Restarted Local Search Algorithms for Continuous Black-Box Optimization

Petr Pošík posik@labe.felk.cvut.cz
Faculty of Electrical Eng., Czech Technical University in Prague, Czech Republic

Waltraud Huyer Waltraud.Huyer@univie.ac.at
Faculty of Mathematics, University of Wien, Austria

Abstract
Several local search algorithms for real-valued domains (axis-parallel line search, Nelder-Mead simplex search, Rosenbrock's algorithm, quasi-Newton method, NEWUOA and VXQR) are described and thoroughly compared in this article, embedding them in a multistart method. Their comparison aims (1) to help the researchers from the evolutionary community to choose the right opponent for their algorithm (to choose an opponent that would constitute a hard-to-beat baseline algorithm), (2) to describe individual features of these algorithms and show how they influence the algorithm on different problems, and (3) to provide inspiration for the hybridization of evolutionary algorithms with these local optimizers. The recently proposed Comparing Continuous Optimizers (COCO) methodology was adopted as the basis for the comparison. The results show that in low dimensional spaces, the old method of Nelder and Mead is still the most successful among those compared, while in spaces of higher dimensions it is better to choose an algorithm based on quadratic modeling, like NEWUOA or a quasi-Newton method.

Keywords
Real-parameter optimization, continuous domain, black-box optimization, benchmarking, local optimization, multistart method, line search, Nelder-Mead simplex search, quasi-Newton method.

1 Introduction

Local search algorithms still constitute a very popular class of optimization problem solvers. Often, they are easy to use, easy to understand and even easy to construct. In the black-box scenario, they are usually the first choice when an experimenter needs to solve her problem.

Evolutionary algorithms (EAs) are not primarily used to solve unimodal problems, for which the local search algorithms are suitable. On the other hand, unimodal problems (1) often constitute basic test cases that should ensure that the EA does not fail on a trivial function, and (2) can be made hard for a large subset of EAs, e.g. by making them ill-conditioned. Moreover, although a particular EA is effective on unimodal problems, it is usually not sufficient; we want it to be also efficient, i.e. to be similarly fast as a good local optimizer (and of course, to be also effective and hopefully efficient on multimodal problems). The real-valued EAs should thus be compared to "good" exemplars of local optimizers.

The results of this article can also be of high interest for the designers of memetic algorithms, MAs (Moscato, 1989). MAs are hybrids between EAs and local search techniques, which often exhibit both the robustness of EAs and the speed of local search

methods at the same time. It is an important asset to know which local search algorithm is suitable for a particular situation. This article compares several local search methods and shows which of them are "good" under which circumstances.

Seven different algorithms for real-valued black-box optimization are compared.[1] The first two fall into the class of the axis-parallel line search, LS (Whitley et al., 1996). They differ in the method used for the univariate search along the axes. One of them, LSfminbnd, uses the MATLAB function `fminbnd`, a univariate bounded local search method. The second one, LSstep, uses the STEP procedure (Swarzberg et al., 1994), a univariate bounded global search method.

The third method was proposed by Rosenbrock (1960). His algorithm, RA, expresses a certain amount of similarity to the line search: it also searches along some perpendicular directions in the space. However, in RA the steps in individual directions alternate and the directions are adaptive.

The next method, valley exploration based on QR factorization, VXQR1 (Neumaier et al., 2010), is rather recent. Line searches along adaptive perpendicular directions are also part of this method, similarly to RA. VXQR1 also adds quadratic modeling of the objective function and a certain amount of stochastic elements which should help it to prevent getting stuck in local optima.

The above mentioned 4 algorithms are described and their results are discussed in detail since we performed the experiments with them ourselves. We chose three other algorithms for this comparison as a reference. For the reference algorithms, we use the results obtained by others and we thus do not have such experience with those methods. Consequently, their description focuses on their main principles only, and their discussion is limited.

BFGS, a restarted quasi-Newton method with the BFGS update formula (as implemented in the MATLAB function `fminunc`), and NEWUOA (Powell, 2006), a recent successful local search method, were selected since—similarly to VXQR1—they also model the objective function by a quadratic model, of course in a different way. The last method in the comparison is the well-known Nelder-Mead simplex search method, NMSS (Nelder and Mead, 1965). It differs from the above mentioned algorithms in that it maintains a population of points (while the others use only a single point possibly complemented with certain kind of model of its neighborhood).

The COCO (Comparing Continuous Optimizers) methodology (Hansen et al., 2009a) was chosen as the tool for the comparison. The framework is able to show the differences among the algorithms at all stages of the search, not just after a certain number of evaluations, as is the usual practice. It was used as the basis of the Black-Box Optimization Benchmarking (BBOB) workshops of the GECCO-2009 and 2010 conferences. The testbed consists of 24 carefully chosen noiseless benchmark functions (Hansen et al., 2009b) which represent various types of difficulties observed in real-world problems (ill-conditioning, multimodality, etc.). The dimension of the search space varied from 2 to 40 during the experiments.

The results of the algorithms obtained using the COCO framework were already separately presented as the workshop papers (Pošík, 2009b,a; Ros, 2009a,b; Hansen, 2009). The results for VXQR1 are, however, presented for the first time and constitute one of the original contributions of this article. Another goal of this paper is to collect

[1] In fact, we also experimented with the algorithm of Solis and Wets (1981). The best results obtained from it were for $D \geq 5$ surprisingly quite coincident with the results of both LS methods, when aggregated over all problems. We decided not to include the algorithm in the comparison due to its bad results and due to the fact that the 8th algorithm would only clutter the ECDF graphs.

the results of all the above mentioned algorithms, compare them conveniently in one place, provide a discussion of the pros and cons of the algorithms compared to each other, and suggest the successful exemplars of local search methods. In the above mentioned original papers, the discussion (if any) was based solely on the results of the respective algorithm and no comparison was made. We also discuss the results in more detail than the summary paper of Hansen et al. (2010).

There are some prior works that attempt to systematically compare various direct search methods, e.g. Schwefel (1995). These comparisons, however, differ in the set of chosen optimizers and in the set of benchmark problems. Even for the benchmark functions which the studies have in common with the BBOB function set, a direct comparison of the result is questionable. The COCO framework transforms individual design variables with smooth non-linear monotonic functions to break the symmetries observed in many benchmark functions. It also uses the functions in a rotated and shifted form. All these features make the functions effectively different and render the results incomparable.

The rest of the article is organized as follows. After describing the algorithms in Sec. 2, the experimental framework in Sec. 3, and the experiment and algorithm parameter settings in Sec. 4, the article continues with the presentation of the benchmarking results in Sec. 5 and discusses them in Sections 6 and 7. The discussion is broken down by the individual function groups and by the algorithms, respectively. The article is summarized and concluded in Sec. 8.

2 Local Search Algorithms

The algorithms used in the comparison are described in this section. In the experiments, they are embedded into the multistart method, i.e. they are restarted after they detect slow progress, or after the budget for a single run is exhausted. The following subsections describe the LS, RA, and VXQR1 algorithms in detail, and briefly also the reference algorithms, BFGS, NEWUOA, and NMSS.

2.1 Axis-Parallel Line Search

The line search algorithm is one of the basic and simplest optimization algorithms. In any comparison, it should serve as a baseline algorithm (Whitley et al., 1996). The axis-parallel line search is effective and often also efficient for separable functions. The results of the line search algorithm should thus indicate those test functions which are (nearly) separable or functions which are easy for algorithms exploiting separability. The line search algorithm is not expected to be effective for non-separable functions.

The algorithm starts from a randomly selected point. Then it iterates through individual directions and optimizes the function with respect to the chosen direction, keeping the other solution components fixed. After the optimization of one direction, it moves to the best solution found and switches to another direction. If the solution does not change after going through all the directions, the algorithm finishes since a local optimum (with respect to its neighborhood) was found.

In this paper, two multistart versions of the axis-parallel line search method are considered and compared; each trial begins in a different initial point chosen uniformly from the search space. The two versions differ in the used univariate optimization technique: the MATLAB function `fminbnd`, and the STEP algorithm are used. Both multivariate optimization algorithms based on the two above-mentioned univariate procedures are *not invariant with respect to the search space rotation*, and both algorithms also directly use the objective function values when deciding where to sample a new point,

thus are *not invariant with respect to the order-preserving transformations of the objective function*.

2.1.1 Line Search with `fminbnd`

The MATLAB `fminbnd` function (revision 1.18.4.11 was used) is based on the golden-section search and parabolic interpolation. It is able to identify the optimum of quadratic functions in a few steps. On the other hand, it is a local search technique, it can miss the global optimum (of the 1D function). Since this is a rather standard ready-to-use algorithm, it will not be described here in more detail.

2.1.2 Line Search with STEP

The acronym STEP stands for *select the easiest point*. The STEP method (Swarzberg et al., 1994) is a univariate global search algorithm based on interval division. It starts from one interval initialized with x_l and x_u, lower and upper bound of the interval, with both points evaluated. In each iteration, it selects one interval and divides it to halves by sampling and evaluating the point in the middle.

The STEP algorithm selects the interval used to sample the next point based on the interval *difficulty*, i.e. by its belief how difficult it would be to improve the best-so-far solution by sampling from the respective interval. The measure of the interval difficulty chosen in STEP is the value of the coefficient a from the quadratic function $f(x) = ax^2 + bx + c$ which goes through both interval boundary points and somewhere on the interval reaches the value of $f_{\text{best}} - \epsilon$ (ϵ is a small positive number, typically from 10^{-3} to 10^{-8}, meaning an improvement of f_{best} by a nontrivial amount). An example of a few STEP iterations can be seen in Fig. 1.

2.2 Rosenbrock's method

The Rosenbrock algorithm, RA (Rosenbrock, 1960), is a classical local search technique for unconstrained black-box optimization. It maintains the best-so-far solution and searches in its neighborhood for improvements. What distinguishes this algorithm from many other local search techniques is the fact that it also maintains a model of the current local neighborhood: it adapts the model orientation and size. This feature can be observed in many recent successful optimization techniques, e.g. in CMA-ES (Hansen and Ostermeier, 2001).

The RA local search technique is depicted as Alg. 1. The model of the local neighborhood consists of D vectors $\mathbf{e}_1, \ldots, \mathbf{e}_D$ forming the orthonormal basis, and of D multipliers (or step lengths) $d_1, \ldots, d_D$, where D is the dimensionality of the search space. In each iteration, the algorithm performs a kind of pattern line search along the directions given by the orthornormal basis. If in one direction $\mathbf{e}_i$ an improvement is found, next time (after trying all other directions) a point α times farther in that direction is sampled; if no improvement is found in the $\mathbf{e}_i$ direction, next time a closer point on the other side is sampled (governed by the β parameter). Usually, the values of parameters are $\alpha = 2$ and $\beta = \frac{1}{2}$.

As soon as at least one successful and one unsuccessful move in each direction was carried out, the algorithm updates its orthonormal basis to reflect the cumulative effect of all successful steps in all directions. It also resets the multipliers to their original values (not used in the current implementation). The update of the orthonormal basis is done using Palmer's orthogonalization method (Palmer, 1969) so that the first basis vector is always parallel to the last vector $\mathbf{x} - \mathbf{x}_0$.

The demonstration of the RA behavior on the 2-D sphere and the 2-D Rosenbrock function can be seen in Fig. 2.

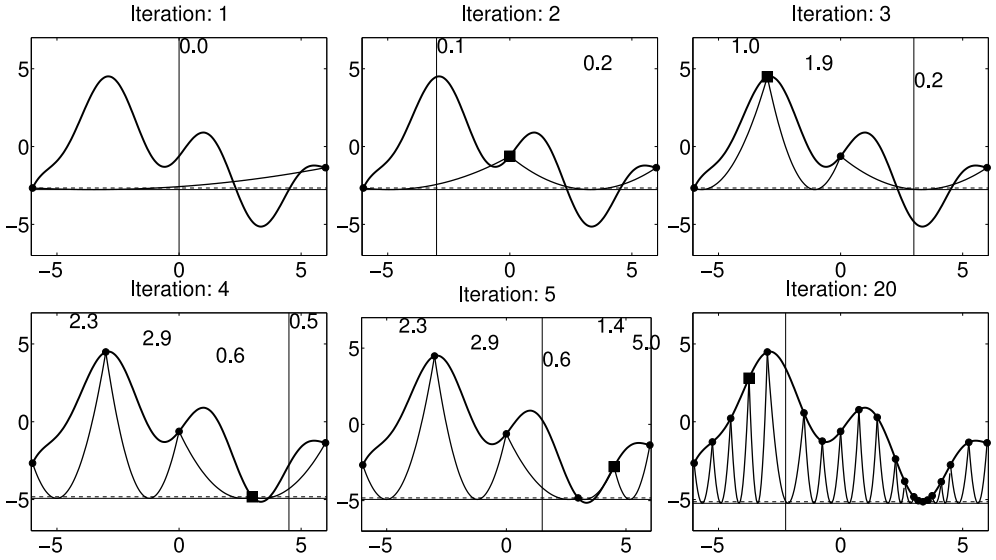

Figure 1: Demonstration of the behavior of the STEP algorithm. Bold line: the objective function. Dots: previously sampled data points, interval boundaries. Dashed horizontal line: f_{best} level. Solid horizontal line: $f_{\text{best}} - \epsilon$ level. Parabolic curves: start and end in the interval boundaries and touch the objective level $f_{\text{best}} - \epsilon$. Big square: last sampled data point. Vertical line: the place where the next point will be sampled. Numbers: the difficulty indices of the respective intervals—the coefficients a of the respective parabolas.

To improve the performance on multimodal functions, a restarting strategy was used. Each restart begins with an initial point uniformly chosen from the search space. The original RA resets the multipliers d_i at each stage of the algorithm (line 14 of Alg. 1), i.e. each time the orthonormal basis is updated. In the particular implementation used in this article, the multipliers are not reset. It was observed (Pošík, 2010) that this modification improves the results on many—mostly low-dimensional—benchmark problems; it spares some function evaluations needed to adapt the multipliers at each stage. Instead, it converges faster, allowing for more algorithm restarts.

With the exception of initialization, the algorithm is *invariant with respect to translation and rotation*. The algorithm is also *invariant with respect to order-preserving transformations of the objective function* since it uses only comparisons between two individuals.

2.3 Valley Exploration Based on QR Factorization (VXQR)

Based on the results of the BBOB-2009 comparison (Hansen et al., 2010), Neumaier et al. (2010) developed the class of VXQR algorithms (*valley exploration based on QR factorizations*) for bound-constrained optimization problems with the aim to preserve the advantages of the multilevel coordinate search, MCS (Huyer and Neumaier, 1999), in case of a low budget or low dimensions while improving the performance in case of a generous budget and in high dimensions. The main features of MCS, a successful initialization strategy and local searches building quadratic models, developed into the scout phase and the subspace phase of the new algorithm, respectively. The deterministic multilevel strategy of dividing boxes was replaced by stochas-

Algorithm 1: Rosenbrock's Algorithm

Input: $\alpha > 1$, $\beta \in (0, 1)$, the initial point $\mathbf{x}$.

begin
  $\mathbf{x}_o \leftarrow \mathbf{x}$
  Init. the multipliers $\{d_1, \ldots, d_D\}$ and the orthonorm. basis $\{\mathbf{e}_1, \ldots, \mathbf{e}_D\}$.
  while the termination condition is not satisfied **do**
    for i=1... D **do**
      $\mathbf{y} \leftarrow \mathbf{x} + d_i \mathbf{e}_i$
      if $\mathbf{y}$ is better than $\mathbf{x}$ **then**
        $\mathbf{x} \leftarrow \mathbf{y}$
        $d_i \leftarrow \alpha \cdot d_i$
      else
        $d_i \leftarrow -\beta \cdot d_i$
    if at least one success and one failure in all directions **then**
      Update the orthonormal basis $\{\mathbf{e}_1, \ldots, \mathbf{e}_D\}$ using vector $\mathbf{x} - \mathbf{x}_o$.
      Re-initialize multipliers $\{d_1, \ldots, d_D\}$.
      $\mathbf{x}_o \leftarrow \mathbf{x}$

tic techniques in order to prevent getting stuck in a non-global minimum. VXQR1 is a particular realization of this class of algorithms tuned to yield good results on a set of test problems. The MATLAB code of VXQR1 can be downloaded from `http://www.mat.univie.ac.at/~neum/software/vxqr1/`. In the following paragraphs, we present the main characteristics of the algorithm.

The VXQR1 algorithm uses two kinds of line searches: the global and the local line search. Both contain random elements and form the stochastic part of the algorithm.

The *global line search* has two parts. In the first part, it tries to improve the current best point by evaluating the function at 10 new points (5 on each side of the best point) on a random subsegment of a line along the search direction. This procedure is executed as long as an improvement of the best point is found, but at most 10 times. In the second part, a further improvement of the best solution is sought by safeguarded quadratic and piecewise linear interpolation steps made from the local minimizers among the 11 points from the last iteration of the first part.

The *local line search* evaluates the function at a random point along the search direction. Then it obtains the third point by reflecting the worse point at the better point. Then 4 more points are generated, always from the current best point, by choosing from several methods (safeguarded quadratic interpolation, geometric mean step, and piecewise linear interpolation) according to heuristic criteria.

The VXQR1 algorithm starts with an initial scaling phase. Afterwards, the so-called scout phase and the subspace phase alternate until a stopping criterion is fulfilled. In the *scaling phase*, the algorithm searches for a well-scaled initial point with a finite function value. It is done by making a local line search from the initial point (an input parameter) in the direction of the point in the search region closest to the origin.

The *scout phase* consists of a sequence of local line searches in the direction of an orthonormal basis (to be efficient for non-separable smooth problems), occasionally preceded or followed by global line searches in all coordinate directions (to be efficient for approximately separable problems). At the beginning of each scout phase, the or-

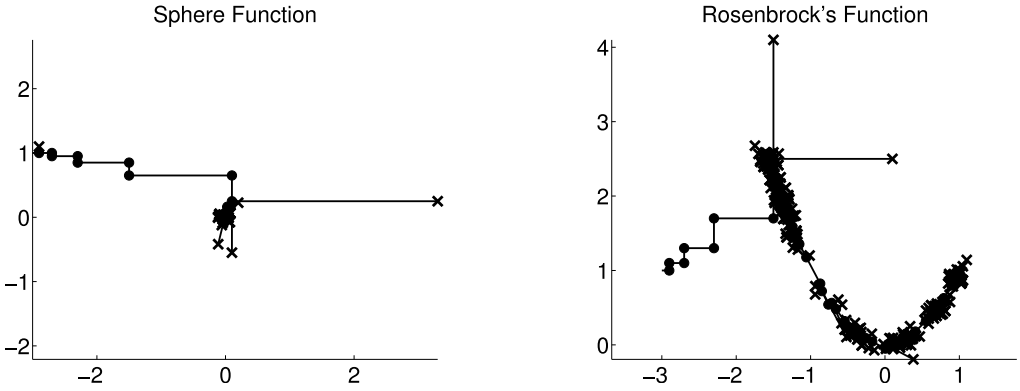

Figure 2: The behavior of Rosenbrock's optimization algorithm on the sphere (left) and on Rosenbrock's (right) function. The hollow circles indicate successful steps.

thonormal basis is created anew by a QR factorization with column pivoting of the matrix $(\mathbf{x}^1 - \mathbf{x}^{\text{best}}, \ldots, \mathbf{x}^D - \mathbf{x}^{\text{best}})$. The point $\mathbf{x}^{\text{best}}$ is the current best point. For the first scout phase, the points $\mathbf{x}^1, \ldots, \mathbf{x}^D$ are randomly generated in the search space, and for all the subsequent scout phases, they are the results of the line searches of the preceding scout phase.

In the *subspace phase*, an affine na-dimensional quadratic model is constructed, with $na \leq na_{\max}$. The saturation dimension $na_{\max}$ depends on the dimension D of the problem: $na_{\max} = 2$ for $D = 1$, and $na_{\max} = \max(3, \min((D+1)/2, 11))$ otherwise. The affine subspace initially consists of the initial point. The point $\mathbf{x}$ obtained after a scout phase is either added to the affine basis if the affine basis has less than $na_{\max}$ elements, or replaces the worst point otherwise. In the new subspace, a local quadratic model is created and minimized, subject to some safeguards. Then a local line search is performed from the current best point to the model minimizer. Since the local quadratic models are only generated in subspaces, these models can already be built when only a few evaluated points are available. The limit $na_{\max}$ makes the quadratic models tractable for high dimensions.

Emphasis is put on a fast descent with a low number of function evaluations, not necessarily on finding the global minimum. Since all line searches start from the current best point, the algorithm has a greedy tendency.

2.4 Reference Algorithms

We selected 3 other local search algorithms for the comparison. The first of them is the quasi-Newton method with the BFGS update formula. Newton methods search for a stationary point of a function (a point with a zero gradient). They assume that in the neighborhood of an optimum, the function can be approximated by a quadratic function. Newton methods use both first-order (gradient) and second-order (Hessian matrix) information about the function. Quasi-Newton methods do not need the precise Hessian matrix; instead, they are able to approximate it based on the individual successive gradients. The individual quasi-Newton methods differ in the way they perform the update of the Hessian matrix. BFGS is one of such methods. In this article, BFGS denotes the restarted quasi-Newton method with the BFGS update formula (as implemented in the MATLAB function `fminunc`). The details can be found in and the results are taken from Ros (2009a).

NEWUOA (Powell, 2006) was chosen as a competitor since it is a relatively recent optimization procedure with very promising reported results on various test functions. It is a deterministic (with the exception of initialization) local search procedure using quadratic modeling and a trust-region approach. The method maintains a quadratic model of the objective function in the trust region. Before each iteration, the model must interpolate the function at m points, with m typically equal to $2D + 1$, which is a much lower number of constraints than the number required to specify a full quadratic model. The remaining degrees of freedom are taken up by minimizing the Frobenius norm of the difference between the new and the old quadratic model. In this paper, a restarted version of NEWUOA is used as described by Ros (2009b).

Any comparison of local search methods in the real-valued space would not be complete without the Nelder-Mead Simplex Search (NMSS) method. The algorithm is rather old (Nelder and Mead, 1965), but is still used very often—it survived the test of time, despite it was shown by McKinnon (1999) that the algorithm can converge to a non-stationary point even on smooth functions. In the D-dimensional space, it maintains the so-called simplex, a set of $D + 1$ points. Their relative positions and function values determine where the next point(s) will be sampled. Since the simplex changes its shape, can become elongated or stretched, the algorithm is sometimes called 'amoeba'. In this article, NMSS denotes the restarted version of the method as described by Hansen (2009).

3 Experimental Framework Description

The experiments presented in this article were carried out using the Comparing Continuous Optimizers (COCO) framework (Hansen et al., 2009a), which was also used as the basis for the Black-box Optimization Benchmarking workshop at the GECCO-2009 and 2010 conferences.

The numerical experiments are performed on a testbed consisting of 24 noiseless test functions (Finck et al., 2009a; Hansen et al., 2009b). These functions have been constructed so that they reflect the real-world application difficulties and are categorized by function properties as multimodality, ill-conditioning, global structure and separability. The role of the categories is to reveal the different aspects of the algorithms. All functions are scalable with the dimension D. The search domain is $[-5; 5]^D$, where $D = 2, 3, 5, 10, 20, 40$. The functions have many instances differing in rotation and offset. Each algorithm is given 15 trials on each function.

An *optimization problem* is defined as a particular (function, requested target value) pair. Each function is used to define several optimization problems differing in the requested target value $f_\mathrm{t} = f_\mathrm{opt} + \Delta f_\mathrm{t}$, where f_opt is the optimal function value, and Δf_t is the precision (or tolerance) to reach. The success criterion of a trial (for each optimization problem) is to reach the requested target value f_t. Many precision levels $\Delta f_\mathrm{t} \in [10^{-8}, 10^{2}]$ are defined. If the optimizer finds a solution with the ultimate precision value 10^{-8}, it actually solves many optimization problems along the way, and we shall say that it has found the optimum of the function, i.e. that it solved the function. If the optimizer cannot reach the ultimate precision, it can gain some points for optimizing the function at least partially.

The main performance measure used in the COCO framework is the **Expected Running Time,** ERT (Hansen et al., 2009a; Price, 1997). The ERT estimates the expected number of function evaluations needed to reach the particular target function value if the algorithm is restarted until a single success. The ERT thus depends on the given target function value, f_t, and is computed as "the number of function evaluations

conducted in all trials, while the best function value was not smaller than f_t during the trial, divided by the number of trials that actually reached f_t" (Hansen et al., 2009a).

The results are conveniently presented using the **Empirical Cumulative Distribution Function (ECDF)**. It shows the empirical cumulated probability of success depending on the allocated budget. The ECDF of the ERT is constructed as a bootstrap distribution of the ERT divided by the problem dimension D. In the bootstrapping process, 100 instances of ERT are generated by repeatedly drawing single trials with replacement until a successful trial is drawn for each optimization problem.

Since the ECDF graphs do not express the reached function values, but rather the proportion of solved problems, it is possible to meaningfully aggregate the ECDF graphs for several functions of the same class into one graph. The downside of this aggregation is that we are not able to distinguish the individual functions. If a graph shows aggregated ECDFs of 5 functions for a certain dimension D, reaching the 20 % level of solved problems after n evaluations may mean many things. On the one hand, the algorithm could have found the minimum of one of the five functions, while the other functions may still remain completely unsolved. On the other hand, it may mean that only the problems related to the loose target levels were solved across all the aggregated functions. The latter case is the usual one. If the former explanation is the right one, we will point it out explicitly.

4 Algorithm and Experiment Parameter Settings

The following subsections describe the experimental setup and the parameter settings of both LS methods, RA, and VXQR1. For the settings of the reference algorithms, we refer the reader to the original reports (Ros, 2009a,b; Hansen, 2009). All the presented algorithms use the same parameter settings across all functions and dimensions.

The LS methods and Rosenbrock's algorithm were benchmarked using the BBOB-2009 settings, i.e. the algorithms were run on the 24 benchmark functions, 5 instances each, 3 trials per instance. The VXQR1 algorithm was benchmarked using the BBOB-2010 settings, i.e. the algorithm was run on 24 benchmark functions, 15 instances each, 1 trial per instance.

4.1 Line Search

The `fminbnd` function does not have any parameters (except the search space bounds and termination criteria, both are described later). The STEP algorithm has 2 parameters: the Jones factor $\epsilon = 10^{-8}$ and the maximum interval difficulty set to 10^7 (value determined by experimenting with the Rastrigin function).

All benchmark functions have their optimum in $\langle -5, 5\rangle^D$ (Hansen et al., 2009b), yet both algorithms were ordered to find the optimum in the hypercube $\langle -6, 6\rangle^D$. This decision was made due to the `fminbnd` function that almost never samples the boundaries of the interval, which would make it extremely difficult for the algorithm to find the solution e.g. for the linear slope function.

Each multistart trial was finished either (1) after finding a solution with a precision $\Delta f_{\text{best}} \leq 10^{-8}$, or (2) after performing more than $10^4 \times D$ function evaluations. Each individual trial of the basic line search algorithm was interrupted (1) if any of the above mentioned criteria were satisfied, or (2) if two consecutive cycles over all directions ended up with solutions with distance lower than 10^{-10}, in which case the algorithm was restarted from a new randomly generated point. Finally, the individual univariate searches were stopped

- in case of `fminbnd`, when the target function value was reached, or the maximum

allowed number of function evaluations for `fminbnd` was reached (100), or when the boundary points of the interval got closer than 10^{-10}, and

- in case of STEP, when the target function value was reached, or when the maximum allowed number of function evaluations was reached (1000); moreover, an interval was not used for further sampling if its boundary points were closer than 10^{-10}, or when the difficulty of the interval was higher than 10^7.

The difference in the maximal number of allowed evaluations (100 vs. 1,000) is due to the fact that `fminbnd` is a local line search technique, i.e. if the function is unimodal, it needs a relatively small number of evaluations to find the global optimum. A larger value would not help the algorithm on multimodal functions. On the other hand, STEP is a global line search technique, and it needs larger budget to converge to the global optimum on multimodal functions.

4.2 Rosenbrock's Algorithm

The algorithm has 2 parameters, α and β, set to their default values: $\alpha = 2$ and $\beta = \frac{1}{2}$. The algorithm was run in the unconstrained setting.

Each multistart trial was finished either (1) after finding a solution with a precision $\Delta f_{\text{best}} \leq 10^{-8}$, or (2) after performing more than $10^4 \times D$ function evaluations. Each individual run of the basic Rosenbrock algorithm was interrupted (1) after finding a solution with a precision $\Delta f_{\text{best}} \leq 10^{-8}$, or (2) after performing more than the allowed number of function evaluations, or (3) after the model converged too much, i.e. when $\max_i |d_i| < 10^{-9}$, in which case the algorithm was restarted.

4.3 VXQR1

For all control variables in the algorithm default values are implemented. The only parameters we used in our experiments were the ones determining the stopping criterion, namely the limit $nf_{\max}$ on the number of function calls and reaching the target value $f_{\text{t}}^* := f_{\text{opt}} + 10^{-8}$, where f_{opt} is the global minimum of the function. Each test function instance was solved with VXQR1 with a limit of $nf_{\max} = 500 \max(D, 10)$ function evaluations. At most 9 independent restarts of VXQR1 with the origin as the initial point were made if f_{t}^* was not reached. I.e., at most 10 independent attempts were made to solve each problem with VXQR1. The same function evaluation budget per call and 10 calls were also used in the experiments carried out by the coauthor with MCS (Huyer and Neumaier, 2009).

5 Results

Results from experiments according to Hansen et al. (2009a) on the benchmark functions (Finck et al., 2009b; Hansen et al., 2009b) are presented in Figures 3, 4, and 5. Only the results for $D = 5$ (exemplar of "low" dimensionality) and $D = 20$ (exemplar of "higher" dimensionality) are presented.

Tables 1 to 10 give the Expected Running Time (ERT) for the target precisions $10^{1,0,-1,-3,-5,-7}$ divided by the best ERT obtained during BBOB 2009 (given in the ERT_{best} row), together with a measure of its spread (the value typeset in parentheses with a smaller font gives the half of the range between the 10th and 90th percentile). Bold entries correspond to the 3 best values among the algorithms compared. The median number of conducted function evaluations is additionally given in *italics*, if $\text{ERT}(10^{-7}) = \infty$. #succ is the number of trials that reached the final target $f_{\text{opt}} + 10^{-8}$.

Each algorithm is tested if it improved the results obtained by a baseline algorithm.

We chose the NMSS as the baseline in the 5-D space, and the NEWUOA as the baseline in the 20-D space, since these two algorithms were the most successful regarding the final success rate. The comparison should thus reveal the situations in which other algorithms are more suitable than these good solvers. The **statistical significance** is tested with the rank-sum test for a given target f_t using, for each trial, either the number of needed function evaluations to reach f_t (inverted and multiplied by -1), or, if the target was not reached, the best Δf-value achieved, measured only up to the smallest number of overall function evaluations for any unsuccessful trial under consideration if available. Entries with the $\downarrow$ symbol are statistically significantly better (according to the rank-sum test) compared to the baseline algorithm, with $p = 0.05$ or $p = 10^{-k}$ where $k > 1$ is the number following the $\downarrow$ symbol, with the Bonferroni correction of 24.

6 Discussion by Function Group

In this section, the discussion of the results is broken down by the function groups. The discussion mostly applies to the presented results for 5-D and 20-D. For a discussion on the individual algorithms, see Sec. 7.

6.1 All Functions Aggregated

The results for all the functions are aggregated in the ECDF graphs of ERT for the 5-D and 20-D functions in Figs. 3 and 4, respectively, in the upper left part.

In the 5-D space, for low evaluation budgets $\#\mathrm{FEs} < 20D$, NEWUOA was the most successful method solving almost 20 % of the problems. For larger budgets, NMSS and VXQR1 solved the largest proportion of the problems among the compared methods, reaching about 80 %. NEWUOA and BFGS were similarly fast but solved only about 65 % of the problems. RA reached this level as well, but was about 10 times slower. Both line search methods were slower and less successful than the other algorithms.

For the 20-D problems, NEWUOA eventually solved almost 60 % of the problems. Its success rate was the highest for almost all evaluation budgets, with the exception of a short range $40D < \#\mathrm{FEs} < 100D$, where RA dominated. Until about $2000D$ evaluations, BFGS closely followed NEWUOA and eventually solved about 50 % of the problems. This level was reached also by VXQR1 and NMSS, but they were slower than NEWUOA and BFGS. RA and both LS algorithms were slow and solved eventually about 40 % of the problems.

6.2 Separable Functions f_1–f_5

The ECDF graphs of ERT for the 5-D and the 20-D separable functions f_1–f_5 are aggregated in Figs. 3 and 4, respectively, in the upper right part. Tables 1 and 2 contain the detailed results for the 5-D and the 20-D functions, respectively.

In the 5-D space, NEWUOA was the fastest to solve functions f_1 and f_5 (it solved 40 % of the problems in less than $4D$ evaluations). After about $20D$ evaluations, LSfminbnd solved also f_2 and reached a level over 60 %. Finally, after about $200D$ evaluations, LSstep and VXQR1 solved also f_3 and f_4 and thus solved 100 % of the problems. VXQR1 even improved the best results for these two functions observed during BBOB-2009 for the loose target levels, some of them significantly. NMSS eventually solved also f_3 and reached a level over 80 %. The other methods did not solve f_3 and f_4, and they thus solved between 60 and 70 % of the problems only.

For the 20-D space, the situation is almost the same. NEWUOA solved f_1 and f_5, dominating in the beginning. After about $20D$ evaluations, LSfminbnd and RA solved also f_2 and reached the level of 60 %. This success rate was eventually reached by all

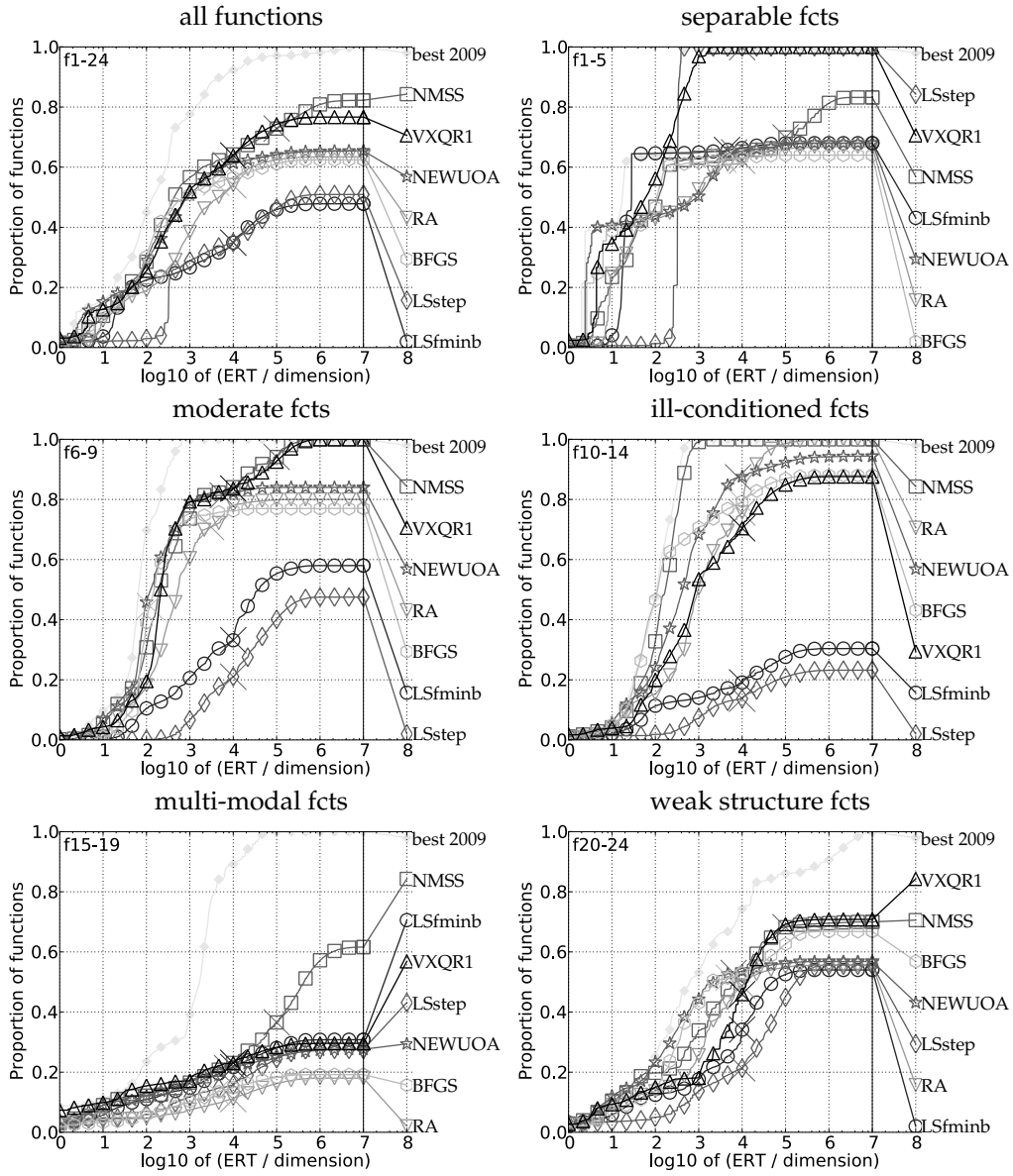

Figure 3: Empirical cumulative distribution of the bootstrapped distribution of ERT over dimension for 50 targets in $10^{[-8..2]}$ for all functions and subgroups in 5-D. The best ever line corresponds to the algorithms from BBOB-2009 with the best ERT for each of the targets considered

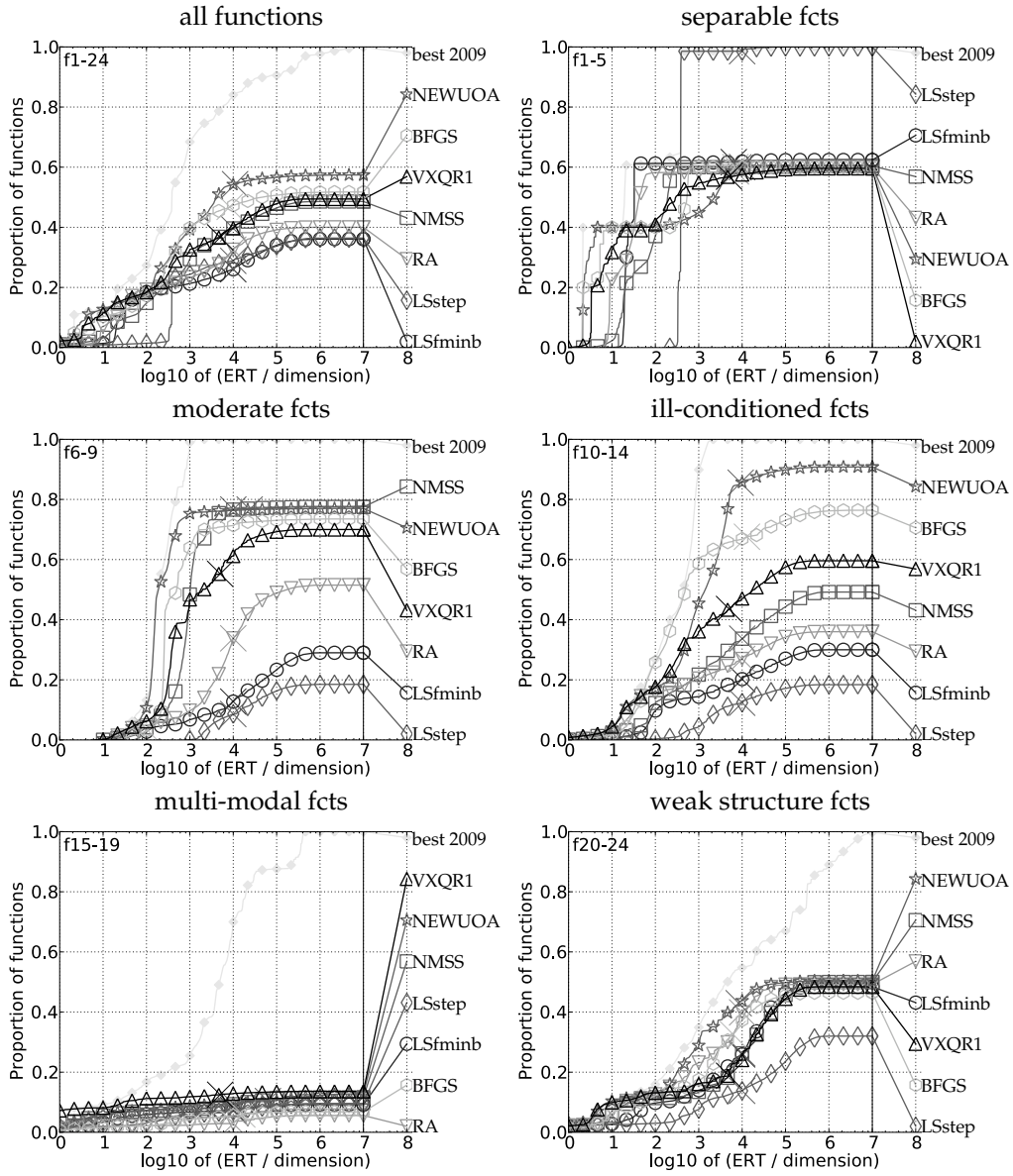

Figure 4: Empirical cumulative distribution of the bootstrapped distribution of ERT over dimension for 50 targets in $10^{[-8..2]}$ for all functions and subgroups in 20-D. The best ever line corresponds to the algorithms from BBOB-2009 with the best ERT for each of the targets considered

Table 1: ERT on f_1–f_5 in 5-D over $\mathrm{ERT}_{\mathrm{best}}$ obtained in BBOB-2009. The NMSS is used as the baseline for statistical comparisons.

1 Sphere

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	11	12	12	12	12	12	15/15
LSfminbnd	6.0(3.0)	6.3(2.6)	6.7(2.5)	6.8(2.6)$^{\downarrow 2}$	6.8(2.6)$^{\downarrow 4}$	6.8(2.6)$^{\downarrow 4}$	15/15
LSstep	92(55)	121(16)	129(0.16)	132(0.08)	132(0.04)	132(0.08)	15/15
NMSS	**1.5**(0.91)	3.3(1.6)	5.4(1.4)	9.2(1.7)	13(1.5)	17(1.6)	15/15
RA	**2.9**(0.86)	4.2(1.9)	5.5(1.0)	8.7(1.6)	12(1.5)	15(2.1)	15/15
NEWUOA	**1.1**(0.0)	**1**(0.0)$^{\downarrow 4}$	**1**(0.0)$^{\downarrow 4}$	**1**(0.0)$^{\downarrow 4}$	**1**(0.0)$^{\downarrow 4}$	**1**(0.0)$^{\downarrow 4}$	15/15
BFGS	**1.2**(0.0)	**1.1**(0.0)$^{\downarrow 4}$	**1.1**(0.0)$^{\downarrow 4}$	**1.1**(0.0)$^{\downarrow 4}$	**1.1**(0.0)$^{\downarrow 4}$	**1.1**(0.0)$^{\downarrow 4}$	15/15
VXQR1	**1.5**(0.55)	**1.7**(0.12)$^{\downarrow 2}$	**1.7**(0.12)$^{\downarrow 4}$	**1.8**(0.0)$^{\downarrow 4}$	**1.8**(0.0)$^{\downarrow 4}$	**1.8**(0.0)$^{\downarrow 4}$	15/15

2 Ellipsoid separable

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	83	87	88	90	92	94	15/15
LSfminbnd	**1**(0.16)$^{\downarrow 4}$	**1**(0.14)$^{\downarrow 4}$	**1**(0.14)$^{\downarrow 4}$	**1**(0.14)$^{\downarrow 4}$	**1**(0.13)$^{\downarrow 4}$	**1**(0.12)$^{\downarrow 4}$	15/15
LSstep	16(2.4)	16(1.9)	16(1.8)	15(1.8)	15(1.8)	15(1.7)	15/15
NMSS	5.0(3.3)	6.8(2.5)	7.4(2.1)	7.9(2.0)	**8.3**(1.9)	**8.6**(1.8)	15/15
RA	13(25)	102(178)	136(284)	153(287)	188(282)	241(288)	12/15
NEWUOA	5.7(4.0)	22(16)	45(30)	85(32)	129(33)	166(54)	15/15
BFGS	**3.8**(1.8)	**5.6**(2.4)	**6.2**(1.8)	**6.6**(1.6)	**6.9**(1.6)	**7.1**(1.7)	15/15
VXQR1	**2.8**(0.82)	**3.2**(1.1)$^{\downarrow 2}$	**4.1**(2.1)$^{\downarrow 2}$	**5.9**(1.7)	12(6.2)	22(11)	15/15

3 Rastrigin separable

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	716	1622	1637	1646	1650	1654	15/15
LSfminbnd	**1**(2.0)	**52**(64)	∞	∞	∞	∞ *2e4*	0/15
LSstep	**2.2**(0.28)	**1**(8.6e−3)$^{\downarrow 4}$	**1**(0.01)$^{\downarrow 4}$	**1**(0.01)$^{\downarrow 4}$	**1**(0.01)$^{\downarrow 4}$	**1**(0.01)$^{\downarrow 4}$	15/15
NMSS	5.4(6.5)	282(269)	**1464**(1537)	**1456**(1387)	**1452**(1515)	**1449**(1512)	3/15
RA	24(36)	394(452)	∞	∞	∞	∞ *5e4*	0/15
NEWUOA	6.1(8.3)	229(257)	∞	∞	∞	∞ *3e4*	0/15
BFGS	107(124)	∞	∞	∞	∞	∞ *2e4*	0/15
VXQR1	**0.32**(0.22)	**0.48**(0.35)$^{\downarrow 4}$	**0.94**(0.74)$^{\downarrow 4}$	**0.97**(0.75)$^{\downarrow 4}$	**1.0**(0.75)$^{\downarrow 4}$	**1.1**(0.78)$^{\downarrow 4}$	15/15

4 Skew Rastrigin-Bueche separ

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	809	1633	1688	1817	1886	1903	15/15
LSfminbnd	**7.8**(13)	∞	∞	∞	∞	∞ *2e4*	0/15
LSstep	**2.0**(9.3e−3)	**1**(8.9e−3)$^{\downarrow 4}$	**1**(0.03)$^{\downarrow 4}$	**1**(0.05)$^{\downarrow 4}$	**1**(0.05)$^{\downarrow 4}$	**1**(0.03)$^{\downarrow 4}$	12/15
NMSS	26(23)	∞	∞	∞	∞	∞ *5e5*	0/15
RA	57(62)	∞	∞	∞	∞	∞ *5e4*	0/15
NEWUOA	27(26)	**305**(339)	∞	∞	∞	∞ *3e4*	0/15
BFGS	169(192)	∞	∞	∞	∞	∞ *2e4*	0/15
VXQR1	**0.28**(0.10)	**0.96**(0.92)$^{\downarrow 4}$	**2.0**(2.0)$^{\downarrow 4}$	**1.9**(1.8)$^{\downarrow 4}$	**1.9**(1.8)$^{\downarrow 4}$	**2.2**(2.0)$^{\downarrow 4}$	15/15

5 Linear slope

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	10	10	10	10	10	10	15/15
LSfminbnd	13(1.7)	14(0.0)	14(0.0)	14(0.0)	14(0.0)	14(0.0)	15/15
LSstep	141(40)	160(0.05)	160(0.05)	160(0.05)	160(0.05)	160(0.05)	15/15
NMSS	**2.5**(1.1)	**4.1**(4.4)	**4.2**(4.6)	4.2(4.6)	4.2(4.6)	4.2(4.6)	15/15
RA	4.0(0.80)	4.2(0.55)	4.2(0.55)	**4.2**(0.55)	**4.2**(0.55)	**4.2**(0.55)	15/15
NEWUOA	**1.3**(0.15)$^{\downarrow 4}$	**1.5**(0.25)$^{\downarrow 3}$	**1.5**(0.20)$^{\downarrow 3}$	**1.5**(0.25)$^{\downarrow 3}$	**1.5**(0.25)$^{\downarrow 3}$	**1.5**(0.25)$^{\downarrow 3}$	15/15
BFGS	**1.9**(0.60)	**3.0**(0.90)	**3.1**(0.90)	**3.1**(0.90)	**3.1**(0.90)	**3.1**(0.90)	15/15
VXQR1	**3.0**(3.2)	4.4(4.3)	4.4(4.3)	4.6(5.8)	4.6(5.8)	4.6(5.8)	15/15

the algorithms with the exception of LSstep which solved all the functions after about $500D$ evaluations. Note that VXQR1 did not find the tight target levels of the separable ellipsoid function f_2. The separable multimodal functions f_3 and f_4 were solved only by the LSstep method, which is especially suitable for such problems. However, its applicability is limited since the real-world problems are usually non-separable.

Table 2: ERT on f_1–f_5 in 20-D over $\mathrm{ERT}_{\mathrm{best}}$ obtained in BBOB-2009. The NEWUOA is used as the baseline for statistical comparisons.

1 Sphere

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	43	43	43	43	43	43	15/15
LSfminbnd	9.3(1.8)	10(1.3)	10(1.2)	10(1.2)	10(1.2)	10(1.2)	15/15
LSstep	164(14)	175(4.7)	176(0.02)	177(0.03)	177(0.02)	177(0.03)	15/15
NMSS	5.2(2.0)	12(4.9)	19(7.7)	32(7.9)	40(6.5)	49(8.7)	15/15
RA	3.8(0.59)	5.8(0.55)	7.2(0.55)	11(0.84)	14(0.99)	17(1.5)	15/15
NEWUOA	**1.0**(0.02)	**1.0**(0.01)	**1.0**(0.01)	**1.0**(0.01)	**1.0**(0.01)	**1.0**(0.01)	15/15
BFGS	**1**(0.0)	**1**(0.0)	**1**(0.0)	**1**(0.0)	**1**(0.0)	**1**(0.0)	15/15
VXQR1	**1.4**(0.17)	**1.5**(0.07)	**1.5**(0.03)	**1.5**(0.03)	**1.6**(0.0)	**1.6**(0.0)	15/15

2 Ellipsoid separable

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	385	386	387	390	391	393	15/15
LSfminbnd	**1**(0.06)$^{\downarrow 4}$	**1**(0.05)$^{\downarrow 4}$	**1**(0.05)$^{\downarrow 4}$	**1**(0.06)$^{\downarrow 4}$	**1**(0.07)$^{\downarrow 4}$	**1**(0.06)$^{\downarrow 4}$	15/15
LSstep	17(0.51)	17(0.22)$^{\downarrow 4}$	17(0.22)$^{\downarrow 4}$	**17**(0.21)$^{\downarrow 4}$	**17**(0.22)$^{\downarrow 4}$	**17**(0.21)$^{\downarrow 4}$	15/15
NMSS	7.0(0.81)$^{\downarrow 3}$	7.8(1.2)$^{\downarrow 4}$	**8.6**(1.4)$^{\downarrow 4}$	**10**(1.2)$^{\downarrow 4}$	**11**(0.86)$^{\downarrow 4}$	**12**(1.3)$^{\downarrow 4}$	15/15
RA	**1.4**(0.26)$^{\downarrow 4}$	**1.6**(0.24)$^{\downarrow 4}$	**5.8**(0.16)$^{\downarrow 3}$	29(39)	73(238)	73(237)	14/15
NEWUOA	18(7.9)	42(21)	71(36)	125(43)	174(51)	219(67)	15/15
BFGS	20(3.5)	24(5.1)$^{\downarrow 2}$	26(4.4)$^{\downarrow 4}$	27(3.4)$^{\downarrow 4}$	28(3.2)$^{\downarrow 4}$	28(3.2)$^{\downarrow 4}$	15/15
VXQR1	**5.6**(0.36)$^{\downarrow 4}$	**7.5**(1.7)$^{\downarrow 4}$	12(3.2)$^{\downarrow 4}$	54(38)$^{\downarrow 2}$	∞	∞ *1e5*	0/15

3 Rastrigin separable

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	5066	7626	7635	7643	7646	7651	15/15
LSfminbnd	∞	∞	∞	∞	∞	∞ *1e5*	0/15
LSstep	**1.5**(0.04)$^{\downarrow 4}$	**1**(1.1e−3)$^{\downarrow 4}$	**1**(2.0e−3)$^{\downarrow 4}$	**1**(1.6e−3)$^{\downarrow 4}$	**1**(1.7e−3)$^{\downarrow 4}$	**1**(1.4e−3)$^{\downarrow 4}$	15/15
NMSS	∞	∞	∞	∞	∞	∞ *2e5*	0/15
RA	∞	∞	∞	∞	∞	∞ *1e5*	0/15
NEWUOA	∞	∞	∞	∞	∞	∞ *1e5*	0/15
BFGS	∞	∞	∞	∞	∞	∞ *1e5*	0/15
VXQR1	**2.0**(1.3)$^{\downarrow 4}$	∞	∞	∞	∞	∞ *1e5*	0/15

4 Skew Rastrigin-Bueche separ

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	4722	7628	7666	7700	7758	1.41e5	9/15
LSfminbnd	∞	∞	∞	∞	∞	∞ *1e5*	0/15
LSstep	**1.6**(9.5e−4)$^{\downarrow 4}$	**1**(1.2e−3)$^{\downarrow 4}$	**1**(5.3e−3)$^{\downarrow 4}$	**1**(8.1e−3)$^{\downarrow 4}$	**1**(0.02)$^{\downarrow 4}$	**1**(1.4)$^{\downarrow 4}$	9/15
NMSS	∞	∞	∞	∞	∞	∞ *2e5*	0/15
RA	∞	∞	∞	∞	∞	∞ *2e5*	0/15
NEWUOA	∞	∞	∞	∞	∞	∞ *2e5*	0/15
BFGS	∞	∞	∞	∞	∞	∞ *2e5*	0/15
VXQR1	**4.4**(3.6)$^{\downarrow 4}$	∞	∞	∞	∞	∞ *1e5*	0/15

5 Linear slope

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	41	41	41	41	41	41	15/15
LSfminbnd	16(0.42)	16(0.0)	16(0.0)	16(0.0)	16(0.0)	16(0.0)	15/15
LSstep	185(4.9)	187(0.01)	187(0.01)	187(0.01)	187(0.01)	187(0.01)	15/15
NMSS	7.4(1.5)	8.8(1.9)	9.2(2.2)	9.2(2.3)	9.2(2.3)	9.2(2.3)	15/15
RA	4.2(0.31)	**4.3**(0.17)	**4.3**(0.17)	**4.3**(0.17)	**4.3**(0.17)	**4.3**(0.17)	15/15
NEWUOA	**1.2**(0.10)	**1.5**(0.38)	**1.6**(0.48)	**1.6**(0.48)	**1.6**(0.48)	**1.6**(0.48)	15/15
BFGS	**2.4**(0.26)	**2.7**(0.52)	**2.8**(0.26)	**2.8**(0.26)	**2.8**(0.26)	**2.8**(0.26)	15/15
VXQR1	**3.9**(2.0)	4.8(2.3)	4.9(2.4)	5.0(2.5)	5.0(2.5)	5.0(2.5)	15/15

6.3 Unimodal Functions with Moderate Conditioning f_6–f_9

The ECDF graphs of ERT for the 5-D and the 20-D unimodal functions f_6–f_9 are aggregated in Figs. 3 and 4, respectively, in the middle left part. Tables 3 and 4 contain the detailed results for the 5-D and the 20-D functions, respectively.

As expected, both the LS algorithms were significantly slower and less successful compared to the other algorithms both in 5- and 20-D, and they will not be discussed

Table 3: ERT on f_6–f_9 in 5-D over $\mathrm{ERT}_{\mathrm{best}}$ obtained in BBOB-2009. The NMSS is used as the baseline for statistical comparisons.

6 Attractive sector

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	114	214	281	580	1038	1332	15/15
LSfminbnd	96(219)	115(141)	135(181)	110(133)	82(98)	65(77)	6/15
LSstep	408(494)	292(292)	296(311)	604(689)	∞	∞ *5e4*	0/15
NMSS	**1**(0.53)	**1.9**(1.7)	**2.8**(2.6)	**2.3**(1.2)	**2.0**(0.89)	**2.6**(1.1)	15/15
RA	**2.2**(1.1)	**2.8**(5.5)	**2.4**(4.2)	4.3(2.4)	**2.8**(1.5)	**2.4**(1.2)	15/15
NEWUOA	**1.7**(1.5)	**2.4**(1.3)	3.6(2.2)	3.3(1.7)	**2.7**(1.2)	**2.9**(1.1)	15/15
BFGS	3.0(1.8)	3.3(1.4)	3.4(1.1)	**2.5**(0.80)	**2.0**(0.82)	7.8(7.5)	15/15
VXQR1	3.1(2.2)	**3.0**(1.7)	**3.2**(1.7)	**2.6**(1.1)	**2.4**(0.52)	**2.9**(0.68)	15/15

7 Step-ellipsoid

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	24	324	1171	1572	1572	1597	15/15
LSfminbnd	49(40)	64(85)	100(109)	∞	∞	∞ *5e4*	0/15
LSstep	374(195)	697(772)	640(683)	∞	∞	∞ *5e4*	0/15
NMSS	**27**(46)	**33**(36)	**56**(57)	**307**(352)	**307**(338)	**302**(333)	9/15
RA	1200(1373)	669(726)	∞	∞	∞	∞ *2e4*	0/15
NEWUOA	**10**(15)	**13**(18)	**60**(56)	∞	∞	∞ *3e4*	0/15
BFGS	∞	∞	∞	∞	∞	∞ *600*	0/15
VXQR1	**19**(31)	**48**(57)	**53**(60)	**455**(509)	**455**(509)	**448**(501)	1/15

8 Rosenbrock original

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	73	273	336	391	410	422	15/15
LSfminbnd	10(17)	287(379)	452(520)	∞	∞	∞ *5e4*	0/15
LSstep	64(30)	95(105)	328(372)	1826(2046)	∞	∞ *5e4*	0/15
NMSS	**1.6**(1.1)	3.7(4.4)	3.3(3.6)	3.1(3.1)	3.1(2.9)	**3.2**(2.9)	15/15
RA	32(5.7)	23(62)	22(53)	25(46)	30(47)	36(57)	13/15
NEWUOA	**1**(0.95)	**1.1**(0.81)	**1.2**(0.49)	**1.2**(0.44)	**1.2**(0.41)	**1.2**(0.40)	15/15
BFGS	**2.1**(1.7)	**1.8**(0.87)	**1.6**(0.71)	**1.5**(0.61)	**1.5**(0.57)	**1.5**(0.55)	15/15
VXQR1	**2.6**(1.6)	**2.0**(0.95)	**2.4**(0.67)	**2.7**(0.50)	**2.8**(0.57)	3.5(1.3)	15/15

9 Rosenbrock rotated

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	35	127	214	300	335	369	15/15
LSfminbnd	13(14)	131(217)	183(151)	∞	∞	∞ *5e4*	0/15
LSstep	519(766)	5594(6102)	∞	∞	∞	∞ *5e4*	0/15
NMSS	**3.1**(3.9)	13(24)	8.2(14)	6.2(10)	5.8(9.1)	5.4(8.3)	15/15
RA	5.3(7.6)	10(14)	10(13)	14(10)	14(9.5)	14(8.6)	15/15
NEWUOA	**1.8**(0.74)	**3.6**(2.8)	**2.5**(1.6)	**1.9**(1.1)	**1.9**(1.0)	**1.7**(0.92)	15/15
BFGS	3.6(2.9)	**3.0**(2.1)	**2.0**(1.2)	**1.6**(0.86)	**1.5**(0.78)	**1.4**(0.71)	15/15
VXQR1	**1.2**(0.82)$^{\downarrow}$	**1.8**(1.1)	**2.5**(0.43)	**2.5**(0.44)	**2.5**(0.31)	**3.2**(1.2)	15/15

any more in the remainder of this subsection. In the 5-D space, until about $1000D$ evaluations, NEWUOA, NMSS, VXQR1, and BFGS behave similarly and solved about 80 % of the problems. RA reached this level as well, but was about 5 to 10 times slower. After $1000D$ evaluations, only NMSS and VXQR1 increased their success rate further, solving eventually also the problems related to f_7, step-ellipsoid, and reached the 100 % success rate. The f_7 function remained unsolved by NEWUOA, RA, and BFGS.

For the 20-D functions, NEWUOA rapidly increased its success rate from about 10 % at $100D$ evaluations to about 75 % at $1000D$. (For many target levels for functions f_6, f_8, and f_9, NEWUOA is actually the best algorithm that took part in BBOB-2009.) NEWUOA is followed by BFGS and NMSS, which were 2 to 10 times slower than NEWUOA and reached a similar success rate around 75 %. VXQR1 was slower than NEWUOA and BFGS, and also reached a lower success rate of about 70 %. RA showed only a slow progress and solved only about 50 % of the problems. None of the methods solved the f_7 function, step-ellipsoid.

Table 4: ERT on f_6–f_9 in 20-D over $\mathrm{ERT}_{\mathrm{best}}$ obtained in BBOB-2009. The NEWUOA is used as the baseline for statistical comparisons.

6 Attractive sector

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	1296	2343	3413	5220	6728	8409	15/15
LSfminbnd	158(178)	564(679)	825(938)	∞	∞	∞ *2e5*	0/15
LSstep	2294(2470)	∞	∞	∞	∞	∞ *2e5*	0/15
NMSS	**2.7**(1.2)	**2.4**(0.86)	**2.3**(0.91)	**2.5**(0.70)	**3.0**(0.54)	**5.4**(3.9)	14/15
RA	31(20)	56(30)	150(151)	572(613)	∞	∞ *2e5*	0/15
NEWUOA	**1**(0.33)	**1**(0.37)	**1**(0.49)	**1.1**(0.54)	**1.3**(0.83)	**1.3**(0.66)	15/15
BFGS	3.6(1.6)	3.5(1.3)	**3.4**(1.1)	**3.5**(0.88)	**3.6**(0.73)	**45**(21)	0/15
VXQR1	**2.5**(1.2)	**2.9**(2.2)	4.8(4.6)	25(28)	210(238)	∞ *1e5*	0/15

7 Step-ellipsoid

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	1351	4274	9503	16524	16524	16969	15/15
LSfminbnd	**1045**(1110)	∞	∞	∞	∞	∞ *2e5*	0/15
LSstep	2184(2369)	∞	∞	∞	∞	∞ *2e5*	0/15
NMSS	**2160**(2453)	∞	∞	∞	∞	∞ *2e5*	0/15
RA	∞	∞	∞	∞	∞	∞ *7e4*	0/15
NEWUOA	∞	∞	∞	∞	∞	∞ *5e5*	0/15
BFGS	∞	∞	∞	∞	∞	∞ *2100*	0/15
VXQR1	**1080**(1147)	∞	∞	∞	∞	∞ *1e5*	0/15

8 Rosenbrock original

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	2039	3871	4040	4219	4371	4484	15/15
LSfminbnd	9.2(16)	114(129)	707(792)	∞	∞	∞ *2e5*	0/15
LSstep	24(4.2)	124(130)	222(248)	∞	∞	∞ *2e5*	0/15
NMSS	3.3(1.5)	**3.8**(3.4)	**4.1**(3.1)	**4.9**(2.9)	**5.3**(2.8)	**5.6**(2.7)	15/15
RA	3.8(6.6)	24(31)	28(34)	42(47)	62(46)	666(714)	0/15
NEWUOA	**1**(0.26)	**1**(0.59)	**1**(0.56)	**1**(0.54)	**1**(0.52)	**1**(0.50)	15/15
BFGS	**1.8**(0.26)	**1.2**(0.15)	**1.2**(0.14)	**1.2**(0.13)	**1.2**(0.13)	**1.2**(0.13)	15/15
VXQR1	**2.5**(0.87)	13(13)	15(19)	26(35)	25(34)	25(33)	6/15

9 Rosenbrock rotated

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	1716	3102	3277	3455	3594	3727	15/15
LSfminbnd	52(29)	467(500)	∞	∞	∞	∞ *2e5*	0/15
LSstep	∞	∞	∞	∞	∞	∞ *2e5*	0/15
NMSS	3.6(1.5)	6.6(5.8)	7.2(5.6)	8.4(4.9)	8.8(4.7)	8.9(4.5)	15/15
RA	8.4(1.7)	31(33)	37(31)	63(31)	∞	∞ *2e5*	0/15
NEWUOA	**1.0**(0.16)	**1**(0.60)	**1**(0.57)	**1**(0.54)	**1**(0.52)	**1**(0.50)	15/15
BFGS	**2.2**(0.37)	**2.2**(0.97)	**2.1**(0.93)	**2.0**(0.88)	**2.0**(0.84)	**1.9**(0.81)	15/15
VXQR1	**0.39**$(0.20)^{\downarrow 4}$	**1.5**(0.77)	**1.9**(0.75)	**2.4**(1.6)	**2.7**(1.6)	**4.1**(4.5)	15/15

6.4 Unimodal Ill-conditioned Functions f_{10}–f_{14}

The ECDF graphs of ERT for the 5-D and the 20-D unimodal ill-conditioned functions f_{10}–f_{14} are aggregated in Figs. 3 and 4, respectively, in the middle right part. Tables 5 and 6 contain the detailed results for the 5-D and the 20-D functions, respectively.

Similarly to the functions with moderate conditioning, both the LS methods exhibit a very bad performance (as expected) and thus will not be discussed in the remainder of this subsection. For the 5-D space, the BFGS method solves about 60 % of the problems in $100D$ evaluations, closely followed by the NMSS. However, when solving the tightest target levels, BFGS lost its performance (possibly because of inappropriate restart conditions). From about $200D$ evaluations, NMSS dominates all the other methods solving all the functions in about $1000D$ evaluations. Interestingly, RA eventually also reaches the success rate of 100 %, while the methods based on quadratic modeling (NEWUOA, BFGS, and VXQR1) solved only about 90 % of the problems (they failed to find the tightest target levels of f_{13}, sharp ridge).

Table 5: ERT on f_{10}–f_{14} in 5-D over $\mathrm{ERT}_{\mathrm{best}}$ obtained in BBOB-2009. The NMSS is used as the baseline for statistical comparisons.

10 Ellipsoid

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	349	500	574	626	829	880	15/15
LSfminbnd	∞	∞	∞	∞	∞	∞ *5e4*	0/15
LSstep	∞	∞	∞	∞	∞	∞ *5e4*	0/15
NMSS	**1.4**(0.90)	**1.3**(0.65)	**1.4**(0.67)	**1.5**(0.72)	**1.2**(0.68)	**1.2**(0.65)	15/15
RA	24(43)	44(55)	40(47)	37(44)	29(34)	37(57)	10/15
NEWUOA	**3.1**(3.3)	5.5(4.4)	**8.1**(6.7)	**14**(8.5)	**16**(7.5)	**21**(7.5)	15/15
BFGS	**1**(0.52)	**1**(0.20)	**1**(0.28)	**1**(0.28)	**1.1**(0.42)	**23**(30)	5/15
VXQR1	4.1(3.1)	**5.1**(2.9)	10(15)	23(21)	110(99)	∞ *5e4*	0/15

11 Discus

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	143	202	763	1177	1467	1673	15/15
LSfminbnd	∞	∞	∞	∞	∞	∞ *5e4*	0/15
LSstep	4909(5698)	∞	∞	∞	∞	∞ *5e4*	0/15
NMSS	**3.2**(2.6)	**5.0**(2.7)	**1.7**(0.53)	**1.5**(0.67)	**1.5**(0.70)	**1.6**(0.71)	15/15
RA	117(176)	88(126)	26(34)	18(22)	14(18)	**13**(16)	12/15
NEWUOA	**3.5**(1.8)	**4.7**(2.2)	**1.8**(0.63)	**1.8**(0.44)	**2.0**(0.35)	**2.2**(0.41)	15/15
BFGS	**1**(0.21)$^{\downarrow 4}$	**1**(0.52)$^{\downarrow 3}$	**1.1**(1.2)	8.2(11)	199(212)	∞ *4e4*	0/15
VXQR1	5.1(5.7)	7.9(4.0)	3.0(1.0)	**3.2**(1.2)	**6.9**(5.5)	41(45)	1/15

12 Bent cigar

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	108	268	371	461	1303	1494	15/15
LSfminbnd	310(462)	1215(1400)	1890(2159)	∞	∞	∞ *5e4*	0/15
LSstep	463(693)	1237(1400)	∞	∞	∞	∞ *5e4*	0/15
NMSS	**2.3**(1.4)	**2.2**(1.7)	**2.2**(1.5)	**2.3**(1.5)	**1**(0.56)	**1**(0.62)	15/15
RA	98(231)	63(98)	91(135)	95(115)	42(57)	**48**(55)	6/15
NEWUOA	**3.5**(3.3)	**2.6**(2.6)	**2.5**(2.3)	**2.6**(2.3)	**1.1**(1.0)	**1.1**(0.99)	15/15
BFGS	**1.1**(0.89)	**1**(0.62)	**1**(0.63)	**1**(0.56)	**2.0**(2.6)	49(68)	5/15
VXQR1	3.6(2.6)	3.7(4.4)	4.2(4.8)	13(14)	41(52)	56(68)	5/15

13 Sharp ridge

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	132	195	250	1310	1752	2255	15/15
LSfminbnd	33(43)	149(146)	544(510)	∞	∞	∞ *5e4*	0/15
LSstep	546(607)	1122(1285)	2873(3203)	∞	∞	∞ *5e4*	0/15
NMSS	**2.0**(2.7)	**3.8**(3.8)	**5.3**(3.2)	**1.3**(0.56)	**1.2**(0.74)	**1.3**(0.95)	15/15
RA	7.6(7.3)	13(10)	**26**(20)	**39**(49)	**63**(65)	**290**(315)	1/15
NEWUOA	**3.1**(3.3)	**9.3**(12)	35(36)	54(54)	335(360)	∞ *4e4*	0/15
BFGS	**1**(0.20)	**1**(0.11)	**1**(0.06)	**4.8**(8.6)	**136**(143)	∞ *5e4*	0/15
VXQR1	50(57)	156(180)	275(300)	538(592)	∞	∞ *5e4*	0/15

14 Sum of different powers

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	10	41	58	139	251	476	15/15
LSfminbnd	5.9(5.2)	4.5(2.8)	4.3(2.5)	70(59)	∞	∞ *5e4*	0/15
LSstep	117(102)	96(59)	97(41)	∞	∞	∞ *5e4*	0/15
NMSS	**1.1**(1.1)	**1.2**(0.59)	**1.5**(0.53)	**1.4**(0.25)	**1.3**(0.25)	**1**(0.13)	15/15
RA	**2.4**(2.0)	**1.2**(0.50)	**1.3**(0.40)	4.6(3.0)	26(20)	**43**(59)	10/15
NEWUOA	**1.7**(0.56)	**1**(0.35)	**1**(0.27)$^{\downarrow 2}$	**1.2**(0.31)	**5.5**(1.9)	2525(2896)	0/15
BFGS	**2.2**(1.8)	**1.7**(1.4)	**1.8**(1.1)	**1.3**(0.61)	**1**(0.37)	350(374)	0/15
VXQR1	**0.88**(0.82)	**2.9**(1.9)	**3.0**(0.84)	3.0(0.88)	6.1(3.2)	**148**(138)	2/15

In the 20-D space, both the pattern search methods, NMSS and RA, lost their performance. RA was only slightly better than the LS methods reaching a success rate about 35 %, while NMSS eventually reached about 50 %. As in 5-D case, BFGS is the fastest to reach a success rate of about 60 % in $1000D$ evaluations. NEWUOA is typically only 2 to 5 times slower, and from $2000D$ evaluations it dominates all the other algorithms, eventually solving 90 % of the problems. VXQR1 is dominated by BFGS from $50D$ evaluations, and from $500D$ evaluations also by NEWUOA. Again, the f_{13}

Table 6: ERT on f_{10}–f_{14} in 20-D over $\mathrm{ERT}_{\mathrm{best}}$ obtained in BBOB-2009. The NEWUOA is used as the baseline for statistical comparisons.

10 Ellipsoid

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	7413	8661	10735	14920	17073	17476	15/15
LSfminbnd	∞	∞	∞	∞	∞	∞ *2e5*	0/15
LSstep	∞	∞	∞	∞	∞	∞ *2e5*	0/15
NMSS	**390**(451)	∞	∞	∞	∞	∞ *2e5*	0/15
RA	∞	∞	∞	∞	∞	∞ *2e5*	0/15
NEWUOA	**1.7**(0.50)	**2.6**(0.78)	**3.3**(1.1)	**4.0**(0.83)	**4.7**(0.76)	**5.8**(1.0)	15/15
BFGS	**1.0**$(0.21)^{\downarrow 3}$	**1**$(0.14)^{\downarrow 4}$	**1**$(0.46)^{\downarrow 4}$	**1.1**$(0.40)^{\downarrow 4}$	**3.1**(4.3)	∞ *1e6*	0/15
VXQR1	∞	∞	∞	∞	∞	∞ *1e5*	0/15

11 Discus

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	1002	2228	6278	9762	12285	14831	15/15
LSfminbnd	∞	∞	∞	∞	∞	∞ *2e5*	0/15
LSstep	∞	∞	∞	∞	∞	∞ *2e5*	0/15
NMSS	41(48)	292(329)	∞	∞	∞	∞ *2e5*	0/15
RA	∞	∞	∞	∞	∞	∞ *2e5*	0/15
NEWUOA	**15**(2.5)	**13**(2.0)	**5.8**(0.55)	**6.1**(0.47)	**6.6**(0.32)	**6.5**(0.29)	15/15
BFGS	**1**$(0.50)^{\downarrow 4}$	**1**$(0.85)^{\downarrow 4}$	**1.3**(0.65)	**147**(157)	∞	∞ *2e5*	0/15
VXQR1	**3.7**$(2.0)^{\downarrow 4}$	**2.6**$(0.60)^{\downarrow 4}$	**1.2**$(0.23)^{\downarrow 4}$	**5.1**(5.6)	**121**(126)	∞ *1e5*	0/15

12 Bent cigar

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	1042	1938	2740	4140	12407	13827	15/15
LSfminbnd	97(192)	414(516)	∞	∞	∞	∞ *2e5*	0/15
LSstep	228(289)	676(774)	∞	∞	∞	∞ *2e5*	0/15
NMSS	19(27)	26(25)	57(62)	338(366)	∞	∞ *2e5*	0/15
RA	14(0.08)	56(103)	208(256)	∞	∞	∞ *2e5*	0/15
NEWUOA	**3.0**(2.9)	**3.0**(2.4)	**3.0**(1.7)	**2.5**(1.2)	**1**(0.42)	**1**(0.37)	15/15
BFGS	**1.6**(0.92)	**1.6**(1.5)	**1.6**(1.1)	**1.6**(1.1)	**1.8**(1.9)	**45**(57)	1/15
VXQR1	**1.3**(1.8)	**2.1**(2.4)	**4.0**(5.1)	**20**(24)	**117**(125)	∞ *1e5*	0/15

13 Sharp ridge

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	652	2021	2751	18749	24455	30201	15/15
LSfminbnd	19(35)	19(25)	68(73)	156(176)	∞	∞ *2e5*	0/15
LSstep	464(484)	1435(1583)	∞	∞	∞	∞ *2e5*	0/15
NMSS	11(14)	29(33)	60(65)	77(83)	∞	∞ *2e5*	0/15
RA	**2.5**(3.2)	**4.2**(5.1)	**8.3**(7.7)	**17**(19)	**122**(131)	∞ *2e5*	0/15
NEWUOA	**1**(1.4)	**3.0**(4.9)	**9.3**(12)	**19**(20)	∞	∞ *2e5*	0/15
BFGS	**1.7**(0.23)	**1**(0.04)	**1**(0.02)	**23**(28)	∞	∞ *5e5*	0/15
VXQR1	10(16)	28(32)	40(51)	76(80)	∞	∞ *1e5*	0/15

14 Sum of different powers

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	75	239	304	932	1648	15661	15/15
LSfminbnd	8.4(2.9)	5.2(0.94)	5.6(1.1)	57(32)	∞	∞ *2e5*	0/15
LSstep	186(91)	115(34)	118(28)	∞	∞	∞ *2e5*	0/15
NMSS	**2.3**(1.1)	**3.0**(1.8)	3.9(1.2)	**2.9**(0.58)	36(31)	∞ *2e5*	0/15
RA	**2.4**(0.54)	**1.2**(0.17)	**1.3**(0.18)	7.4(2.8)	∞	∞ *2e5*	0/15
NEWUOA	**1.5**(0.75)	**1**(0.32)	**1**(0.28)	**1**(0.18)	**9.1**(0.95)	**43**(32)	0/15
BFGS	**2.7**(0.98)	**1.8**(0.66)	**2.0**(0.69)	**1.2**(0.25)	**1.1**$(0.25)^{\downarrow 4}$	∞ *2e5*	0/15
VXQR1	**1.0**(0.44)	**0.76**(0.15)	**0.88**(0.13)	**1.6**(0.39)	**4.2**(1.2)	∞ *1e5*	0/15

function, sharp ridge, was the hardest and no algorithm solved it to the tightest target levels.

6.5 Multimodal Functions f_{15}–f_{19}

The ECDF graphs of ERT for the 5-D and the 20-D multimodal functions f_{15}–f_{19} are aggregated in Figs. 3 and 4, respectively, in the bottom left part. Tables 7 and 8 contain

Table 7: ERT on f_{15}–f_{19} in 5-D over $\mathrm{ERT}_{\mathrm{best}}$ obtained in BBOB-2009. The NMSS is used as the baseline for statistical comparisons.

15 Rastrigin

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	511	9310	19369	20073	20769	21359	14/15
LSfminbnd	**35**(42)	∞	∞	∞	∞	∞ *5e4*	0/15
LSstep	1381(1517)	**80**(89)	∞	∞	∞	∞ *5e4*	0/15
NMSS	**20**(24)	**43**(54)	**83**(90)	**80**(91)	**77**(84)	**75**(86)	4/15
RA	311(350)	∞	∞	∞	∞	∞ *5e4*	0/15
NEWUOA	**5.8**(5.7)	**41**(42)	∞	∞	∞	∞ *3e4*	0/15
BFGS	87(85)	∞	∞	∞	∞	∞ *2e4*	0/15
VXQR1	46(68)	∞	∞	∞	∞	∞ *5e4*	0/15

16 Weierstrass

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	120	612	2663	10449	11644	12095	15/15
LSfminbnd	**3.2**(4.8)	**28**(34)	**133**(150)	∞	∞	∞ *5e4*	0/15
LSstep	14(22)	276(280)	∞	∞	∞	∞ *5e4*	0/15
NMSS	**4.4**(9.2)	**28**(45)	**23**(22)	**95**(98)	**302**(322)	**597**(662)	1/15
RA	40(59)	1191(1307)	∞	∞	∞	∞ *5e4*	0/15
NEWUOA	**2.1**(1.8)	**29**(23)	∞	∞	∞	∞ *4e4*	0/15
BFGS	153(140)	960(1082)	∞	∞	∞	∞ *4e4*	0/15
VXQR1	8.2(21)	71(90)	**125**(141)	∞	∞	∞ *5e4*	0/15

17 Schaffer F7, condition 10

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	5.2	215	899	3669	6351	7934	15/15
LSfminbnd	238(693)	**64**(118)	**175**(195)	∞	∞	∞ *5e4*	0/15
LSstep	151(192)	676(816)	∞	∞	∞	∞ *5e4*	0/15
NMSS	**55**(190)	170(159)	**295**(288)	∞	∞	∞ *5e5*	0/15
RA	2695(4810)	∞	∞	∞	∞	∞ *5e4*	0/15
NEWUOA	**2.3**(1.5)	**40**(47)	617(670)	∞	∞	∞ *3e4*	0/15
BFGS	120(201)	645(729)	∞	∞	∞	∞ *2e4*	0/15
VXQR1	**11**(12)	**113**(135)	**388**(413)	∞	∞	∞ *5e4*	0/15

18 Schaffer F7, condition 1000

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	103	378	3968	9280	10905	12469	15/15
LSfminbnd	64(168)	**71**(76)	∞	∞	∞	∞ *5e4*	0/15
LSstep	148(246)	396(464)	**86**(94)	∞	∞	∞ *5e4*	0/15
NMSS	**45**(43)	**228**(303)	**321**(328)	∞	∞	∞ *5e5*	0/15
RA	3376(3849)	∞	∞	∞	∞	∞ *5e4*	0/15
NEWUOA	**31**(28)	1351(1734)	∞	∞	∞	∞ *9e4*	0/15
BFGS	57(59)	∞	∞	∞	∞	∞ *2e4*	0/15
VXQR1	**43**(48)	**94**(104)	∞	∞	∞	∞ *5e4*	0/15

19 Griewank-Rosenbrock F8F2

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	1	1	242	1.20e5	1.21e5	1.22e5	15/15
LSfminbnd	54(57)	**2950**(3564)	∞	∞	∞	∞ *5e4*	0/15
LSstep	914(600)	9487(9241)	1463(1548)	∞	∞	∞ *5e4*	0/15
NMSS	**12**(5.5)	**2885**(5139)	**590**(575)	∞	∞	∞ *5e5*	0/15
RA	1.3e4(2.5e4)	7.1e5(8.0e5)	∞	∞	∞	∞ *5e4*	0/15
NEWUOA	**14**(2.5)	2.7e4(2.4e4)	**1415**(1927)	∞	∞	∞ *5e5*	0/15
BFGS	1655(1611)	2.2e4(2.2e4)	1780(1929)	∞	∞	∞ *3e4*	0/15
VXQR1	**1.9**$(0.0)^{\downarrow 4}$	**2.0**$(0.0)^{\downarrow 4}$	**0.45**$(0.16)^{\downarrow 4}$	∞	∞	∞ *5e4*	0/15

the detailed results for the 5-D and the 20-D functions, respectively.

It is not surprising that the (restarted) local optimizers compared in this article do not work properly on the multimodal functions. In the 5-D space, their progress is very slow (the best of them needed about $4000D$ evaluations to solve 20 % of the problems). BFGS and RA are the worst for this function group reaching a success rate about 20 % only. The other methods solved about 30 % of the problems eventually, with the exception of NMSS. It managed to solve the f_{15}, Rastrigin, and f_{16}, Weierstrass

Table 8: ERT on f_{15}–f_{19} in 20-D over $\mathrm{ERT}_{\mathrm{best}}$ obtained in BBOB-2009. The NEWUOA is used as the baseline for statistical comparisons.

15 Rastrigin

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	30378	1.47e5	3.12e5	3.20e5	4.49e5	4.59e5	15/15
LSfminbnd	∞	∞	∞	∞	∞	∞ *2e5*	0/15
LSstep	∞	∞	∞	∞	∞	∞ *2e5*	0/15
NMSS	∞	∞	∞	∞	∞	∞ *2e5*	0/15
RA	∞	∞	∞	∞	∞	∞ *2e5*	0/15
NEWUOA	∞	∞	∞	∞	∞	∞ *1e5*	0/15
BFGS	∞	∞	∞	∞	∞	∞ *1e5*	0/15
VXQR1	∞	∞	∞	∞	∞	∞ *1e5*	0/15

16 Weierstrass

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	1384	27265	77015	1.88e5	1.98e5	2.20e5	15/15
LSfminbnd	**160**$_{(205)}$	∞	∞	∞	∞	∞ *2e5*	0/15
LSstep	239$_{(244)}$	∞	∞	∞	∞	∞ *2e5*	0/15
NMSS	**17**$_{(21)}$	∞	∞	∞	∞	∞ *2e5*	0/15
RA	∞	∞	∞	∞	∞	∞ *2e5*	0/15
NEWUOA	**16**$_{(17)}$	∞	∞	∞	∞	∞ *2e5*	0/15
BFGS	∞	∞	∞	∞	∞	∞ *3e5*	0/15
VXQR1	198$_{(193)}$	∞	∞	∞	∞	∞ *1e5*	0/15

17 Schaffer F7, condition 10

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	63	1030	4005	30677	56288	80472	15/15
LSfminbnd	992$_{(1593)}$	∞	∞	∞	∞	∞ *2e5*	0/15
LSstep	1698$_{(3175)}$	∞	∞	∞	∞	∞ *2e5*	0/15
NMSS	**237**$_{(585)}$	∞	∞	∞	∞	∞ *2e5*	0/15
RA	2.1e4$_{(2.5e4)}$	∞	∞	∞	∞	∞ *2e5*	0/15
NEWUOA	**16**$_{(4.0)}$	∞	∞	∞	∞	∞ *2e6*	0/15
BFGS	359$_{(613)}$	∞	∞	∞	∞	∞ *4e5*	0/15
VXQR1	**285**$_{(794)}$	∞	∞	∞	∞	∞ *1e5*	0/15

18 Schaffer F7, condition 1000

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	621	3972	19561	67569	1.31e5	1.47e5	15/15
LSfminbnd	**4518**$_{(5156)}$	∞	∞	∞	∞	∞ *2e5*	0/15
LSstep	∞	∞	∞	∞	∞	∞ *2e5*	0/15
NMSS	∞	∞	∞	∞	∞	∞ *2e5*	0/15
RA	∞	∞	∞	∞	∞	∞ *2e5*	0/15
NEWUOA	**1.2e4**$_{(1.3e4)}$	∞	∞	∞	∞	∞ *2e6*	0/15
BFGS	∞	∞	∞	∞	∞	∞ *4e5*	0/15
VXQR1	∞	∞	∞	∞	∞	∞ *1e5*	0/15

19 Griewank-Rosenbrock F8F2

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	1	1	3.43e5	6.22e6	6.69e6	6.74e6	15/15
LSfminbnd	1173$_{(1068)}$	∞	∞	∞	∞	∞ *2e5*	0/15
LSstep	7789$_{(2841)}$	∞	∞	∞	∞	∞ *2e5*	0/15
NMSS	**165**$_{(149)}$	**1.4e6**$_{(1.6e6)}$	∞	∞	∞	∞ *2e5*	0/15
RA	∞	∞	∞	∞	∞	∞ *2e5*	0/15
NEWUOA	**76**$_{(51)}$	**4.3e6**$_{(5.2e6)}$	∞	∞	∞	∞ *2e6*	0/15
BFGS	1.2e6$_{(1.2e6)}$	∞	∞	∞	∞	∞ *2e5*	0/15
VXQR1	**2.0**$_{(0.0)}^{\downarrow 4}$	**2.0**$_{(0.0)}^{\downarrow 4}$	**1.2e-3**$_{(1.8e-4)}$	∞	∞	∞ *1e5*	0/15

functions, while the other algorithms found the loose target precisions $\Delta f_{\mathrm{t}} \geq 10^{-1}$ only.

In the 20-D space, none of the algorithms was able to make any substantial progress. The best of them is VXQR1, which seems to be able to cope at least a bit with f_{19}, Griewank-Rosenbrock function. The final success rate of all the methods is well below 15 %. Despite the fact that it should be possible to reveal a global structure in these functions, none of the algorithms was successful in this respect.

Table 9: ERT on f_{20}–f_{24} in 5-D over $\mathrm{ERT}_{\mathrm{best}}$ obtained in BBOB-2009. The NMSS is used as the baseline for statistical comparisons.

20 Schwefel x*sin(x)

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	16	851	38111	54470	54861	55313	14/15
LSfminbnd	8.2(7.1)	18(22)	∞	∞	∞	∞ *5e4*	0/15
LSstep	232(125)	41(50)	**18**(20)	**13**(15)	**13**(14)	**13**(14)	1/15
NMSS	**1.5**(1.1)	25(28)	∞	∞	∞	∞ *5e5*	0/15
RA	**2.9**(0.75)	4.6(5.9)$^{\downarrow}$	∞	∞	∞	∞ *5e4*	0/15
NEWUOA	**1**(0.25)	**3.3**(3.6)$^{\downarrow 2}$	∞	∞	∞	∞ *3e4*	0/15
BFGS	**1.8**(0.75)	**2.5**(2.2)$^{\downarrow 3}$	**10**(11)$^{\downarrow 2}$	**7.2**(8.3)$^{\downarrow 2}$	**7.1**(7.8)$^{\downarrow 2}$	**7.1**(7.9)$^{\downarrow 2}$	1/15
VXQR1	**1.2**(0.28)	**4.1**(8.9)$^{\downarrow}$	**4.4**(4.6)$^{\downarrow 3}$	**3.1**(3.3)$^{\downarrow 3}$	**3.0**(3.2)$^{\downarrow 3}$	**3.0**(3.2)$^{\downarrow 3}$	4/15

21 Gallagher 101 peaks

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	41	1157	1674	1705	1729	1757	14/15
LSfminbnd	30(39)	38(39)	39(42)	44(42)	44(41)	44(43)	8/15
LSstep	564(649)	122(151)	123(149)	125(147)	130(145)	134(140)	2/15
NMSS	12(27)	8.4(8.2)	**10**(18)	**10**(17)	**10**(17)	**10**(17)	15/15
RA	10(25)	**7.9**(7.9)	15(20)	15(19)	15(19)	15(16)	12/15
NEWUOA	**1.1**(0.62)	**2.2**(2.5)	**1.8**(2.1)	**1.8**(2.1)	**1.8**(2.1)	**1.9**(2.0)	15/15
BFGS	**3.8**(5.1)	**1.4**(1.6)	**1.9**(2.9)	**1.9**(2.8)	**1.9**(2.8)	**2.0**(2.7)	15/15
VXQR1	**3.7**(4.4)	29(32)	32(34)	32(34)	31(35)	31(34)	9/15

22 Gallagher 21 peaks

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	71	386	938	1008	1040	1068	14/15
LSfminbnd	**13**(27)	47(71)	29(40)	62(68)	122(124)	220(234)	2/15
LSstep	191(355)	177(220)	380(418)	∞	∞	∞ *5e4*	0/15
NMSS	19(33)	**13**(25)	13(10)	13(10)	12(9.3)	12(9.1)	15/15
RA	19(14)	13(10)	**10**(11)	**10**(10)	**10**(10)	**11**(10)	15/15
NEWUOA	**2.1**(3.0)	**2.1**(2.2)	**2.0**(3.1)	**2.1**(2.8)	**2.3**(2.8)	**2.4**(2.7)	15/15
BFGS	**3.1**(3.8)	**2.9**(2.8)	**2.1**(1.6)	**2.0**(1.5)	**2.0**(1.4)	**2.6**(2.2)	14/15
VXQR1	72(107)	53(78)	29(35)	27(32)	27(31)	26(30)	12/15

23 Katsuuras

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	3.0	518	14249	31654	33030	34256	15/15
LSfminbnd	**1.8**(1.7)	11(12)	∞	∞	∞	∞ *5e4*	0/15
LSstep	**1.4**(1.3)	6.6(8.3)	51(54)	∞	∞	∞ *5e4*	0/15
NMSS	**2.9**(3.3)	**3.5**(5.9)	**2.7**(3.4)	**4.0**(3.8)	**4.6**(4.4)	**5.6**(4.8)	14/15
RA	**1.6**(1.8)	**1.8**(1.7)	**4.6**(4.7)	∞	∞	∞ *2e4*	0/15
NEWUOA	6.2(4.2)	**2.4**(2.5)	**7.1**(8.2)	∞	∞	∞ *3e4*	0/15
BFGS	11(18)	31(34)	∞	∞	∞	∞ *2e4*	0/15
VXQR1	**2.4**(2.0)	9.3(10)	12(12)	∞	∞	∞ *5e4*	0/15

24 Lunacek bi-Rastrigin

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	1622	2.16e5	6.36e6	9.62e6	1.28e7	1.28e7	3/15
LSfminbnd	**9.1**(9.3)	∞	∞	∞	∞	∞ *5e4*	0/15
LSstep	203(231)	∞	∞	∞	∞	∞ *5e4*	0/15
NMSS	**11**(11)	**5.6**(6.1)	∞	∞	∞	∞ *5e5*	0/15
RA	212(241)	∞	∞	∞	∞	∞ *5e4*	0/15
NEWUOA	**2.9**(2.2)	**2.1**(2.3)	∞	∞	∞	∞ *3e4*	0/15
BFGS	69(76)	∞	∞	∞	∞	∞ *2e4*	0/15
VXQR1	38(46)	∞	∞	∞	∞	∞ *5e4*	0/15

6.6 Multimodal Functions with Weak Structure f_{20}–f_{24}

The ECDF graphs of ERT for the 5-D and the 20-D multimodal functions with weak structure f_{20}–f_{24} are aggregated in Figs. 3 and 4, respectively, in the bottom right part. Tables 9 and 10 contain the results for the 5-D and the 20-D functions, respectively.

For the 5-D space, NEWUOA needed $10^4\,D$ evaluations to solve about 55 % of the problems and it dominates the other algorithms up to this limit. Nevertheless, RA,

Table 10: ERT on f_{20}–f_{24} in 20-D over $\mathrm{ERT}_{\mathrm{best}}$ obtained in BBOB-2009. The NEWUOA is used as the baseline for statistical comparisons.

20 Schwefel x*sin(x)

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	82	46150	3.10e6	5.54e6	5.59e6	5.64e6	14/15
LSfminbnd	11(3.6)	5.9(6.1)	∞	∞	∞	∞ *2e5*	0/15
LSstep	280(59)	11(12)	∞	∞	∞	∞ *2e5*	0/15
NMSS	3.5(1.9)	∞	∞	∞	∞	∞ *2e5*	0/15
RA	**2.6**(0.57)	**2.9**(3.3)$^{\downarrow}$	∞	∞	∞	∞ *2e5*	0/15
NEWUOA	**1**(0.45)	15(18)	∞	∞	∞	∞ *4e5*	0/15
BFGS	**2.1**(0.38)	**5.8**(6.1)	∞	∞	∞	∞ *4e5*	0/15
VXQR1	**0.95**(0.24)	**1.1**(1.1)	∞	∞	∞	∞ *1e5*	0/15

21 Gallagher 101 peaks

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	561	6541	14103	14643	15567	17589	15/15
LSfminbnd	30(60)	27(35)	20(23)	19(23)	18(22)	16(19)	7/15
LSstep	124(185)	204(229)	202(234)	197(212)	187(206)	168(176)	1/15
NMSS	**7.7**(12)	20(19)	24(23)	23(22)	22(22)	20(19)	7/15
RA	7.8(10)	**7.6**(11)	**4.7**(6.0)	**4.5**(5.8)	**4.3**(5.4)	**3.8**(4.8)	14/15
NEWUOA	**1.7**(2.5)	**2.2**(2.1)	**1.2**(1.8)	**1.2**(1.8)	**1.1**(1.7)	**1**(1.5)	15/15
BFGS	**1.9**(3.6)	**5.5**(6.4)	**4.6**(5.7)	**4.5**(5.5)	**4.3**(5.2)	**7.3**(8.6)	2/15
VXQR1	39(89)	25(31)	21(25)	20(24)	19(23)	17(20)	4/15

22 Gallagher 21 peaks

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	467	5580	23491	24948	26847	1.35e5	12/15
LSfminbnd	59(71)	16(18)	37(40)	36(40)	34(37)	**7.2**(7.4)	3/15
LSstep	280(343)	∞	∞	∞	∞	∞ *2e5*	0/15
NMSS	17(35)	18(21)	61(67)	58(63)	54(58)	11(11)	2/15
RA	**3.4**(5.6)	**4.3**(5.5)	**12**(11)	**12**(12)	**11**(10)	**2.2**(2.0)	8/15
NEWUOA	**1**(1.2)	**4.9**(6.4)	**6.8**(8.1)	**6.4**(7.7)	**6.0**(6.9)	**1.2**(1.4)	7/15
BFGS	**2.5**(2.0)	**1.8**(2.0)	**8.1**(8.7)	**7.7**(8.9)	**10**(10)	14(16)	0/15
VXQR1	140(214)	55(63)	60(68)	57(62)	53(61)	11(12)	1/15

23 Katsuuras

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	3.2	1614	67457	4.89e5	8.11e5	8.38e5	15/15
LSfminbnd	4.4(5.8)	206(224)	∞	∞	∞	∞ *2e5*	0/15
LSstep	**2.2**(1.3)	81(67)	∞	∞	∞	∞ *2e5*	0/15
NMSS	**2.1**(3.6)	**3.3**(5.3)	**43**(46)	∞	∞	∞ *2e5*	0/15
RA	**1.7**(1.7)	**4.6**(7.2)	∞	∞	∞	∞ *8e4*	0/15
NEWUOA	12(8.3)	**3.5**(3.3)	**32**(35)	∞	∞	∞ *2e5*	0/15
BFGS	47(26)	304(330)	∞	∞	∞	∞ *1e5*	0/15
VXQR1	**1.3**(1.6)	39(45)	∞	∞	∞	∞ *1e5*	0/15

24 Lunacek bi-Rastrigin

Δftarget	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathrm{ERT}_{\mathrm{best}}$	1.34e6	7.48e6	5.19e7	5.20e7	5.20e7	5.20e7	3/15
LSfminbnd	∞	∞	∞	∞	∞	∞ *2e5*	0/15
LSstep	∞	∞	∞	∞	∞	∞ *2e5*	0/15
NMSS	∞	∞	∞	∞	∞	∞ *2e5*	0/15
RA	∞	∞	∞	∞	∞	∞ *2e5*	0/15
NEWUOA	∞	∞	∞	∞	∞	∞ *2e5*	0/15
BFGS	∞	∞	∞	∞	∞	∞ *1e5*	0/15
VXQR1	∞	∞	∞	∞	∞	∞ *1e5*	0/15

NMSS, VXQR1, or BFGS are comparable with NEWUOA for certain evaluations intervals in this range. However, NMSS, VXQR1, and BFGS eventually solved about 70 % of the problems, while the other algorithm reached a success rate of about 55 %. Gallagher's functions f_{21} and f_{22} were solved by all the algorithms (an exception being LSstep on f_{22}). Although NMSS, VXQR1, and BFGS reached similar success rates, VXQR1 and BFGS solved Schwefel's f_{20} (NMSS failed), while NMSS solved Katsuura's f_{23} (VXQR1 and BFGS failed). The f_{24} function, Lunacek bi-Rastrigin, was too hard for

Table 11: The number of functions (out of 24) for which the algorithm found the ultimate precision of 10^{-8} for at least 1 run (out of 15) on the function.

D	2	3	5	10	20	40
LSfminbnd	10	9	6	5	5	4
LSstep	11	9	7	5	6	5
NMSS	24	23	18	11	8	7
RA	20	16	13	8	5	5
NEWUOA	21	18	11	11	11	9
BFGS	17	14	11	8	7	6
VXQR1	22	17	15	8	6	6

any of the algorithms.

In the 20-D space, the performance of several algorithms is comparable up to $200D$ evaluations. They solved about 15 % of the problems by this limit. After $200D$ evaluations, NEWUOA dominated the other algorithms, followed by the RA and BFGS methods. All the algorithms eventually reached the success rate of 50 % with the exception of LSstep, which reached about 30 % only. Again, 40 % of the success rate can be accounted to solving the two Gallagher's functions. The f_{24} function remained unsolved.

7 Discussion by Algorithm

In this section, we look at the results from the point of view of the individual algorithms. A global view of the algorithm results is presented in Table 11. The table items describe the number of functions for which we are able to compute a finite ERT, i.e. the number of functions for which at least 1 instance was solved to the ultimate precision 10^{-8} by the respective algorithm.

The table shows only one very rough and particular view of the results. It can be seen that the NMSS is very effective for $D \leq 5$. For $D \geq 10$, NEWUOA starts to be better. The third most successful method is VXQR1, closely followed by RA, and BFGS. The LS methods are the least effective ones; LSstep has only slightly better results than LSfminbnd.

The following subsections present a detailed discussion of the results broken down by the main algorithms under the study: LS, RA, and VXQR1.

7.1 Discussion on Line Search methods

The axis-parallel line search methods were included to provide the baseline results, to show what can be accomplished by methods which rely on the assumption of separability. To reiterate, the LSfminbnd method uses univariate *local* search technique to identify a local optimum in each of the coordinates. Based on this description it is clear that it should work effectively and reliably on unimodal separable functions only. The results confirm it: LSfminbnd works on the functions f_1, f_2, and f_5. However, if the functions are well-conditioned (f_1 and f_5), there are usually better approaches (NEWUOA, BFGS) than trying to optimize the variables one by one—it is better to optimize all variables at once. LSfminbnd is, however, the best algorithm for f_2; this exception is caused by the ill-conditioning, which (together with the univariate nonlinear monotonic transformations of the design variables used by the COCO framework to break the symmetry of the functions) makes this function really hard for the approaches that use quadratic modeling. Here, it seems profitable to optimize each

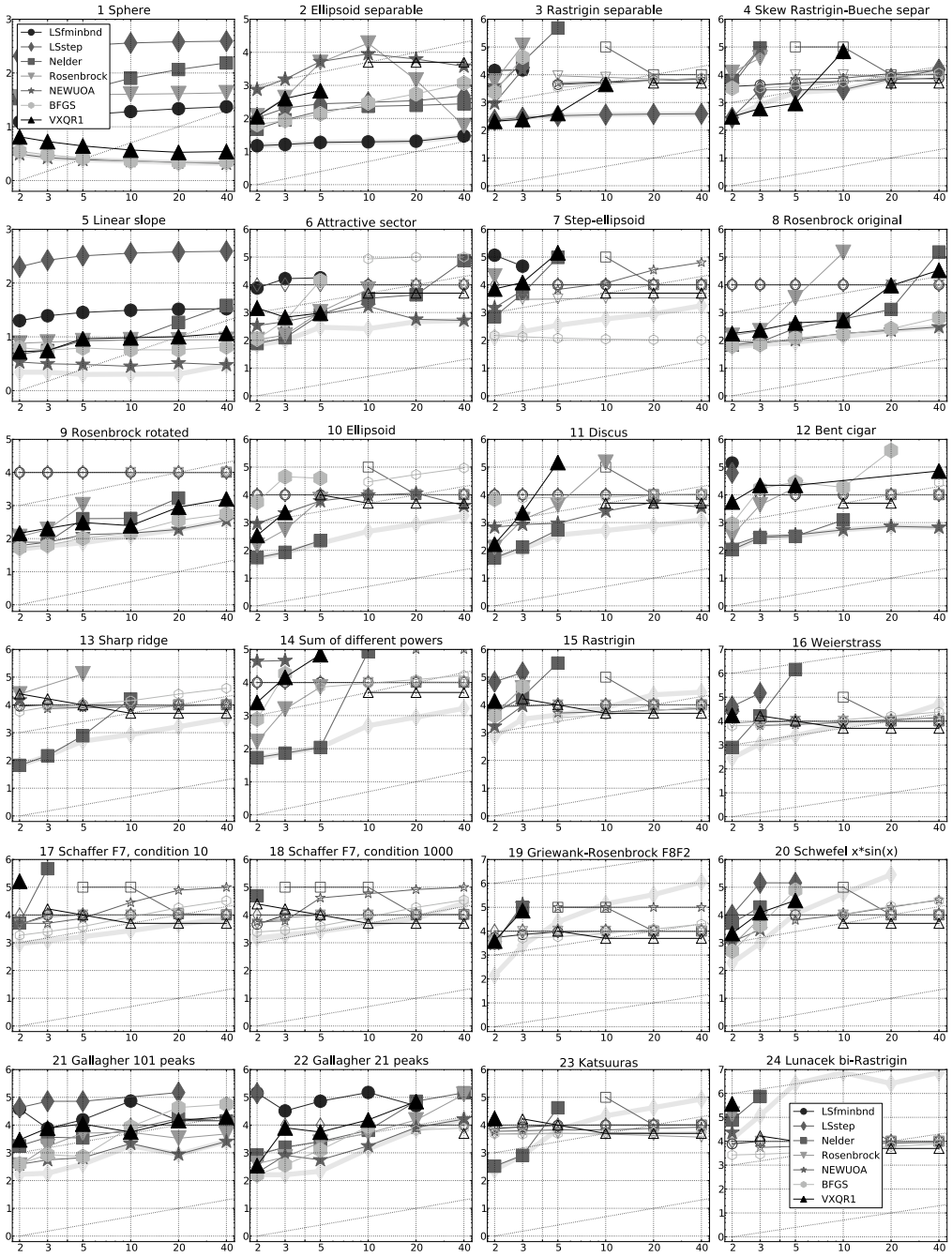

Figure 5: Expected running time (ERT) divided by dimension versus dimension in log-log presentation for the target function value 10^{-8} (filled symbols). Different symbols correspond to different algorithms given in legend of f_1 and f_{24}. Hollow symbols give the average number of function evaluations divided by the dimension when there is no success. Horizontal lines give linear scaling, dashed give quadratic. Legend: $\circ$: LSfminbnd, $\diamond$: LSstep, $\square$: NMSS, $\triangledown$: RA, $\star$: NEWUOA, $\hexagon$: BFGS, $\triangle$: VXQR1.

variable separately. On the other hand, if BFGS or NEWUOA were modified to estimate only the diagonal elements of the Hessian matrix, they are expected to provide competitive, or even better results than LSfminbnd.

The LSstep method uses a univariate *global* search technique to search for the coordinate-wise optima. This global method (STEP) is more demanding—it consumes more function evaluations to identify the optimum. It is generally much slower than LSfminbnd. This solver is effective on separable multimodal functions, exemplified by f_3 and f_4, which have an exponentially increasing number of local optima. Essentially, there is no other reasonable way to solve such functions except performing axis-parallel global line search. Despite this fact, there are still more efficient ways to perform such a search than the way implemented by LSstep—see the results of VXQR1 for dimensions $D = 2, 3, 5$ for the functions f_3 and f_4.

Note that purely separable functions are hardly encountered in practice. The practical value of these methods is thus limited. Nevertheless, in certain situations even these methods can perform better than some other conventional local optimizers (see the results for multimodal functions with adequate structure in lower dimensions). Note also that both the line search methods were run in $\langle -6, 6\rangle^D$, while the optimum lies in $\langle -5, 5\rangle^D$. They thus searched an unnecessarily large space (larger by a factor that increases with the dimension). Theoretically, they could provide better results if they searched the smaller hypercube only. The changes would be most distinguishable on functions that have their optimum on the space boundary, e.g on f_5, linear slope function. If the $\langle -5, 5\rangle^D$ space was used, the results of LSfminbnd on f_5 would be surely worse, since it virtually never samples the boundary points. On the other hand, the results of LSstep would be better on f_5, since it samples the boundaries. However, in general we do not expect any significant effect on the aggregated results.

7.2 Discussion on Rosenbrock's method

Rosenbrock's method is a simple adaptive pattern local search technique. It is able to rotate its internal coordinate system, it should be thus invariant with respect to rotation with the exception of initialization. However, if the function is separable for a certain rotation of coordinates, the right initialization can be of a great help to RA. For the unimodal problems, this can be seen on the results for the ellipsoidal functions f_2 and f_{10}. RA can solve f_2 in all dimensions $D = 2, \ldots, 40$, for 40-D it is almost as efficient as the champion for f_2, LSfminbnd. On the rotated version of the same problem, however, RA is only able to solve the problems for $D = 2, 3, 5$ and fails for higher dimensions.

Among the unimodal functions, the step-ellipsoid function f_7 is the hardest for RA. The plateaus of this function do not give the algorithm the necessary "signal" where to search next. On the other hand, the sharp ridge function f_{13} is hard for the other algorithms; RA solved it in 5-D and was the most successful algorithm for f_{13} in 20-D, probably thanks to the fact that it uses only the *better-than* relation and not the exact values of the objective function (it does not build a model of the objective function).

RA is really weak on multimodal functions. The only exceptions in this testbed are the Gallagher functions f_{21} and f_{22}, which were solved by RA in all tested dimensions. This suggests that RA converges quickly and can be restarted often enough to solve these functions.

7.3 Discussion on VXQR1

The VXQR1 method solved the functions f_1, f_5, f_8, and f_9 in all dimensions, i.e. functions with "well-behaved" graphs. On the other hand, VXQR1 failed to solve the func-

tions f_{13} and f_{18} (which were not solved in any dimension), and generally behaves poorly on multimodal functions (which were solved in low dimensions only, up to 5- or 10-D).

All instances of the separable functions f_1–f_5 were solved at least up to 5-D. This is consistent with the claim that VXQR1 works well for separable functions. For several target levels of f_3 and f_4, VXQR1 produced significantly better results than the best competitor in BBOB-2009. For $D > 10$, VXQR1 did not solve the functions f_2–f_4. This is due to the fact that the global line search algorithm (used for the coordinate searches) uses only a limited number of points independent of D. With increasing dimension, it becomes harder to achieve the required Δf_t, the number of points for the global line search may become insufficient, and VXQR1 therefore has difficulties with finding sufficiently good value in all the coordinates. In addition, the limit $500 \cdot \max(D, 10)$ on the number of function evaluations per independent call to VXQR1 favors dimensions $D < 10$. VXQR1 performed worse on the 20-D and 40-D problems than on the 10-D ones, but there was not much difference between 20-D and 40-D.

The unimodal non-separable functions f_6–f_{14} were generally solved by VXQR1 only up to 5-D; with increasing dimension, the reached Δf_best values also get larger. Both the Rosenbrock functions, f_8 and f_9, are exceptions—they were solved in all dimensions. For most of the other functions which were solved by VXQR1 at least in the low dimensions, the ERT grows very quickly with the dimension, cf. Fig. 5. For the f_7 function, step-ellipsoid, this can be explained by the statement in Neumaier et al. (2010) that VXQR1 typically performs poorly on functions where the graphs along many directions are piecewise constant. For the functions f_{10}, f_{11}, and f_{14}, the explanation is the ill-conditioning of these functions. The worst results were obtained for f_{13}, sharp ridge, which was not solved by VXQR1 in any of the tested dimensions, because the shape of the ridge is not amenable to quadratic modeling.

VXQR1 does not perform well on the multimodal functions f_{15}–f_{24} with the exception of both the Gallagher functions (which are easily solvable by many local optimization techniques with the help of restarting). VXQR1 was able to solve the majority of these functions in 2- or 3-D only. Again, the reached Δf_best values also get larger with increasing dimension. In many cases, VXQR1 obtains only non-global optima (some of them repeatedly). Neumaier et al. (2010) state that the VXQR class of algorithms aims for a rapid decrease of the objective function rather than for necessarily reaching the global minimum. It is thus clear from the algorithm design that VXQR1 will often reach only a non-global minimizer of a multimodal function.

The results are consistent with the claim that VXQR1 performs well for separable problems and problems with "well-behaved" graphs, but poorly on multimodal, piecewise constant, and rugged functions.

8 Summary and Conclusions

Several local optimization methods were compared in detail in this article. Some of them rely on univariate search procedures (LSfminbnd and LSstep) or use them as an integral part of its algorithm (VXQR1), some use an adaptive pattern of points to choose the next candidate point (Nelder-Mead and Rosenbrock's method), some of them build quadratic models of the function (LSfminbnd, VXQR1, NEWUOA, BFGS). Although there is no single winner of this comparison, a few interesting conclusions can be made.

The NMSS method, almost after 50 years from its birth, still belongs to the most efficient solvers among those compared for the low-dimensional spaces. In higher than 5-dimensional spaces, however, its efficiency quickly drops and from 10-D other meth-

ods are preferable.

NEWUOA is a carefully designed procedure with several years of active development. It paid off; for many unimodal problems in higher dimensions, it belongs to the best solvers in the BBOB-2009 comparison. It should be part of the workbench of every experimenter.

The methods discussed and compared here can enrich the evolutionary community in several other ways. Hybrids of EAs and local search procedures (often called memetic algorithms) can be constructed using the results presented in this paper. The compared algorithms may be useful as a part of other hybrid approaches like variable neighborhood search (Mladenovic, 1997), or various portfolio approaches (Peng et al., 2010). Moreover, many of the local search algorithms are much more successful in the initial stages of the search than EAs. It may be profitable to use their sampling process to initialize the population of EAs.

Acknowledgements. The authors would like to thank to N. Hansen, R. Ros, and A. Auger for the tremendous work during the development of the COCO framework, and L. Pál for his cooperation. The first author was supported by the Grant Agency of the Czech Republic with the grant no. 102/08/P094 entitled "Machine learning methods for solution construction in evolutionary algorithms". The authors wish to thank the anonymous reviewers for their useful constructive comments.

References

Finck, S., Hansen, N., Ros, R., and Auger, A. (2009a). Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions. Technical Report 2009/20, Research Center PPE.

Finck, S., Hansen, N., Ros, R., and Auger, A. (2009b). Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions. Technical Report 2009/20, Research Center PPE. Updated February 2010.

Hansen, N. (2009). Benchmarking the Nelder-Mead downhill simplex algorithm with many local restarts. In Rothlauf (2009), pages 2403–2408.

Hansen, N., Auger, A., Finck, S., and Ros, R. (2009a). Real-parameter black-box optimization benchmarking 2009: Experimental setup. Technical Report RR-6828, INRIA.

Hansen, N., Auger, A., Ros, R., Finck, S., and Pošík, P. (2010). Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. In *Workshop Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2010)*. ACM Press.

Hansen, N., Finck, S., Ros, R., and Auger, A. (2009b). Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Technical Report RR-6829, INRIA.

Hansen, N. and Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195.

Huyer, W. and Neumaier, A. (1999). Global optimization by multilevel coordinate search. *Journal of Global Optimization*, 14(4):331–355–355.

Huyer, W. and Neumaier, A. (2009). Benchmarking of MCS on the noiseless function testbed. `http://www.mat.univie.ac.at/~neum/papers.html`. P. 989.

McKinnon, K. I. M. (1999). Convergence of the Nelder-Mead simplex method to a nonstationary point. *SIAM Journal on Optimization*, 9:148–158.

Mladenovic, N. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

Moscato, P. (1989). On evolution, search, optimization, GAs and martial arts: toward memetic algorithms. Caltech Concurrent Comput. Prog. Report 826, California Institute of Technology, Pasadena, CA.

Nelder, J. and Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, 7(4):308–313.

Neumaier, A., Fendl, H., Schilly, H., and Leitner, T. (2010). VXQR: Derivative-free unconstrained optimization based on QR factorizations. *Soft Computing*. To appear.

Palmer, J. R. (1969). An improved procedure for orthogonalising the search vectors in Rosenbrock's and Swann's direct search optimisation methods. *The Computer Journal*, 12(1):69–71.

Peng, F., Tang, K., Chen, G., and Yao, X. (2010). Population-Based algorithm portfolios for numerical optimization. *IEEE Transactions on Evolutionary Computation*, 14(5):782–800.

Pošík, P. (2009a). BBOB-benchmarking the Rosenbrock's local search algorithm. In Rothlauf (2009), pages 2337–2342.

Pošík, P. (2009b). BBOB-benchmarking two variants of the line-search algorithm. In Rothlauf (2009), pages 2329–2336.

Pošík, P. (2010). Rosenbrock's algorithm: Should we reset the multipliers after each coordinate system update? Technical Report, Czech Technical University in Prague. Available online, `http://labe.felk.cvut.cz/~posik/papers/Rosenbrock/RosOrigVsRosMod.pdf`.

Powell, M. J. D. (2006). The NEWUOA software for unconstrained optimization without derivatives. *Large Scale Nonlinear Optimization*, pages 255–297.

Price, K. (1997). Differential evolution vs. the functions of the second ICEO. In *Proceedings of the IEEE International Congress on Evolutionary Computation*, pages 153–157.

Ros, R. (2009a). Benchmarking the BFGS algorithm on the BBOB-2009 function testbed. In Rothlauf (2009), pages 2409–2414.

Ros, R. (2009b). Benchmarking the NEWUOA on the BBOB-2009 function testbed. In Rothlauf (2009), pages 2421–2428.

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

Rothlauf, F., editor (2009). *Genetic and Evolutionary Computation Conference, GECCO 2009, Proceedings, Montreal, Québec, Canada, July 8-12, 2009, Companion Material*. ACM.

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

Solis, F. J. and Wets, R. J. B. (1981). Minimization by random search techniques. *MATHEMATICS OF OPERATIONS RESEARCH*, 6(1):19–30.

Swarzberg, S., Seront, G., and Bersini, H. (1994). S.T.E.P.: The Easiest Way to Optimize a Function. In *IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 519–524 vol.1.

Whitley, D., Mathias, K., Rana, S., and Dzubera, J. (1996). Evaluating evolutionary algorithms. *Artificial Intelligence*, 85:245–276.