated solution, where *alpha* is a parameter satisfying $0 < \alpha \ll 1$ and δ is the current step length. We suggest that such strategies similarly provide improvements for other methods when function evaluations are costly (as in [7] for HJ). See also [11] for alternatives based on tabu search.

One disadvantage of using the Scatter Search framework is that usually the method has more parameters to be tuned than the simpler classical methods. However, an advantage of using a direct search/Scatter Search method in the context considered here is that the problem of tuning parameters can be treated as an instance of the same general non-linear optimization problem we are studying. In other words, we treat the tuning problem as that of minimizing a function that simulates the application of Scatter Search to a given set of test problems, taking the parameters to be tuned as variables of the function. Using this "bootstrapping" idea suggested in [10], we formulate a function whose value is determined by the number of evaluations required by our method to solve a small set of selected problem instances, and whose variables correspond to the parameters of the Scatter Search. Upon starting the Scatter Search with some initial guess of parameters, Procedure 1 can then be used to determine its own parameters. In this case, there was a total of 10 parameters (some additional parameters were fixed to the same values as used in the traditional direct search methods), and the Scatter Search (using Procedure 2 to create subsets) quickly found settings vastly better than, and very different from, the initial parameters.

The outcomes found by this bootstrapping approach can be summarized as follows. Interestingly, it proved best to select the reference set simply as the 360 points closest to the best point, x_b, i.e., $b_c = b_1 = 360$ and $b_2 = 0$. According to the parameters found, no points need to be selected based on their use as generators in step 7 of Procedure 1. The maximum number of subsets used for making combinations in steps 5–13 is 3. If no improvement is found after this, the search continues by sequentially examining each of the $2n$ directions allowed in steps 16–19. The factor used in the updates of δ was

fixed at 2, the same value as used in the majority of the other direct search methods. Using these parameters, we examined the use of Procedure 3 for generation of subsets. Results indicated that a value of $k = 6$ works well, and this was used in the computational testing described in Section 3.1.

3.1. Computational results

Using the parameters found by the bootstrap procedure as described above, the Scatter Search procedure was tested on the same functions as the other methods. Table 9 shows an example where the new Scatter Search approaches, SSR (the version with randomized subset generation, using Procedure 2) and SSC (the version with clustering based subset generation, using Procedure 3), are remarkably successful, giving results that are much better than any of the other methods. The table only reports results for the direct search methods, since as previously mentioned, neither HPS nor SPSA produces results that are competitive against the best direct search methods. Table 10 gives an exceptional example where SSR and SSC are not very successful. However, there is only one other method that ever converges within 50,000 evaluations when the number of dimensions is equal to 64 for this problem.

It is additionally interesting to note that both versions of the Scatter Search adaptations of direct search, SSR and SSC, exhibit very similar behavior on almost all of the test functions. The only two exceptions to this are shown in Tables 11 and 12, and in both cases the clustering based approach appears to be more robust than the randomized subset generation. For the Zakharov function (see Table 12) the clustering based subset selection similarly performs better, but notably only when the number of dimensions exceeds 8. With respect to the Rosenbrock function (see Table 11), the results are more mixed, with SSR producing best results for the function with 64-dimensions, and SSC giving preferable results for dimensions from 2 to 32.

Table 13 is an update of Table 6, with SSR and SSC being included. Clearly, the Scatter Searches are among the top contenders for every function, only being surpassed by Rosenbrock's method in 3 out of the 14 functions, and being the clear winner in nine cases. Moreover, the dominance of SSR and SSC on certain functions is significant. For example, on functions 1 (Branin), 28 (SumSquares), and 42 (Rotated High Conditioned Elliptic Function, F3), SSR and SSC solve

Table 9
Results for function 1, Branin, with the number of variables ranging from 2 to 512

n	NM	MDS	CS	HJ	ROS	SW	SSR	SSC
2	43.8	57.2	51.6	67.9	44.3	71.1	57.1	59.4
4	224.0	188.4	116.0	171.5	116.6	143.5	102.8	103.3
8	(0.0)	663.7	290.5	394.8	282.9	341.1	197.1	202.8
16	(0.0)	3015.7	680.5	979.9	647.8	928.9	406.6	374.2
32	(0.0)	11516.6	1550.7	(0.9)	1542.2	2032.8	779.7	783.2
64	(0.0)	46751.7	3628.4	(0.9)	6544.6	4558.6	1636.6	1626.4
128	(0.0)	(0.0)	8031.5	(0.8)	28666.7	9922.3	3448.8	3443.7
256	(0.0)	(0.0)	18172.5	(0.6)	(0.1)	24043.6	7173.4	7184.6
512	(0.0)	(0.0)	(0.9)	(0.7)	(0.0)	(0.2)	15206.9	15221.8

Table 10
Results for function 41, Shifted Schwefel's Problem 1.2 (F2), with the number of variables ranging from 2 to 128

n	NM	MDS	CS	HJ	ROS	SW	SSR	SSC
2	48.8	57.4	79.4	102.2	71.2	106.2	90.8	89.2
4	351.4	218.6	324.3	459.3	255.8	243.6	322.4	349.0
8	(0.0)	870.9	1708.7	2685.2	711.1	1404.5	1404.1	1313.1
16	(0.0)	3435.0	7842.5	10980.0	2428.1	6109.4	5085.8	4280.8
32	(0.0)	14710.8	34937.3	(0.7)	7822.9	24328.7	15512.2	14880.8
64	(0.0)	(0.0)	(0.0)	(0.0)	25145.7	(0.0)	(0.2)	(0.1)
128	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)

Table 11
Results for the two Scatter Search variations on function 11, Rosenbrock, with the number of variables ranging from 2 to 128

n	SSR	SSC
2	606.5	313.2
4	13410.0	2256.4
8	25831.6	8165.6
16	(0.9)	12500.4
32	(0.9)	21397.7
64	39193.2	(0.2)
128	(0.0)	(0.0)

Table 12
Results for the two Scatter Search variations on function 12, Zakharov, with the number of variables ranging from 2 to 128

n	SSR	SSC
2	49.2	49.3
4	168.1	173.2
8	1060.9	1101.1
16	6827.8	4030.5
32	32626.4	16167.1
64	(0.0)	(0.1)
128	(0.0)	(0.0)

Table 13
Summary of the best methods, including SSR and SSC, for each function

Function	Winner	Runner-up
1	SSR, SSC	CS
7	SSR, SSC	CS, SW
8	SSR, SSC	CS, SW
11	ROS, SSR, SSC	CS, SW
12	ROS	SSC
24	ROS	SSR, SSC
28	SSR, SSC	CS, HJ
39	SSR, SSC	SW
41	ROS	MDS, SSR, SSC
42	SSR, SSC	CS, HJ
44	SSR, SSC	CS
51	SSR, SSC	CS, HJ
60	ROS, SSR, SSC	CS, HJ
61	SSR, SSC	CS

the 512-dimensional problem using fewer function evaluations than required by the second best method (CS) to solve the much smaller 256-dimensional problem.

4. Conclusions and future research

The study of this paper makes two main contributions. First, we conduct a computational examination of several existing derivative free optimization methods, with a particular focus on direct search methods. Our objective is to investigate the behavior on high-dimensional problems when only a local optimum is sought. The results reveal that some well-known and frequently used methods have severe limitations, especially when the functions at hand have many dimensions. We also found that two of the simplest methods, the Compass Search method and Rosenbrock's method (with the improvement by Palmer, [28]), seem to perform relatively well, under the assumptions present.

Second, we developed a new direct search/Scatter Search method based on combining insights from a computational study of traditional direct search methods with ideas from the Scatter Search metaheuristic. The new method achieved significantly improved results for the majority of the functions tested, and obtained results comparable to those obtained by the best previous method for the other functions.

Our study appears to be the first to demonstrate the merit of using clustering based subset generation techniques in Scatter Search. The quality of the results obtained motivates further examination of the processes for embedding clustering within Scatter Search, both for continuous and discrete optimization. Each of the two variations of the Scatter Search we implemented embody theoretical convergence properties, but our computational results show that such theoretical properties are not necessarily meaningful in practice. In particular, although each of CS, HJ, and MDS have been shown to have similar convergence properties [38], the latter performs overall much worse than the other two methods.

Some limitations of this work should be noted. Due to the vast number of direct search methods described in the literature, it is difficult to carry out extensive studies that include all the variations. Hence, some recent developments are not considered here. Among these is the Mesh Adaptive Direct search [3] which is claimed to extend and improve on Generalized Pattern Search. MDS, CS, and HJ constitute the pattern search methods considered in the study in Section 2. However, in [1] the authors observe that their approach falters when large numbers of decision variables are present. Also, we did not consider parallel versions of the direct search methods, such as Parallel Pattern Search. However, the authors of [17] state that Parallel Pattern Search is intended to be useful for problems with a small number of variables, in the range from 10 to 50.

Similarly, methods that primarily focus on global optimization are not included. A prominent example is the DIRECT (DIviding RECTangles) method, although there is likewise an indication that this method may not be suited for high-dimensional problems [18].

One reason for omitting several of the newer versions of pattern search is that they are not in common use as subsolvers for global optimization approaches. Although they may have several interesting properties in terms of theoretical convergence, they do not yet have the same widespread acceptance by practitioners based on demonstrated merit in actual performance. Our work demonstrates that several of the commonly used direct search methods have severe limitations in terms of finding local optima of high-dimensional problems, and that alternatives should be considered. Beyond this, we have shown that one can combine features of the methods having established convergence theory with well-known ideas from metaheuristic research to create robust and highly efficient hybrid methods.

Future research will undoubtedly profit by a more thorough examination of the balance between intensification and diversification of the search, as by drawing on strategies proposed in tabu search for exploiting the tradeoffs between these interrelated functions. Within the present context we anticipate that a variant of a multi-start strategy can prove useful, by performing iteratively deeper Scatter Search runs, starting from points that are filtered based on merit and on proximity to previously explored regions of the search space.

Acknowledgement

We thank three anonymous referees, whose comments have led to improvements in this paper.

Appendix A

We now state the details of functions 44, 60, and 61. Function 44, F5 from [35], is taken as

$$f(x) = \max_i \{|\mathbf{A}_i x|\}, \tag{5}$$

where $x \in [-50, 100]^n$ and $\mathbf{A}$ is taken to be the $n \times n$ upper left submatrix of $\mathbf{B}$ being generated using Matlab as follows:

`rand('state',0); B = round(rand(512)*1000 - 500).`

Function 60 is as follows:

$$f(x) = g(x) + r(x - \mathbf{1}), \tag{6}$$

where $r(x)$ is the Rosenbrock function (3), and

$$g(x) = \sum_{i=0}^{n} \lfloor |x_i| \rfloor, \tag{7}$$

whereas function 61 is

$$f(x) = g(x) + \sum_{i=0}^{n} \log_2(x_i + 1) \tag{8}$$

and where $x \in [-5, 10]^n$ for both Function 60 and 61. Each of the functions 44, 60, and 61 have a single local optimum at $x^* = [0, \ldots, 0]$ with a value of $f(x^*) = 0$.

Appendix B

Here follows complete results for all functions and all methods tested (Tables 14–27).

Table 14
Results for function 1, Branin

n	NM	MDS	CS	HJ	ROS	SW	HPS	SPSA	SSR	SSC
2	43.8	57.2	51.6	67.9	44.3	71.1	88.6	1142.7	57.1	59.4
4	224.0	188.4	116.0	171.5	116.6	143.5	155.2	1406.9	102.8	103.3
8	(0.0)	663.7	290.5	394.8	282.9	341.1	272.9	1538.4	197.1	202.8
16	(0.0)	3015.7	680.5	979.9	647.8	928.9	934.9	1926.5	406.6	374.2
32	(0.0)	11516.6	1550.7	(0.9)	1542.2	2032.8	(0.8)	2819.7	779.7	783.2
64	(0.0)	46751.7	3628.4	(0.9)	6544.6	4558.6	5664.5	(0.7)	1636.6	1626.4
128	(0.0)	(0.0)	8031.5	(0.8)	28666.7	9922.3	21469.0	(0.0)	3448.8	3443.7
256	(0.0)	(0.0)	18172.5	(0.6)	(0.1)	24043.6	(0.0)	(0.0)	7173.4	7184.6
512	(0.0)	(0.0)	(0.9)	(0.7)	(0.0)	(0.2)	(0.0)	(0.0)	15206.9	15221.8

Table 15
Results for function 7, Booth

n	NM	MDS	CS	HJ	ROS	SW	HPS	SPSA	SSR	SSC
2	35.2	56.5	71.4	106.9	51.3	75.4	103.8	957.9	63	70.2
4	316.3	177.6	189.5	274.6	150.7	189.8	172.8	1183.1	203.1	192.7
8	(0.3)	760.3	491.3	640.7	387.0	510.9	525.2	1701.6	422.8	413
16	(0.0)	3217.2	1163.3	1505.7	915.3	1284.6	(0.9)	2660.8	809.4	737.2
32	(0.0)	13946.4	2643.8	2872.2	2512.3	2878.5	(0.7)	5373.6	1732.4	1436.9
64	(0.0)	(0.0)	6040.0	6304.9	9873.0	6148.3	(0.7)	13896.5	3154.8	3148.5
128	(0.0)	(0.0)	13403.9	16967.0	(0.8)	13015.3	(0.5)	41173.0	6737.8	7516.3
256	(0.0)	(0.0)	29754.9	41187	(0.0)	27799.6	(0.0)	(0.0)	16293.8	16915.3
512	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	36840.3	38635.4

Table 16
Results for function 8, Matyas

n	NM	MDS	CS	HJ	ROS	SW	HPS	SPSA	SSR	SSC
2	35.5	221.4	57.2	71.4	46.2	59.7	98.1	(0.1)	60.3	47.6
4	282.5	999.4	185.4	236.6	141.5	240.3	316.8	(0.0)	209.9	167.2
8	(0.0)	5031.4	479.3	659.7	334.0	582.4	791.0	(0.0)	432.4	380.0
16	(0.0)	21069.7	1273.0	1816.3	899.5	1589.0	2503.4	(0.0)	1109.8	902.1
32	(0.0)	(0.0)	2926.0	4376.6	2356.6	3614.0	5329.8	(0.0)	2422.8	1834.6
64	(0.0)	(0.0)	6961.2	9928.1	6351.4	8433.8	20274.1	(0.0)	3854.2	3827.1
128	(0.0)	(0.0)	15601.1	22486.5	30923.4	18173.9	(0.0)	(0.0)	8575.0	9687.9
256	(0.0)	(0.0)	35836.5	(0.5)	(0.0)	39372.8	(0.0)	(0.0)	24207.0	24903.1
512	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)

Table 17
Results for function 11, Rosenbrock

n	NM	MDS	CS	HJ	ROS	SW	HPS	SPSA	SSR	SSC
2	466.4	(0.8)	17083.2	(0.7)	237.8	9434.1	(0.0)	(0.0)	606.5	313.2
4	2290.3	(0.0)	(0.8)	(0.4)	1125.0	25580.4	(0.1)	(0.0)	13410.0	2256.4
8	(0.0)	(0.0)	(0.4)	(0.0)	2761.9	(0.2)	(0.0)	(0.0)	25831.6	8165.6
16	(0.0)	(0.0)	(0.0)	(0.0)	7820.2	(0.0)	(0.0)	(0.0)	(0.9)	12500.4
32	(0.0)	(0.0)	(0.0)	(0.0)	21183.1	(0.0)	(0.0)	(0.0)	(0.9)	21397.7
64	(0.0)	(0.0)	(0.0)	(0.0)	(0.1)	(0.0)	(0.0)	(0.0)	39193.2	(0.2)
128	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)

Table 18
Results for function 12, Zakharov

n	NM	MDS	CS	HJ	ROS	SW	HPS	SPSA	SSR	SSC
2	53.4	43.8	46.9	69.8	35.9	70.7	87.3	779.7	49.2	49.3
4	339.1	161.0	174.9	260.0	134.3	161.7	150.7	5867.0	168.1	173.2
8	(0.2)	1037.7	1535.9	1930.1	477.6	630.8	7050.5	(0.3)	1060.9	1101.1
16	(0.0)	10872.5	21579.7	26317.6	1784.2	3217.7	(0.7)	(0.0)	6827.8	4030.5
32	(0.0)	(0.0)	(0.0)	(0.0)	7696.0	21876.6	(0.0)	(0.0)	32626.4	16167.1
64	(0.0)	(0.0)	(0.0)	(0.0)	28022.2	(0.0)	(0.0)	(0.0)	(0.0)	(0.1)
128	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)

Table 19
Results for function 24, Trid

n	NM	MDS	CS	HJ	ROS	SW	HPS	SPSA	SSR	SSC
2	36.0	72.6	40.3	53.5	39.0	54.0	88.9	3005.7	39.5	43.0
4	750.5	969.8	172.2	247.9	165.2	178.2	211.6	32171.9	166.0	152.7
8	(0.0)	14409.0	1230.4	1498.0	646.0	1802.1	5633.0	(0.0)	676.3	563.5
16	(0.0)	(0.0)	8811.5	8981.8	3041.8	16053.8	(0.6)	(0.0)	4357.8	3447.5
32	(0.0)	(0.0)	(0.0)	(0.0)	14499.4	(0.0)	(0.0)	(0.0)	(0.8)	(0.3)
64	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)

Table 20
Results for function 28, SumSquares

n	NM	MDS	CS	HJ	ROS	SW	HPS	SPSA	SSR	SSC
2	42.7	39.0	43.1	61.6	33.3	63.5	88.7	846.6	41.4	40.4
4	405.9	169.8	119.3	157.7	102.5	117.4	130.8	832.1	77.7	81.2
8	(0.4)	692.2	292.6	362.6	247.6	347.5	391.5	846.6	146.3	146.2
16	(0.0)	3255.4	683.8	798.1	743.1	1262.9	(0.4)	2463.8	298.3	313.0
32	(0.0)	15233.0	1543.3	1746.3	2507.4	4751.7	(0.2)	11616.7	622.5	647.5
64	(0.0)	(0.0)	3457.0	3791.5	15871.6	18346.8	(0.0)	(0.0)	1369.0	1348.4
128	(0.0)	(0.0)	7572.9	8245.4	(0.0)	(0.0)	(0.3)	(0.0)	2840.7	2808.8
256	(0.0)	(0.0)	16411.9	17771.5	(0.0)	(0.0)	(0.0)	(0.0)	6057.1	6105.4
512	(0.0)	(0.0)	35319.0	38177.1	(0.0)	(0.0)	(0.0)	(0.0)	12866.2	13021.4

Table 21
Results for function 39, Sphere

n	NM	MDS	CS	HJ	ROS	SW	HPS	SPSA	SSR	SSC
2	67.2	31.0	36.3	52.3	25.3	55.3	65.2	767.1	39.4	34.0
4	283.2	126.6	95.8	129.5	64.9	89.0	104.1	901.1	63.8	63.3
8	(0.9)	524.2	223.4	284.3	156.0	172.3	177.9	1092.6	114.7	120.6
16	(0.2)	2385.0	618.4	618.4	371.8	373.5	406.3	1277.3	227.2	232.8
32	(0.0)	10535.4	1107.2	1256.3	1082.8	784.8	1062.2	1652.4	462.5	463.2
64	(0.0)	46657.0	2440.1	2784.0	(0.9)	1602.2	3847.2	2536.7	958.0	983.9
128	(0.1)	(0.0)	5128.7	5642.5	(0.7)	3515.3	18226.9	5170.5	1979.4	2111.7
256	(0.0)	(0.0)	11113.2	12565.9	(0.3)	7401.3	(0.0)	(0.0)	4236.8	4258.6
512	(0.0)	(0.0)	23070.3	25148.6	(0.0)	15770.9	(0.0)	(0.0)	8890.4	8890.2

Table 22
Results for function 41, Shifted Schwefel's Problem 1.2 (F2)

n	NM	MDS	CS	HJ	ROS	SW	HPS	SPSA	SSR	SSC
2	48.8	57.4	79.4	102.2	71.2	106.2	137.0	10072.2	90.8	89.2
4	351.4	218.6	324.3	459.3	255.8	243.6	368.4	18630.8	322.4	349.0
8	(0.0)	870.9	1708.7	2685.2	711.1	1404.5	(0.7)	23913.9	1404.1	1313.1
16	(0.0)	3435.0	7842.5	10980.0	2428.1	6109.4	(0.4)	(0.8)	5085.8	4280.8
32	(0.0)	14710.8	34937.3	(0.7)	7822.9	24328.7	(0.0)	(0.0)	15512.2	14880.8
64	(0.0)	(0.0)	(0.0)	(0.0)	25145.7	(0.0)	(0.0)	(0.0)	(0.2)	(0.1)
128	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)

Table 23
Results for function 42, Shifted Rotated High Conditioned Elliptic Function (F3)

n	NM	MDS	CS	HJ	ROS	SW	HPS	SPSA	SSR	SSC
2	91.5	118.2	129.1	196.6	256.4	(0.0)	(0.0)	(0.0)	133.4	160.7
4	(0.9)	421.8	291.1	419.4	4046.7	(0.0)	(0.0)	(0.0)	201.3	218.0
8	(0.0)	1596.2	666.3	918.3	(0.9)	(0.0)	(0.0)	(0.0)	346.0	370.6
16	(0.0)	6129.0	1422.7	1848.3	(0.0)	(0.0)	(0.0)	(0.0)	612.9	610.8
32	(0.0)	24353.0	3027.9	3855.6	(0.0)	(0.0)	(0.0)	(0.0)	1193.3	1177.1
64	(0.0)	(0.0)	6292.1	7809.5	(0.0)	(0.0)	(0.0)	(0.0)	2283.1	2301.1
128	(0.0)	(0.0)	12865.8	15981.4	(0.0)	(0.0)	(0.0)	(0.0)	4650.3	4643.6
256	(0.0)	(0.0)	26436.7	32546.0	(0.0)	(0.0)	(0.0)	(0.0)	9441.1	9441.1
512	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	18557.7	18557.7

Table 24
Results for function 44, Modified Schwefel's Problem 2.6 with Global Optimum on Bounds (F5)

n	NM	MDS	CS	HJ	ROS	SW	HPS	SPSA	SSR	SSC
2	160.3	142.2	151.9	215.6	129.6	222.7	307.5	(0.0)	131.8	130.4
4	(0.0)	(0.9)	376.6	504.7	355.5	370.6	(0.4)	(0.0)	226.1	229.4
8	(0.0)	(0.2)	874.1	1057.1	1355.7	(0.0)	(0.0)	(0.0)	392.9	388.9
16	(0.0)	(0.5)	2040.2	2492.3	(0.9)	(0.0)	(0.0)	(0.0)	771.3	776.7
32	(0.0)	(0.8)	4455.5	5533.3	(0.0)	(0.0)	(0.0)	(0.0)	1528.3	1541.9
64	(0.0)	(0.0)	8676.9	9873.5	(0.0)	(0.0)	(0.0)	(0.0)	3068.5	3097.8
128	(0.0)	(0.0)	18732.2	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	6879.1	6937.4
256	(0.0)	(0.0)	45051.3	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	13638.4	13722.7
512	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	29280.9	28671.2

Table 25
Results for function 51, McCormic

n	NM	MDS	CS	HJ	ROS	SW	HPS	SPSA	SSR	SSC
2	22.7	36.6	33.8	41.1	31.7	44.9	55.8	807.0	36.8	34.3
4	352.4	156.0	89.5	99.0	78.0	92.8	(0.9)	911.0	73.8	78.2
8	(0.0)	631.8	232.4	216.7	184.1	201.5	220.6	1321.6	145.4	161.4
16	(0.0)	3202.8	550.8	497.5	408.3	(0.6)	518.4	2076.5	316.9	312.3
32	(0.0)	14871.3	1266.2	1227.3	(0.9)	(0.8)	1589.7	4769.1	641.3	645.0
64	(0.0)	(0.0)	2880.2	2659.1	(0.1)	(0.7)	(0.6)	16487.6	1386.5	1387.4
128	(0.0)	(0.0)	6616.1	6021.1	(0.8)	(0.7)	(0.7)	(0.1)	2909.1	2953.1
256	(0.0)	(0.0)	13894.7	12119.1	(0.0)	(0.4)	(0.0)	(0.0)	6383.6	6196.6
512	(0.0)	(0.0)	30634.1	27184.6	(0.0)	(0.0)	(0.0)	(0.0)	13614.7	13419.2

Table 26
Results for function 60, Staircased Rosenbrock

n	NM	MDS	CS	HJ	ROS	SW	HPS	SPSA	SSR	SSC
2	(0.9)	(0.5)	14244.6	13582.7	206.4	(0.9)	(0.2)	(0.0)	462.8	282.1
4	(0.3)	(0.1)	(0.8)	(0.4)	964.4	(0.3)	(0.0)	(0.0)	15370.1	3774.3
8	(0.0)	(0.0)	(0.3)	(0.1)	3118.2	(0.0)	(0.0)	(0.0)	26864.6	6722.2
16	(0.0)	(0.0)	(0.0)	(0.0)	9391.3	(0.0)	(0.0)	(0.0)	31978.3	13575.7
32	(0.0)	(0.0)	(0.0)	(0.0)	24789.4	(0.0)	(0.0)	(0.0)	26315.3	22706.1
64	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	37090.5	(0.0)
128	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)

Table 27
Results for function 61, Staircased LogAbs

n	NM	MDS	CS	HJ	ROS	SW	HPS	SPSA	SSR	SSC
2	83.3	86.6	83.9	119.6	72.0	120.0	(0.9)	(0.0)	76.3	72.9
4	(0.0)	326.6	209.1	271.9	202.6	(0.4)	(0.9)	(0.0)	127.2	118.3
8	(0.0)	1524.2	498.7	605.7	(0.9)	(0.0)	(0.0)	(0.0)	230.4	225.4
16	(0.0)	5674.6	1107.5	1290.8	(0.3)	(0.0)	(0.0)	(0.0)	441.8	445.8
32	(0.0)	25313.0	2412.6	2725.7	(0.0)	(0.0)	(0.0)	(0.0)	893.7	896.3
64	(0.0)	(0.0)	5193.9	5742.2	(0.0)	(0.0)	(0.2)	(0.0)	1858.8	1861.0
128	(0.0)	(0.0)	11069.2	12152.9	(0.0)	(0.0)	(0.0)	(0.0)	3880.2	3899.9
256	(0.0)	(0.0)	23445.1	25588.7	(0.0)	(0.0)	(0.0)	(0.0)	8070.3	8212.7
512	(0.0)	(0.0)	49372.9	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	17180.3	16947.2

References

[1] M.A. Abramson, C. Audet, J.E. Dennis Jr., Nonlinear programming by mesh adaptive direct searches, SIAG/Optimization Views-and-News 17 (2006) 2–11.
[2] C. Audet, J.E. Dennis Jr., Analysis of generalized pattern searches, SIAM Journal on Optimization 13 (2003) 889–903.
[3] C. Audet, J.E. Dennis Jr., Mesh adaptive direct search algorithms for constrained optimization, SIAM Journal on Optimization 17 (2006) 188–217.
[4] R. Chelouah, P. Siarry, Genetic and Nelder–Mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions, European Journal of Operational Research 148 (2003) 335–348.
[5] R. Chelouah, P. Siarry, A hybrid method combining continuous tabu search and Nelder Mead simplex algorithms for the global optimization of multiminima functions, European Journal of Operational Research 161 (2005) 636–654.
[6] A.R. Conn, K. Scheinberg, Ph.L. Toint, Recent progress in unconstrained nonlinear optimization without derivatives, Mathematical Programming 79 (1997) 397–414.
[7] E.D. Dolan, Pattern Search Behaviour in Nonlinear Optimization. Thesis, 1999.
[8] F. Glover, G.A. Kochenberger (Eds.), Handbook of Metaheuristics, Kluwer, Boston, 2003.
[9] F. Glover, M. Laguna, R. Martí, Fundamentals of scatter search and path relinking, Control and Cybernetics 29 (2000) 653–684.
[10] F. Glover, Future paths for integer programming and links to artificial intelligence, Computers and Operations Research 13 (1986) 533–549.
[11] F. Glover, Tabu search for nonlinear and parametric optimization (with links to genetic algorithms), Discrete Applied Mathematics 49 (1994) 231–255.
[12] L. Han, M. Neumann, Effect of dimensionality on the Nelder–Mead simplex method, Optimization Methods and Software 21 (2006) 1–16.
[13] A. Hedar, M. Fukushima, Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization, Optimization Methods and Software 19 (2004) 291–308.
[14] A. Hedar, M. Fukushima, Tabu search directed by direct search methods for nonlinear global optimization, European Journal of Operational Research 170 (2006) 329–349.

[15] F. Herrera, M. Lozano, D. Molina, Continuous scatter search: An analysis of the integration of some combination methods and improvement strategies, European Journal of Operational Research 169 (2006) 450–476.
[16] R. Hooke, T.A. Jeeves, Direct search solution of numerical and statistical problems, Journal of the Association for Computing Machinery 8 (1961) 212–229.
[17] P.D. Hough, T.G. Kolda, V.J. Torczon, Asynchronous parallel pattern search for nonlinear optimization, SIAM Journal on Scientific Computing 23 (2001) 134–156.
[18] D.R. Jones, C.D. Perttunen, B.E. Stuckman, Lipschitzian optimization without the Lipschitz constant, Journal of Optimization Theory and Applications 79 (1993) 157–181.
[19] T.G. Kolda, R.M. Lewis, V.J. Torczon, Optimization by direct search: New perspectives on some classical and modern methods, SIAM Review 45 (2003) 385–482.
[20] M. Laguna, R. Martí, Scatter Search: Methodology and Implementations, C. Kluwer Academic Publishers, 2003.
[21] M. Laguna, R. Martí, Experimental testing of advanced scatter search designs for global optimization of multimodal functions, Journal of Global Optimization 33 (2005) 235–255.
[22] R.M. Lewis, V.J. Torczon, M.W. Trosset, Direct search methods: Then and now, Numerical Analysis 2000, vol. 4, Elsevier, 2001, pp. 191–207.
[23] J.B. MacQueen, Some methods for classification and analysis of multivariate observations, in: L.M. LeCam, N. Neyman (Eds.), Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, Berkeley, 1967, pp. 281–297.
[24] K. Madsen, J. Zilinskas, Testing real and interval methods for global optimization. Technical Report IMM-2000-05, Institute of Mathematical Modelling, Technical University of Denmark, 2000.
[25] R. Martí, M. Laguna, F. Glover, Principles of scatter search, European Journal of Operational Research 169 (2006) 359–377.
[26] J.A. Nelder, R. Mead, A simplex method for function minimization, The Computer Journal 7 (1965) 308–313.
[27] J. Nocedal, S.J. Wright, Numerical optimization, Springer Series in Operations Research and Financial Engineering, second ed., Springer, New York, 2006.
[28] J.R. Palmer, An improved procedure for orthogonalising the search vectors in Rosenbrock's and Swann's direct search optimisation methods, The Computer Journal 12 (1969) 69–71.
[29] H.H. Rosenbrock, An automatic method for finding the greatest or least value of a function, The Computer Journal 3 (1960) 175–184.
[30] C.D. Rosin, R. Scott Halliday, W.E. Hart, R.K. Belew, A comparison of global and local search methods in drug docking, in: Thomas Bäck (Ed.), Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97), Morgan Kaufmann, San Francisco, CA, 1997, pp. 221–228.
[31] Y.-W. Shang, Y.-H. Qiu, A note on the extended Rosenbrock function, Evolutionary Computation 14 (2006) 119–126.
[32] F.J. Solis, R.J.-B. Wets, Minimization by random search techniques, Mathematical Operations Research 6 (1981) 19–30.
[33] J.C. Spall, Multivariate stochastic approximation using a simultaneous perturbation gradient approximation, IEEE Transactions on Automatic Control 37 (1992) 332–341.
[34] J.C. Spall, Implementation of the simultaneous perturbation algorithm for stochastic optimization, IEEE Transactions on Aerospace and Electronic Systems 34 (1998) 817–823.
[35] P.N. Suganthan, N. Hansen, J.J. Jiang, K.Deb, Y.-P. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Technical Report, Nanyang Technological University, Singapore, 2005.
[36] V.J. Torczon, M.W. Trosset, From evolutionary operation to parallel direct search: Patter search algorithms for numerical optimization, Computing Science and Statistics 29 (1998) 396–401.
[37] V.J. Torczon, *Multi-Directional Search: A Direct Search Algorithm for Parallel Machines*, PhD thesis, Rice University, 1989.
[38] V.J. Torczon, On the convergence of the multidirectional search algorithm, SIAM Journal on Optimization 1 (1991) 123–145.
[39] M.H. Wright, Direct search methods: Once scorned, now respectable, in: D.F. Griffiths, G.A. Watson (Eds.), Numerical Analysis 1995, Addison Wesley Longman, Harlow, United Kingdom, 1996, pp. 191–208.