Global Line Search Algorithm Hybridized with Quadratic Interpolation and Its Extension to Separable Functions

Petr Baudiš
Czech Technical University in Prague
Fac. of Electrical Eng., Dept. of Cybernetics
Technická 2, 16627 Prague 6, Czech Republic
baudipet@fel.cvut.cz

Petr Pošík*
Czech Technical University in Prague
Fac. of Electrical Eng., Dept. of Cybernetics
Technická 2, 16627 Prague 6, Czech Republic
petr.posik@fel.cvut.cz

ABSTRACT

We propose a novel hybrid algorithm "Brent-STEP" for univariate global function minimization, based on the global line search method STEP and accelerated by Brent's method, a local optimizer that combines quadratic interpolation and golden section steps. We analyze the performance of the hybrid algorithm on various one-dimensional functions and experimentally demonstrate a significant improvement relative to its constituent algorithms in most cases. We then generalize the algorithm to multivariate functions, proposing a scheme to interleave evaluations across dimensions to achieve smoother and more efficient convergence. We experimentally demonstrate the highly competitive performance of the proposed multivariate algorithm on separable functions of the BBOB benchmark. The combination of good performance and smooth convergence on separable functions makes the algorithm an interesting candidate for inclusion in algorithmic portfolios or hybrid algorithms that aim to provide good performance on a wide range of problems.

CCS Concepts

•Mathematics of computing → Solvers; Nonconvex optimization; •Computing methodologies → Continuous space search;

Keywords

Black-box optimization, Line search, Separable functions, Hybrid algorithm

1. INTRODUCTION

Continuous black-box optimization concerns finding a minimum of a function with no accessible analytical form. One class of multivariate functions investigated regarding black-box optimization are *separable* functions — that is, functions that can be decomposed such that $f(\vec{x}) = \sum_i f_i(x_i)$.

For some very hard separable functions, exploiting separability is the only way to quickly find the minimum.[1] A natural idea to optimize such functions is to use univariate optimization algorithms on individual dimensions. In [9], Brent's method (as implemented in MATLAB `fminbnd` function) and the STEP algorithm were used to separately optimize the function along each dimension. Brent's method was shown to be fast in case of unimodal functions, but due to its local nature it fails on multimodal functions. The global STEP method was able to solve both the uni- and multimodal functions, but needed much larger number of function evaluations. Moreover, their multidimensional variants were constructed inefficiently: the dimensions were optimized sequentially, one by one. As a result, the optimization process made hardly any significant progress until the algorithm started to optimize the last dimension. Another disadvantage of this solution is that the user must specify additional parameter, the budget for individual line searches.

Although these algorithms rely on the function separability and despite the above disadvantages, both of these methods proved to be useful in algorithmic portfolios [1] and hybrid algorithms [7] that strive to be successful on a broad range of functions, including the separable ones.

This paper builds on the above mentioned methods, and provides the following contributions:

1. We combine Brent's method and STEP into a single algorithm which converges faster than STEP (in many cases, it is almost as fast as Brent's method), while it preserves the global search ability of STEP (thus solving a larger proportion of functions than Brent's method, and often doing it faster).

2. We suggest a better way of making a multidimensional variant of this method. As opposed to solving the 1D problem in all dimensions sequentially, we propose to interleave the steps in individual dimensions, updating the full coordinates of sampled points based on results obtained in other dimensions so far.

The paper is organized as follows: In Sec. 2 we describe our hybrid univariate optimization algorithm and its multivariate extension. In Sec. 3 we outline our experimental setup for benchmarking both univariate and multivariate optimization performance, and in Sec. 4 we present and analyze the benchmark results. We conclude and outline future work in Sec. 5.

*Contact author

GECCO '15, July 11 - 15, 2015, Madrid, Spain

ISBN 978-1-4503-3472-3/15/07…$15.00

DOI: http://dx.doi.org/10.1145/2739480.2754717

[1] A good example is the Skew Rastrigin-Bueche (f4) in the BBOB benchmark [3, 6].

2. ALGORITHM PRESENTATION

The algorithm proposed in this paper is a hybrid of two techniques: Brent's method and STEP. Algorithms based on these two methods belong to the best performing BBOB-2009 algorithms for the class of separable functions [5]. Let us first shortly review the two methods. We will then outline the approach to combine them and consequent required modifications to the individual methods. Finally, we will consider extending them from univariate to multivariate optimization without optimizing dimensions one by one.

2.1 Brent's Method

Brent's method [2] is a classic local line search method enriching a golden section search with quadratic interpolation (QI) to speed up its convergence on functions with continuous second derivative. Detailed description of this standard algorithm can be found in [12, Sec. 10.2]. Its implementation is also part of many scientific toolboxes (see e.g. function `fminbnd` in MATLAB or `minimize_scalar` in scipy.optimize).

In each iteration of the Brent's method, a parabola is interpolated through the three best-so-far sampled points. The point at the minimum of the parabola is considered for the next sample, provided that it passes a convergence criterion which (roughly speaking) ensures that the samples converge to a single point by the virtue of each sample being closer to the minimum best-so-far point than to the other two.[2] If the point is not accepted, a golden section step splits the interval between the best-so-far point and the domain bound which contained the proposed point.

2.2 STEP

STEP [11] is the acronym of "Select the Easiest Point". It is a global line search method which iteratively divides the initial domain into increasing number of intervals by evaluating a point in the middle of one of them. The interval to be split into halves is chosen on the basis of its *difficulty* which estimates how hard it will be to find an improvement by sampling from that particular interval.

To estimate the difficulty of (finding an improvement inside) an interval, a quadratic model is used. Assume that a particular interval is defined by 2 previously sampled and evaluated boundary points, (x_1, f_1) and (x_2, f_2). A parabola $y(x)$ is estimated such that it intersects both boundary points and improves the best solution so far (f_{BSF}), such that $y(x) = y_{\min} = f_{\mathrm{BSF}} - \epsilon$ for some $x \in [x_1, x_2]$. The minimal curvature required for such a quadratic function is used as the *interval difficulty*.

2.3 Hybrid Brent-STEP Algorithm

The proposed hybrid combines Brent's and STEP methods in a *divide and conquer* manner. Both methods divide intervals into smaller parts. If possible, a step of local Brent's method is applied in a chosen search space part. If Brent's method does not seem to be profitable, a step of the global STEP method is applied.

In any time instant, the division of the search space into intervals is given by a sequence of points $(x_1, \ldots, x_N)$, $x_1 < x_2 < \ldots < x_N$. All these points are evaluated, their function values are $(f_1, \ldots, f_N)$, $f_i = f(x_i)$. Each pair of successive x-values defines an interval, each triple of successive x-values defines a so-called *neighboring intervals pair* (NIP). There are thus $N-1$ intervals, and $N-2$ NIPs.

Our hybrid algorithm repeatedly inspects the available intervals bounded by points sampled so far, checks if Brent's iteration seems profitable (see below) in any NIP, and falls back to the STEP method to perform a global optimization step if that is not the case.

Brent's branch: Since each NIP is defined by 3 evaluated points, it is easy to check if these points bracket a minimum (that is, $f_i > f_{i+1} < f_{i+2}$). In that case it is possible to use quadratic interpolation to find an estimate of the minimum inside this NIP. Among all the NIPs which bracket a minimum, we choose the NIP with the best estimated minimum.[3] If the estimated minimum is nonnegligibly better than the best solution so far ($y_{\min} \leq f_{\mathrm{BSF}} - \epsilon$), we make an iteration of Brent's method. Whereas the original Brent's method used the three best-so-far points for QI, we use the three NIP boundary points instead as we can apply Brent's step in a different NIP in each iteration.[4]

STEP branch: If there is no NIP promising to improve the best solution, we perform an iteration of the STEP method — i.e. the interval with the lowest difficulty is selected and a point in the middle of the interval is sampled.

This process is repeated until a solution of sufficient quality is found, the function evaluation budget is exhausted, or until there is no interval sufficiently large to be halved. A high-level description of this process is presented as Alg. 1.

Let us reiterate that this algorithm uses two different quadratic models for two different purposes:

1. In Brent's method, to estimate the profitability of using a QI step inside a NIP, the quadratic model is fit to three points (interval boundaries). The estimated minimum value of the parabola is used to judge whether it will be profitable to sample in this NIP; if so, the coordinate of the minimum is used as the next sampling point.

2. In STEP, to estimate the suitability of an interval to be split, the difficulty of an interval is computed by fitting a parabola in such a way that it must pass through both points defining that interval, and its minimum must lie on the level $f_{\mathrm{BSF}}-\epsilon$ somewhere in that interval. The coefficient of the quadratic term is then used as a measure of interval difficulty; the coordinate of the minimum is not used in any way.

In some cases (e.g. the f_{14} function shown below), Brent's method will zone in on the optimum, but eventually the expected improvement (as estimated by the quadratic interpolation) will become insufficient, and Brent's method is interrupted (although it would still be profitable to continue with the local search). The algorithm switches to STEP which will never sample the interval again (never triggering a possible Brent's method restart) because its difficulty

[2] In fact, there are more sample acceptance tests and the convergence criterion uses a heuristical time delay, but this is not relevant in our application.

[3] Unimodal functions will typically have only a single NIP bracketing a minimum.

[4] Considering the standard description of Brent's method, e.g. in [12], we use a modified variable assignment as follows: variables a and b are set to the boundary points of a NIP, x is set to the third point inside the NIP, w and v are set to the better and worse of a and b, respectively, u is set to x, and e is set to the size of the smaller interval in the pair.

Algorithm 1: Univariate Brent-STEP algorithm

Input: f – function to be optimized, $X = (x_i, f_i)_{i=1}^{3}$ – three evaluated points, such that x_1 and x_3 are lower and upper boundary, respectively, k – the period of unconditionally triggering Brent's iteration.

Output: $(x_{\mathrm{BSF}}, f_{\mathrm{BSF}})$ - estimate of the minimum.

```
begin
    t ⟵ 0
    (x_BSF, f_BSF) ⟵ best of X.
    while termination criteria are not met do
        t ⟵ t + 1
        B ⟵ choose the NIPs bracketing a minimum.
        i, y_min ⟵ find the most promising NIP in B.
        if y_min ≤ f_BSF − ϵ or mod(t, k) = 0 then
            x_s ⟵ sample from NIP i using Brent's iteration.
        else
            i ⟵ find interval with lowest difficulty.
            x_s ⟵ sample the middle point of interval i.
        f_s ⟵ f(x_s)
        (x_BSF, f_BSF) ⟵ update BSF using (x_s, f_s) if needed.
        Incorporate (x_s, f_s) into X such that
        x_1 < x_2 < ... < x_N.
```

measure is too unfavorable. Therefore, as an important relaxation of the conditions above, every k iterations we allow an iteration of Brent's method even if it is not estimated as a sufficient improvement to f_{BSF}.

2.4 Multivariate Generalization

As we already mentioned, solving D-dimensional separable problem as D independent 1D problems is a plausible idea, but not very efficient. Rate of convergence and especially its steadiness would be much improved if the solvers in individual dimensions could cooperate. Steady convergence from the beginning is especially important for expensive optimization scenarios and in portfolios with online algorithm selection based on performance within the same run.

The sequential multivariate versions of Brent's method and STEP (in [9] denoted as LSfminbnd and LSstep, respectively) worked in the following way. A random point was first chosen in the search space; we call this point *a context* (and this is the only stochastic part of the algorithm). The fitness function was then optimized along the first axis taking all the other coordinate values from the context. After the optimization in the first dimension was finished, the first coordinate of the context was updated to the best solution found during the univariate search. The same process was then repeated for all the remaining dimensions.

We extend this approach by interleaving the individual univariate solvers. A single dimension is chosen in a round-robin fashion, and the algorithm associated with that dimension performs a single iteration. If an improvement Δf_{BSF} of the best-so-far solution is found, the context is updated and propagated to all the other univariate solvers immediately to update their states. Moreover, each other solver updates (by function separability) the function values of all the points sampled so far by Δf_{BSF}.

A related STEP "semi-interleaving" method was used in the HCMA algorithm [7], but the context was updated only after a complete pass through all dimensions, which allowed the hybrid algorithm to test the separability of the function.

It is worth noting that we extensively experimented with various dimension selection strategies [8] "smarter" than round-robin, especially based on the minimal interval difficulty, but the improvement compared to the round-robin strategy was insignificant.

3. EXPERIMENTAL PROCEDURE

To study the behavior of the Brent-STEP (B-S) algorithm in *univariate search* in contrast to its constituent methods[5] (Brent and STEP), we evaluated the BBOB benchmark functions with scalar arguments, even though they are typically used for multivariate benchmarking; most of them can be successfuly used this way. We do not consider functions 8, 9, 17, 18, 19, 20 and 24 since these cannot be evaluated with scalar argument or yield a constant function. We also do not detail results for 1D versions of functions 2, 3, 11, 12, and 13, which are very similar in shape and results to some of the other reference functions which are part of the comparison.

To test *multivariate performance*, we compared the proposed interleaved multivariate Brent-STEP method (NDsqistep) with

- the interleaved version of the STEP method (NDstep), which should reveal the benefit of hybridizing STEP with Brent's method,
- the non-interleaved versions of Brent (LSfminbnd) and STEP (LSstep) as submitted for BBOB-2009 [9], which shall reveal the benefit of interleaving the interations in individual dimensions, and
- HCMA [7] which is a competitive hybrid algorithm combining BFGS, STEP and a CMA-ES variant (and uses semi-interleaved STEP on separable functions until non-separability is detected).

These algorithms are compared on the whole testbed again.

We ran experiments according to [4] on the benchmark functions given in [3, 6]. We used the BBOB2015 set of instances for conducting benchmarks.

We follow the STEP algorithm settings as described in [9], in particular we set $\epsilon = 10^{-8}$. To facilitate continued Brent runs on functions with poor STEP convergence as described above, we unconditionally trigger the Brent's method every $k = 10$ iterations. The algorithms are wrapped in a multi-start strategy that performs a random restart if the algorithm did not yield an improving result for 2000 iterations.

4. RESULTS AND DISCUSSION

First, we show results of the hybrid Brent-STEP algorithm on various univariate functions and analyze its performance relative to the constituent algorithms. Next, we turn to the multivariate case and compare the performance of Brent-STEP as we proposed it to other previously published algorithms focused on separable functions.

[5]We used our own Python STEP implementation for benchmarking STEP and the `bounded` scalar minimization method of SciPy 0.14.0 for benchmarking Brent's method.

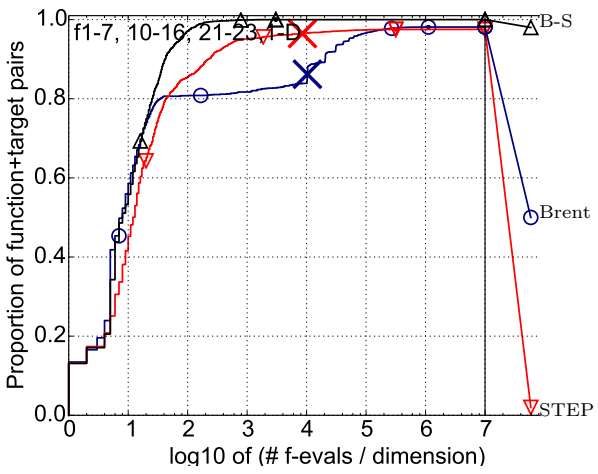

Figure 1: Bootstrapped empirical cumulative distribution of the number of objective function evaluations (FEvals) for 50 targets in $10^{[-8..2]}$ for all considered univariate functions.

The **expected running time (ERT)**, used in the figures and tables, depends on a given target function value, $f_t = f_{opt} + \Delta f$. It is computed over all relevant trials as the number of function evaluations executed during each trial while the best function value did not reach f_t, summed over all trials, and divided by the number of trials that actually reached f_t [4, 10]. **Statistical significance** is tested with the rank-sum test for a given target Δf_t using, for each trial, either the number of needed function evaluations to reach Δf_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.

4.1 Univariate Experiments

The empirical cumulative distribution function (ECDF) of ERT across all considered functions is shown in Fig. 1; the ECDF on individual functions is compared in Figure 2.

We can immediately observe that even though there are many functions for which either STEP or Brent exhibits some pathological behavior and has difficulty converging, the Brent-STEP algorithm converges on all of the considered functions, and moreover is always better than the worse of the constituent methods. The hybrid algorithm is therefore robust in this regard.

Looking at each of the detailed functions, we can notice three modes of behavior — either the hybrid algorithm closely matches the performance of the better of the methods (functions 1, 5, 7, 10, 14, 16, 21 and 22), it lags behind the better of the methods (only function 6), or the two methods cooperate in harmony on speeding up the search (highly multi-modal functions 4, 15, 23).

In case of function 6, we can observe that it is almost flat on one side of the optimum but very steep on the other side. Our modification of Brent does not perform well in this case since instead of the three best-so-far points, we fit the parabola through the three bracketing NIP boundary points. While the three best-so-far points would all lie on the flat side of the optimum, the rightmost NIP boundary

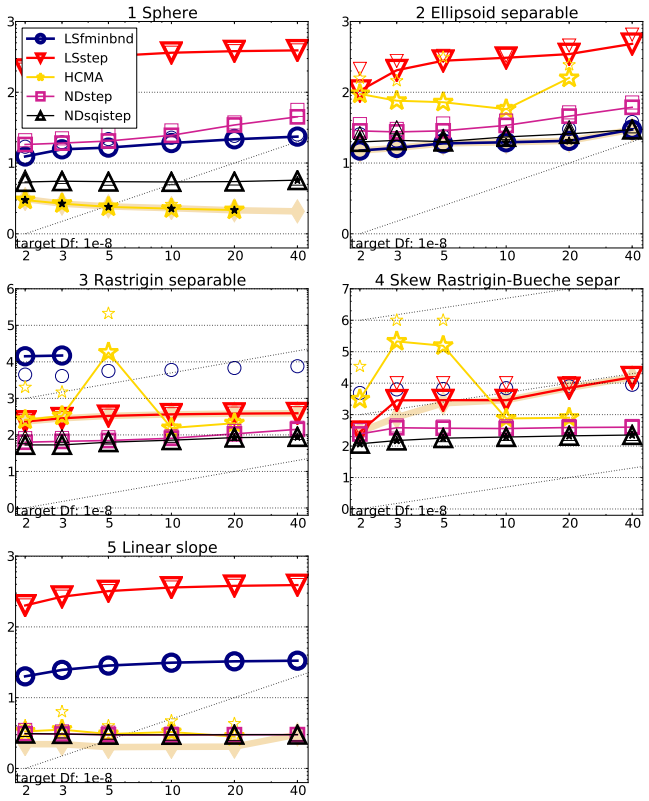

Figure 3: Expected running time (ERT in number of f-evaluations as $\log_{10}$ value), divided by dimension for target function value 10^{-8} versus dimension. Slanted grid lines indicate quadratic scaling with the dimension. Different symbols correspond to different algorithms given in the legend of f_1. Light symbols give the maximum number of function evaluations from the longest trial divided by dimension. Black stars indicate a statistically better result compared to all other algorithms with $p < 0.01$ and Bonferroni correction number of dimensions (six). Legend: ○:LSfminbnd, ▽:LSstep, ⋆:HCMA, □:NDstep, △:NDsqistep

remains at the steep side, causing repeated overshooting of the fitted parabola minimum to the flat side. Brent-STEP still converges and is faster than plain STEP, though.

The beneficial interplay of the two methods is most apparent on function 23, which has many local optima. The Brent algorithm, as a very local method, will always slide into the local optimum near the middle of the domain, while the STEP method will not explore the optima sufficiently. The combination of both methods enables simultaneous exploration of all local optima by Brent, with dynamic preference to spending evaluations on the most promising one.

4.2 Multivariate Experiments

The multivariate results are presented in Fig. 3 (expected running time — ERT scaling), Fig. 5 (convergence rate across many instances — ECDF) and in Tables 1 and 2. To highlight the effect of smooth convergence thanks to the interleaved dimension evaluation, we also show ERT scaling plotted for the "expensive" BBOB scenario in Figure 4, where

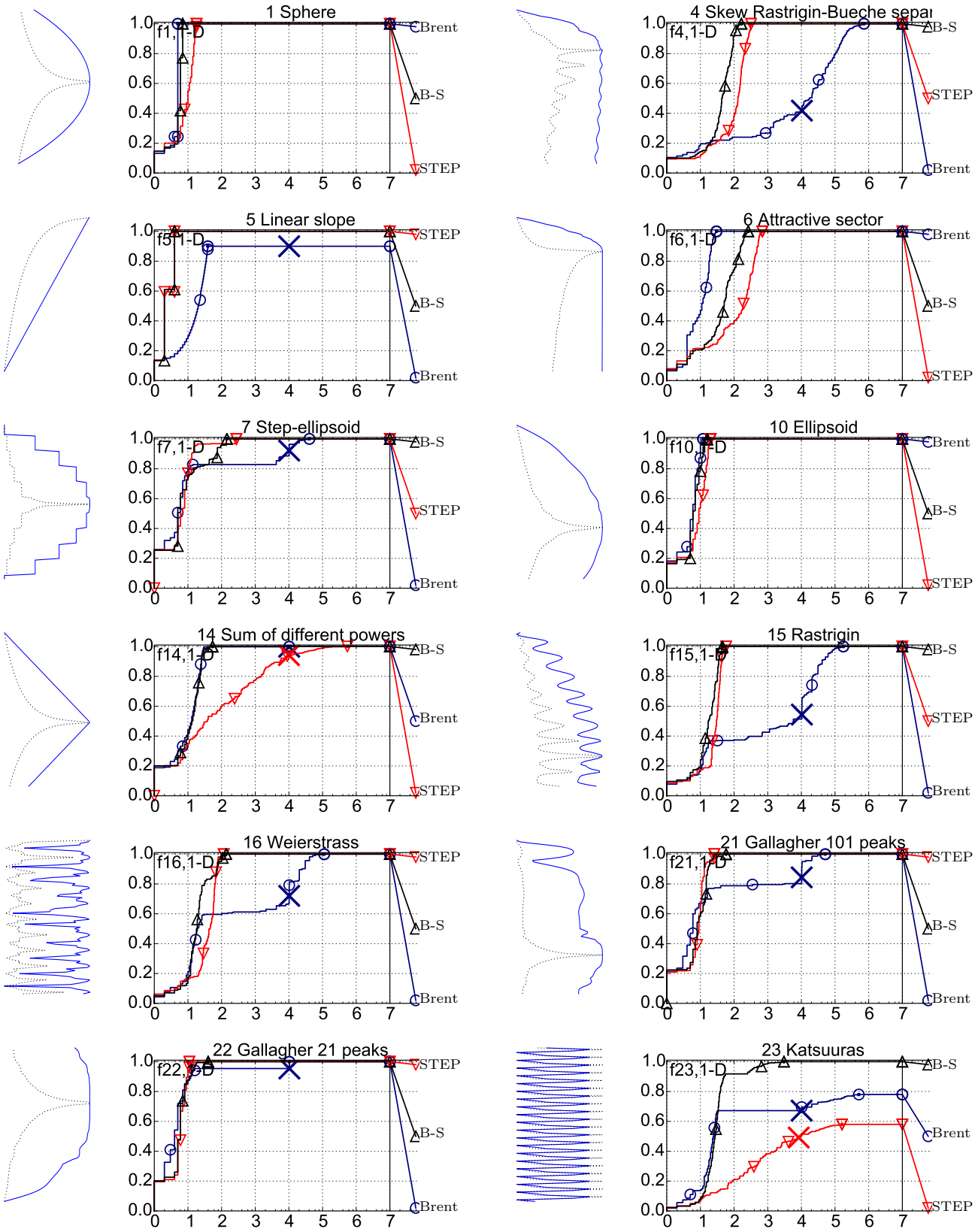

Figure 2: Bootstrapped empirical cumulative distribution (ECDF) of the number of objective function evaluations (FEvals) for 50 targets in $10^{[-8..2]}$ for each of the detailed univariate functions. The horizontal axis shows $\log_{10}$ number of FEvals required to reach the proportion of targets marked on the vertical axis. Near each ECDF graph is also shown a sketch (rotated sideways) of the corresponding univariate function in its whole domain; the dashed line shows the function log-scaled.

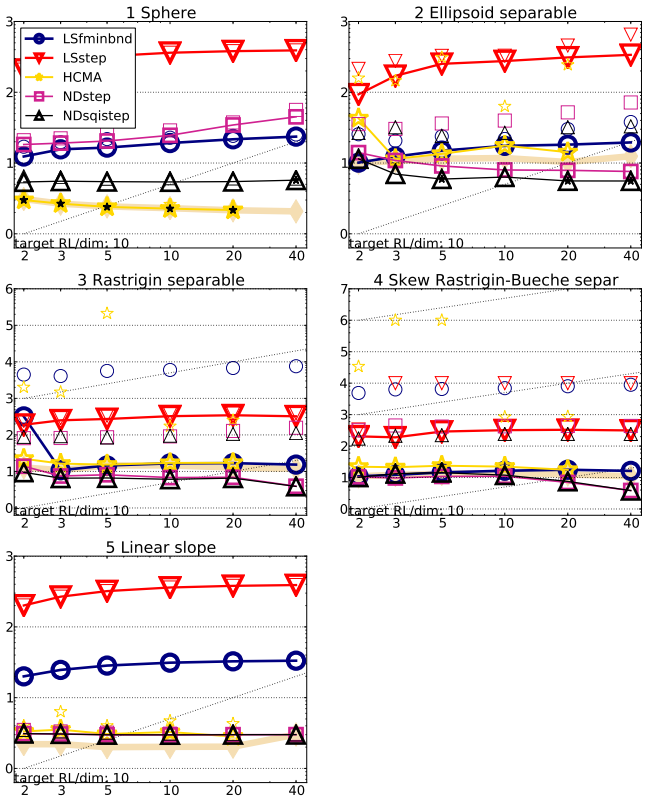

Figure 4: Expected running time (ERT in number of f-evaluations as $\log_{10}$ value) divided by dimension versus dimension. The target function value is chosen such that the bestGECCO2009 artificial algorithm just failed to achieve an ERT of $10 \times \mathrm{DIM}$. Different symbols correspond to different algorithms given in the legend of f_1. Light symbols give the maximum number of function evaluations from the longest trial divided by dimension. Black stars indicate a statistically better result compared to all other algorithms with $p < 0.01$ and Bonferroni correction number of dimensions (six). Legend: ○:LSfminbnd, ▽:LSstep, ⋆:HCMA, □:NDstep, △:NDsqistep

a substantial progress right from the beginning of the optimization is highly important.

Let us shortly discuss the results of HCMA. Figure 5 shows it has an advantage in the beginning of the search. The margin is caused by the use of BFGS method, which helps HCMA to optimize f1 very quickly (see Figs. 3 and 4). It also helps the method to quickly reach some less demanding targets on other functions. From this point of view, HCMA has a considerable advantage. On the other hand, being a general method aimed also at non-separable functions, HCMA must carefully decide whether it will use STEP or other constituent methods; this is not the case for the other methods with the assumption of independence hard-wired. From this point of view, HCMA has a significant disadvantage for comparison on separable functions.

The difference of NDstep (interleaved) to LSstep (non-interleaved) is predictable — NDstep converges comparably smoothly as apparent from the ECDF figure, and with lower ERT as there is no need to preemptively spend the whole alotted budget in each dimension.

Moving from STEP to Brent-STEP in multivariate case (the NDsqistep algorithm) has effects consistent with the univariate behavior — it always (except f5) improves the STEP variant, and on f2, where Brent's method dominated, it performs comparably.

Thanks to smooth convergence, the ERT for expensive scenario targets is also improved; on all non-trivial separable functions, the Brent-STEP algorithm improves the BBOB-2009 baseline.

5. CONCLUSION

We have reviewed two popular line search methods (Brent and STEP) that represent the BBOB-2009 baseline for performance on separable functions. We have introduced a new hybrid algorithm "Brent-STEP" combining these two methods non-trivially and demonstrated that on univariate and separable functions the hybrid algorithm in general outperforms both of them, in the univariate case often by a wide margin, and that it is behaving robustly even when one of the constituent methods is failing to converge.

Separable functions are not a very common class in practice, but we envision inclusion of the proposed algorithm in other hybrid methods (akin to HCMA introduced above) and in algorithm portfolios to efficiently handle the case of separable or near-separable functions. Even on non-separable functions, a short run of Brent-STEP might for example serve to generate an initial population for a followup evolutionary (or other optimization) algorithm run. To facilitate this direction of research, we are making our implementation of the algorithm available as open source.[6]

We consider efficient usage of Brent-STEP in larger algorithm ensembles to be the main course of future work. Brent-STEP also exhibits suboptimal performance in case of some pathologically shaped functions (see f6 above), that could be improved by some heuristic tweaks.

Acknowledgement

This work was supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS14/194/OHK3 /3T/13.

6. REFERENCES

[1] B. Bischl, O. Mersmann, H. Trautmann, and M. Preuß. Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 313–320. ACM, 2012.

[2] R. Brent. Algorithms for minimization without derivatives. *Prentice-Hall series in automatic computation*, 1973.

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

[4] N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking

[6] https://github.com/pasky/step

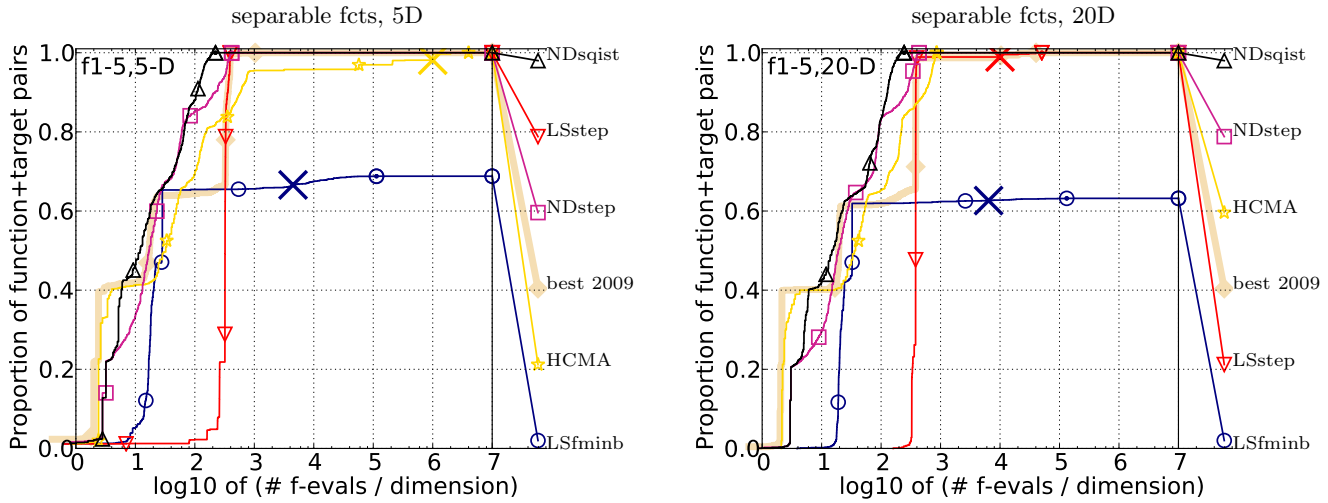

Figure 5: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 50 targets in $10^{[-8..2]}$ for all separable functions in 5-D and 20-D. The "best 2009" line corresponds to the best ERT observed during BBOB 2009 for each single target.

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f1	11	12	12	12	12	12	12	15/15
LSfminb	6.0(2)	6.3(2)	6.7(3)	6.7(3)	6.8(2)	6.8(2)	6.8(2)	15/15
LSstep	92(46)	121(0.0)	129(17)	132(0.0)	132(0.1)	132(0.1)	132(0.1)	15/15
HCMA	**1.1**(0)$^{\star 2}$	**0.98**(0)$^{\star 4}$	**0.98**(0)$^{\star 4}$	**0.98**(0)$^{\star 4}$	**0.98**(0)$^{\star 4}$	**0.98**(0)$^{\star 4}$	**0.98**(0)$^{\star 4}$	15/15
NDstep	1.6(0.2)	2.1(0.1)	2.8(0.1)	3.5(0.1)	4.2(0.2)	5.6(0.1)	6.8(0.3)	15/15
NDsqist	1.6(0.1)	1.9(0.2)	2.1(0.3)	2.2(0.2)	2.2(0.1)	2.2(0.2)	2.2(0.2)	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f2	83	87	88	89	90	92	94	15/15
LSfminb	1(0.2)	1(0.1)	1(0.1)	1(0.1)	1(0.1)	1(0.1)	**1**(0.2)	15/15
LSstep	16(3)	16(0.9)	16(2)	15(2)	15(2)	15(2)	15(0.1)	15/15
HCMA	1.5(0.1)	1.6(0.2)	1.8(0.2)	2.0(0.2)	2.2(0.3)	2.5(0.3)	2.8(0.2)	15/15
NDstep	0.72(0.1)$_{\downarrow 4}$	0.81(0.0)$_{\downarrow 3}$	0.88(0.1)	0.97(0.1)	1.1(0.1)	1.2(0.1)	1.4(0.1)	15/15
NDsqist	**0.56**(0.1)$^{\star}_{\downarrow 4}$	**0.59**(0.1)$^{\star 3}_{\downarrow 4}$	**0.63**(0.1)$^{\star 3}_{\downarrow 4}$	**0.72**(0.1)$^{\star 3}_{\downarrow 3}$	**0.79**(0.2)$^{\star 2}_{\downarrow 2}$	**0.90**(0.2)	1.0(0.2)	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f3	716	1622	1637	1642	1646	1650	1654	15/15
LSfminb	1(1)	52(49)	∞	∞	∞	∞	∞ *2e4*	0/15
LSstep	2.2(3e-3)	1(8e-3)	1(0.0)	1(0.0)	1(9e-3)	1(0.0)	1(8e-3)	15/15
HCMA	0.29(0.2)	3.0(2)	55(44)	54(43)	55(0.1)	55(320)	55(86)	15/15
NDstep	0.12(0.0)	**0.12**(0.0)	**0.14**(0.0)	**0.15**(0.0)	**0.16**(0.0)	**0.17**(0.0)	0.20(0.0)	15/15
NDsqist	**0.09**(0.0)$_{\downarrow}$	0.13(0.0)	0.16(0.0)	0.17(0.0)	0.17(0.1)	0.18(0.0)	**0.18**(0.0)	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f4	809	1633	1688	1758	1817	1886	1903	15/15
LSfminb	7.8(4)	∞	∞	∞	∞	∞	∞ *2e4*	0/15
LSstep	2.0(7e-3)	1(7e-3)	1(0.0)	1(0.0)	1(0.0)	1(0.1)	1(0.1)	15/15
HCMA	0.29(0.2)$_{\downarrow 3}$	74(263)	457(1482)	439(2134)	425(688)	410(663)	406(0.2)	13/15
NDstep	**0.14**(0.0)$_{\downarrow 4}$	0.24(0.1)$_{\downarrow 4}$	0.40(0.1)	0.53(0.1)	0.61(0.1)	0.88(0.1)	0.94(0.1)	15/15
NDsqist	0.15(0.1)$_{\downarrow 4}$	**0.21**(0.1)$_{\downarrow 4}$	**0.29**(0.1)*	**0.29**(0.1)$^{\star 3}$	**0.30**(0.1)$^{\star 4}$	**0.32**(0.1)$^{\star 4}$	**0.40**(0.1)$^{\star 4}$	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f5	10	10	10	10	10	10	10	15/15
LSfminb	13(3)	14(0)	14(0)	14(0)	14(0)	14(0)	14(0)	15/15
LSstep	141(40)	160(0.1)	160(0.1)	160(0.1)	160(0.1)	160(0.1)	160(0.1)	15/15
HCMA	**1.3**(0.1)$^{\star 2}$	**1.4**(0.2)	1.5(0.3)	1.5(0.2)	1.5(0.3)	1.5(0.2)	1.5(0.2)	15/15
NDstep	1.5(0.1)	1.5(0.1)	**1.5**(0.1)	**1.5**(0.1)	**1.5**(0.1)	**1.5**(0.1)	**1.5**(0.1)	15/15
NDsqist	1.5(0.1)	1.5(0.1)	**1.5**(0.1)	**1.5**(0.1)	**1.5**(0.1)	**1.5**(0.1)	**1.5**(0.1)	15/15

Table 1: Expected running time (ERT in number of function evaluations) divided by the respective best ERT measured during BBOB-2009 in dimension 5. The ERT and in braces, as dispersion measure, the half difference between 90 and 10%-tile of bootstrapped run lengths appear for each algorithm and target, the corresponding best ERT in the first row. The different target Δf-values are shown in the top row. #succ is the number of trials that reached the (final) target $f_{\mathrm{opt}} + 10^{-8}$. The median number of conducted function evaluations is additionally given in *italics*, if the target in the last column was never reached. Entries, succeeded by a star, are statistically significantly better (according to the rank-sum test) when compared to all other algorithms of the table, with $p = 0.05$ or $p = 10^{-k}$ when the number k following the star is larger than 1, with Bonferroni correction by the number of instances.

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f1	43	43	43	43	43	43	43	15/15
LSfminb	9.3(2)	10(1)	10(1)	10(1.0)	10(1)	10(1.0)	10(1)	15/15
LSstep	164(14)	175(2)	176(2)	177(0.0)	177(0.0)	177(0.0)	177(0.0)	15/15
HCMA	**1.00**(0.0)$^{\star 4}$	**1.0**(0.0)$^{\star 4}$	**1.0**(0.0)$^{\star 4}$	**1.0**(0.0)$^{\star 4}$	**1.0**(0.0)$^{\star 4}$	**1.0**(6e-3)$^{\star 4}$	**1.0**(6e-3)$^{\star 4}$	15/15
NDstep	2.1(0.1)	2.8(0.1)	3.6(0.1)	4.4(0.1)	5.1(0.0)	6.7(0.1)	8.3(0.1)	15/15
NDsqist	1.9(0.1)	2.3(0.1)	2.5(0.3)	2.5(0.3)	2.5(0.2)	2.5(0.3)	2.5(0.3)	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f2	385	386	387	388	390	391	393	15/15
LSfminb	1(0.1)	1(0.0)	1(0.1)	1(0.1)	1(0.1)	**1**(0.1)	**1**(0.0)	15/15
LSstep	17(0.7)	17(0.4)	17(0.4)	17(0.0)	17(0.2)	17(0.2)	17(0.2)	15/15
HCMA	1.3(0.0)	1.5(0.1)	1.7(0.1)	1.8(0.1)	2.1(0.2)	2.4(0.2)	2.8(0.3)	15/15
NDstep	0.69(0.0)$_{\downarrow 4}$	0.78(0.0)$_{\downarrow 4}$	0.88(0.2)	0.95(0.1)	1.1(0.1)	1.2(0.0)	1.4(0.1)	15/15
NDsqist	**0.59**(0.1)$^{\star 2}_{\downarrow 4}$	**0.65**(0.1)$^{\star 3}_{\downarrow 4}$	**0.69**(0.1)$^{\star 3}_{\downarrow 4}$	**0.79**(0.1)$^{\star 2}_{\downarrow 4}$	**0.87**(0.1)$^{\star 3}_{\downarrow 3}$	1.0(0.2)	1.1(0.1)	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f3	5066	7626	7635	7637	7643	7646	7651	15/15
LSfminb	∞	∞	∞	∞	∞	∞	∞ *1e5*	0/15
LSstep	1.5(0.1)	1(1e-3)	1(2e-3)	1(2e-3)	1(2e-3)	1(1e-3)	1(2e-3)	15/15
HCMA	0.26(0.0)$_{\downarrow 4}$	0.37(0.1)$_{\downarrow 4}$	0.46(0.1)$_{\downarrow 4}$	0.47(0.1)$_{\downarrow 4}$	0.49(0.1)$_{\downarrow 4}$	0.51(0.1)$_{\downarrow 4}$	0.53(0.1)	15/15
NDstep	**0.12**(0.0)$_{\downarrow 4}$	**0.15**(0.0)$_{\downarrow 4}$	0.19(0.0)$_{\downarrow 4}$	0.20(0.0)$_{\downarrow 4}$	0.21(0.0)$_{\downarrow 4}$	0.22(0.0)$_{\downarrow 4}$	0.23(0.0)$_{\downarrow 4}$	15/15
NDsqist	0.14(0.0)$_{\downarrow 4}$	0.16(0.0)$_{\downarrow 4}$	**0.19**(0.1)$_{\downarrow 4}$	**0.19**(0.0)$_{\downarrow 4}$	**0.20**(0.0)$_{\downarrow 4}$	**0.20**(0.0)$_{\downarrow 4}$	**0.21**(0.1)$_{\downarrow 4}$	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f4	4722	7628	7666	7686	7700	7758	1.4e5	9/15
LSfminb	∞	∞	∞	∞	∞	∞	∞ *1e5*	0/15
LSstep	1.6(8e-4)	1(4e-3)	1(5e-3)	1(7e-3)	1(9e-3)	1(0.0)	1(1)	9/15
HCMA	0.42(0.1)$_{\downarrow 4}$	0.67(0.0)	0.90(0.1)	1.1(0.1)	1.5(0.1)	1.8(0.2)	0.11(0.0)	15/15
NDstep	0.19(0.0)$_{\downarrow 4}$	0.30(0.1)$_{\downarrow 4}$	0.46(0.1)$_{\downarrow 4}$	0.60(0.1)	0.72(0.1)	0.86(0.1)	0.05(8e-3)	15/15
NDsqist	**0.18**(0.1)$_{\downarrow 4}$	**0.29**(0.1)$_{\downarrow 4}$	**0.31**(0.0)$^{\star 3}_{\downarrow 4}$	**0.32**(0.0)$^{\star 4}_{\downarrow 4}$	**0.33**(0.0)$^{\star 4}_{\downarrow 4}$	**0.36**(0.1)$^{\star 4}_{\downarrow 4}$	**0.02**(3e-3)$^{\star 4}_{\downarrow 4}$	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f5	41	41	41	41	41	41	41	15/15
LSfminb	16(0.4)	16(6e-3)	16(0)	16(0)	16(0)	16(0)	16(0)	15/15
LSstep	185(5)	187(2)	187(0.0)	187(0.0)	187(0.0)	187(0.0)	187(0.0)	15/15
HCMA	**1.2**(0.1)$^{\star 4}$	**1.4**(0.4)	**1.4**(0.4)	**1.4**(0.3)	**1.4**(0.4)	**1.4**(0.4)	**1.4**(0.2)	15/15
NDstep	1.5(0.0)	1.5(0.0)	1.5(0.0)	1.5(0.0)	1.5(0.0)	1.5(0.0)	1.5(0.0)	15/15
NDsqist	1.5(0.0)	1.5(0.0)	1.5(0.0)	1.5(0.0)	1.5(0.0)	1.5(0.0)	1.5(0.0)	15/15

Table 2: Expected running time (ERT in number of function evaluations) divided by the respective best ERT measured during BBOB-2009 in dimension 20. The ERT and in braces, as dispersion measure, the half difference between 90 and 10%-tile of bootstrapped run lengths appear for each algorithm and target, the corresponding best ERT in the first row. The different target Δf-values are shown in the top row. #succ is the number of trials that reached the (final) target $f_{\text{opt}} + 10^{-8}$. The median number of conducted function evaluations is additionally given in *italics*, if the target in the last column was never reached. Entries, succeeded by a star, are statistically significantly better (according to the rank-sum test) when compared to all other algorithms of the table, with $p = 0.05$ or $p = 10^{-k}$ when the number k following the star is larger than 1, with Bonferroni correction by the number of instances.

2012: Experimental setup. Technical report, INRIA, 2012.

[5] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík. 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, 2010.

[6] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Technical Report RR-6829, INRIA, 2009. Updated February 2010.

[7] I. Loshchilov, M. Schoenauer, and M. Sebag. Bi-population cma-es agorithms with surrogate models and line searches. In *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*, pages 1177–1184. ACM, 2013.

[8] P. Pošík and P. Baudiš. Dimension selection in axis-parallel Brent-STEP method for black-box optimization of separable continuous functions. In *Proceedings of the 2015 Conference on Genetic and Evolutionary Computation Companion*, New York, NY, USA, 2015. ACM. Accepted for publication.

[9] P. Pošík. BBOB-benchmarking two variants of the line-search algorithm. In *GECCO '09: Proceedings of the 11th annual conference companion on Genetic and evolutionary computation conference*, pages 2329–2336, New York, NY, USA, 2009. ACM.

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

[11] S. Swarzberg, G. Seront, and H. Bersini. Step: The easiest way to optimize a function. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 519–524, 1994.

[12] W. T. Vetterling, S. A. Teukolsky, and W. H. Press. *Numerical recipes in C (2nd edition)*. Press Syndicate of the University of Cambridge, 1992.