# Adaptability of Algorithms for Real-Valued Optimization

Mike Preuss

Technische Universität Dortmund, Chair of Algorithm Engineering
44221 Dortmund, Germany
mike.preuss@tu-dortmund.de

**Abstract.** We investigate the adaptability of optimization algorithms for the real-valued case to concrete problems via tuning. However, the focus is not primarily on performance, but on the tuning potential of each algorithm/problem system, for which we define the *empirical tuning potential* measure (ETP). It is tested if this measure fulfills some trivial conditions for usability, which it does. We also compare the best obtained configurations of 4 adaptable algorithms (2 evolutionary, 2 classic) with classic algorithms under default settings. The overall outcome is quite mixed: Sometimes adapting algorithms is highly profitable, but some problems are already solved to optimality by classic methods.

## 1 Introduction

The *evolutionary computation* (EC) field is on the move. After years of self-containedness, the necessity to demonstrate the effectiveness of *evolutionary algorithms* (EAs) in direct comparison to other optimization algorithms which are frequently applied for solving engineering problems has become obvious. However, such studies do already exist, as e.g provided by Schwefel already in 1979 [14]. Since then, the usage of many algorithms or benchmark problem sets has almost remained the same. However, the enormously increased computing power entailed a matching growth in the availability of experimental results, which lead to some new insight.

Triggered by several warning voices who critisized the arbitrariness of published EC algorithm comparison studies (one of which belongs to Eiben and Jelasity [9]), researchers of the field have put some effort into strengthening the experimental methodology, e.g. by standardization of test problems and performance measures. Additionally, the enormous effects of algorithm parametrization have been unveiled, and consequently, tuning methods were presented, like the *sequential parameter optimization* (SPO) [2], or the F-race by Birattari et al. [6], the latter being especially well suited for combinatorial optimization. Other approaches like meta-EAs tackle the same goal. These methods adapt the parameters of an algorithm to a concrete setting which shall contain a single problem or small problem class, the desired performance measure, and other environmental conditions as a maximum run length. On a first glance, the best achieved performances are of great value, because they let us compare algorithms under fair conditions, approximately at the top of their capacity.

Provided with automated tuning methods, we may also tackle the question for the *adaptability* of optimization algorithms towards a given situation, with the overall task to generalize on this algorithm property. Adaptability does not only consider top performance, but also the relative gain when starting from an 'average' (default?) algorithm configuration, as well as the effort needed to get there. To give an example, algorithm A may perform reasonably well on some problems, but does not allow for much adaptation because it does not provide any parameters which can be employed as handles for modifying the algorithm. Algorithm B may perform worse under default parameter values, but enables changing its behavior so as to obtain much better results by means of a tuning process. Initially, there is good reason to prefer algorithm A, but if tuning can be applied, algorithm B may turn out to be much better due to its adaptability.

Of course, an optimization algorithm may also be improved by introducing new or modified operators reflecting problem knowledge. However, there is little chance to do that in an automated and thereby measurable way, whereas tuning methods may be applied as soon as some parameters are available which can be adapted. Yet, it is hard to measure adaptability without imposing too many constraints on the employed problems. It shall be possible even in situations when the global optimum (of the optimization problem) is not known. It should also not depend on detecting optimal parameters for an algorithm, as this cannot be guaranteed by the tuning methods. We need to define a measure and then do an empirical investigation to see if the definition proves worthwhile. This work provides a first attempt into that direction and obeys the following scheme: §2 concretizes the aims pursued and introduces the chosen test problems and optimization algorithms. §3 discusses the meaning of parameters and suggests an adaptability measure. §4 reports on a thorough experimental investigation, followed by conclusions.

## 2 Aims and Methods

We want to investigate two related aspects of adaptability:

- How may adaptability be measured, and what do the measures tell us, especially concerning 'classic' (e.g. quasi-Newton) optimization algorithms?
- For any single algorithm, does it really pay off to tune? Is there a significant difference between default and tuned configurations' performances?

For automated parameter tuning, we employ the *sequential parameter optimization* (SPO) [2] algorithm, in the variant suggested in [3]. Starting from a *latin hypercube sample* (LHS) based kriging model, it uses random permutation tests to decide if a suggested new configuration is better than an old one. The maximum budget is always 1000 algorithm runs, with a minimum of 4 repeats. For more than 2 parameters, a grid search would be much more expensive. The tuning results are investigated using Kendall's non-parametric tau test for correlations; random permutation tests are employed for detecting significant differences between different algorithm configurations.

**Table 1.** Selected problems from the CEC 2005 library and some of their properties. No problem is separable and/or unimodal. All problems used in 10 dimensions.

| no. | problem | multimodal | characteristic | optimum | domain |
|---|---|---|---|---|---|
| 6 | shifted Rosenbrock's | yes | narrow valley from local to global optimum | 390 | -100/100 |
| 10 | shifted rotated Rastrigin's | highly | regular local optima structure | -330 | -5/5 |
| 12 | Schwefel's 2.13 | yes | best optima are far from each other | -460 | $-\pi/\pi$ |
| 13 | (F8F2) shifted expanded Griewank's plus Rosenbrock's | extremely | global structure very flat | -130 | -5/5 |
| 16 | rotated hybrid composition 1 | extremely | irregular with local patterns and plateaus | 120 | -5/5 |
| 17 | rotated hybrid composition 1 with fitness noise | extremely | same as 16 plus noise | 120 | -5/5 |
| 23 | non-continuous rotated hybrid composition 3 | highly | local patterns with many plateaus | 360 | -5/5 |

### 2.1 Test Problems

We select some problem instances from the CEC 2005 contest library [15], as enlisted in table 1. These may all be considered as hard, with different degrees of multimodality, and some additional properties like non-continuity and added noise. None of the problems is separable. This choice is motivated by the hope to discriminate the performance of the algorithms according to different problem characteristics. We acknowledge that the chosen instances are generally not very well suited for the classic optimization methods like the one of *Broyden-Fletcher-Goldfarb-Shanno* (BFGS). However, by allowing multistarts, in principle every method can reach the global optimum, and the CEC 2005 contest was intended to be hard, as simple and smooth problems are the typical domain of gradient-based optimization algorithms.

### 2.2 Algorithms

For the classical optimization algorithms, we basically rely on the *optim* method in the R stats package, which provides a quasi-Newton BFGS, a *conjugate gradients* (CG), and a Nelder-Mead [13] method. These three are derived from the implementation of Nash [12]. Furthermore, we employ the L-BFGS-B algorithm of Byrd et al. [7] and the *simulated annealing* (SA) variant of Belisle [4] contained in optim. Additionally, the alternative BFGS (hereafter denoted ALT-BFGS) implementation of Clausen [8] is used, as it provides access to several parameters usually hidden in backend libraries.

From the domain of metaheuristics, we test two related evolutionary algorithms, namely a generic *evolution strategy* (ES) with self adaptation as suggested by Schwefel [5], and the *covariance matrix adaptation evolution strategy* (CMA-ES) by Hansen and Ostermeier [10].

## 3 Parameters and Adaptability

What does it conceptually mean to adapt parameters of optimization algorithms? We shall question an often voiced opinion which marks large parameter sets as bad and strives for reducing their size. So what are the advantages and disadvantages of parameterized (optimization) algorithms? On the negative side, we have at least two:

– A large set of parameters makes a method more difficult to handle, because at least the unexperienced user does not know how to set these right.
– Parameter-parameter and parameter-problem interactions may enormously complicate evaluating algorithm performance, also making it more difficult to obtain good parameter settings.

Nevertheless, we here want to further a slightly unorthodox view of parameters, namely as handles to modify algorithms as desired. Simple standard algorithms are usually considered as being parameterless, but may easily be extended into paramerized ones (e.g. clever-quicksort). However, it is the advantage of simple algorithms with only one parameter that one can often prove a particular parameter value to deliver optimal performance. For evolutionary algorithms running on different optimization problems, this approach is not feasible. This is partly due to their rather complex stochastic behavior, but also stems from the inherent imprecision of the term *algorithm* as it is often applied to EAs. A concrete algorithm is instantiated only if the two following conditions are met.

1. The problem is clearly specified (not e.g., an ES applied to TSP).
2. All (exogenous) parameter values are fixed.

If this is not the case, one actually deals with an algorithm family, and one usually does so based on theoretical or empirical findings obtained from single algorithm instances. Whereas the first condition leads us to an uninevitable dilemma between practical usability and generalizability and thus between the real-world application and the scientific approach, the second one can be dealt with e.g. by parameter tuning. But still, it is not obvious which of the resulting tuned instances a scientific inquiry shall be based on. The best that is obtained by a concrete tuning method with a given time limit? Or a reasonably good (e.g. average) one? De Jong [11] points out that 'getting in the ball park' by achieving a reasonably well performing parameter setting is often sufficient for practical purposes, and that doing so may not be too difficult because of a certain robustness of EAs towards parameter changes. However, even for classic optimization algorithms for the real-valued domain, adapting parameters to the concrete problem may be advantageous. This is probably so for highly multimodal or otherwise considered difficult problems. Nevertheless, adaptation comes at a cost, and one shall take into account how difficult it is to 'get into the ball park', as well as the achieved performance differences relative to default parameter configurations.

## 3.1 Adaptability

When treating a real-world problem, (evolutionary) optimization algorithms are often adapted after a simple scheme. Based on the experience of the algorithm designer, a canonical EA is adjusted according to the interpretation of first results, thereby largely following intuitive reasoning. However, no generally applicable structured method exists to attain a good optimization algorithm for a yet untreated problem. Besides, evolutionary algorithms are so flexible that it seems unreasonable to approach such a method. Parameter tuning may help to a certain extent, but only insofar as a) there is something to tune, that is, the employed operators are parametrized, and b) the chosen components of the tuned algorithm are well suited to the problem at hand.

However, availability of fairly automated tuning procedures enables us to look at the adaptation process from the opposite direction: *What does parameter tuning of an optimization algorithm towards one or several problems tell us about the algorithm?* Applying a tuning method lets us obtain an estimate for peak and average performance. It surely depends on the employed performance measure how these two values can be aggregated into one, but this quantity may then serve as a measure for the tuning potential of an algorithm-problem system. However, we also need to make the assumption that the peak performance detected via tuning is a good estimator of the achievable maximum. The accuracy of this claim clearly depends on the quality of the employed tuning method, and its proper use. Time may be an important factor that limits tuning quality, because tuning requires multiple optimization algorithm runs, which is very costly for a non-trivial problem. Leaving these objections aside, and presuming that we have obtained two samples representing peak $\mathbf{y}_p$ and average performance $\mathbf{y}_a$, we suggest to compute an *empirical tuning potential* (ETP) measure (for minimization) from these samples. We deliberately abstain from using any 'target' performance as it is not generally known.

$$\mathrm{ETP}(\mathbf{y}_p, \mathbf{y}_a) := \frac{\mathrm{median}(\mathbf{y}_a) - \mathrm{median}(\mathbf{y}_p)}{\mathrm{sq}(\mathbf{y}_a)} \cdot \frac{\mathrm{median}(\mathbf{y}_a) - \mathrm{median}(\mathbf{y}_p)}{\mathrm{sq}(\mathbf{y}_p)} \tag{1}$$

In the given formula, *sq* stands for semi-quartile range, meaning half of the distance between the lower and the upper quartile. Its advantage lies in the fact that it allows for an estimation of the spread without depending on a specific distribution type (e.g. normal). The ETP consists of two components, namely the relative improvement and the spreads. The rationale behind the latter is that algorithms which mostly reach a near optimal value usually exhibit a much smaller spread than ones that often get stuck far away from it. We thus employ distribution properties of the average and the best performance sample to describe how good the improved value is.

## 4 Experimental Investigation

When optimizing a real-world problem, time is usually the most important constraint. But as one often possesses a good estimation of the expected time for each function evaluation, one can easily give the maximum number of evaluations allowed. The concrete value of this number may be specific to every single optimization task, but according to our experience, $1E3$ to $1E4$ is a reasonable size for many applications. We therefore run each algorithm twice, with a budget of $1E3$ and $1E4$ evaluations.

**Table 2.** Median (51 runs) performance of classic algorithms under default settings

| | ALT-BFGS | | BFGS | | CG | | L-BFGS-B | | SANN | | Nelder-Mead | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| no. | 1000 | 10000 | 1000 | 10000 | 1000 | 10000 | 1000 | 10000 | 1000 | 10000 | 1000 | 10000 |
| 6 | 3.82E9 | 393.98 | 390.00 | 390.00 | 390.00 | 390.00 | 390.00 | 390.00 | 1.48E10 | 399.77 | 8.77E5 | 3.67E4 |
| 10 | -173.30 | -260.35 | -144.94 | -254.88 | -133.99 | -216.57 | -143.44 | -235.97 | -178.89 | -192.90 | -23.087 | -113.05 |
| 12 | 3.18E5 | -296.21 | -459.99 | -459.99 | -459.99 | -459.99 | -459.99 | -459.99 | 1670.8 | -113.98 | 2498.9 | -274.72 |
| 13 | 81.314 | -112.10 | -123.64 | -128.01 | -122.34 | -124.68 | -83.996 | -113.10 | -124.78 | -126.65 | -99.094 | -126.16 |
| 16 | 1144.4 | 587.90 | 638.97 | 477.68 | 794.33 | 465.56 | 809.03 | 521.49 | 735.76 | 636.86 | 704.18 | 474.07 |
| 17 | 1124.2 | 745.19 | 1064.2 | 621.97 | 1051.3 | 621.25 | 1624.9 | 1070.5 | 871.62 | 660.13 | 1660.7 | 1097.5 |
| 23 | 1953.9 | 1816.1 | 1948.2 | 1774.1 | 2052.3 | 1785.8 | 2112.6 | 1908.6 | 1658.4 | 1661.3 | 1986.7 | 1724.3 |

Whereas most classic optimization algorithms as BFGS and CG have internal mechanisms to detect when to stop, this is not necessarily true for metaheuristics. As they often do not possess explicit gradient or hessian information of their current search region (the CMA-ES is a notable exception here), defining such a criterion is not trivial. For the generic ES, we thus employ a very simple heuristic and terminates the search if no progress has been made after 10 generations.

In order to enable a fair comparison and use up the maximum number of function evaluations, a simple restart mechanism is applied. It starts the algorithm anew at a random location whenever the budget is not yet exceeded. For all algorithms which terminate autonomously and do not give us access to the code that does so (true for all methods of R optim), we estimate the expected number of function evaluations needed for each single search by averaging the finished searches. Whenever the number of consumed evaluations plus the expected number of evaluations for one search lies below `max evaluations` +10%, the algorithm is given another random start. This mode of operation adds some variation to the results, as the algorithms do not always use *exactly* $1E3$ or $1E4$ evaluations, but some number near to it. However, we expect that this influence is rather small for $1E4$, where often many restarts are done, and unfortunately unavoidable for $1E3$, as long as the algorithms cannot be told from the start how many evaluations to use. As all tested problems are multimodal, it is clear that we cannot expect a single run to detect the global optimum in all cases as it may be trapped in a local one.

*Experiment: How adaptable are the tested optimization algorithms via tuning?*

**Pre-experimental planning.** As a reference for comparison with the adaptable algorithms, a performance table of the considered classical methods is derived, depicted in table 2. All algorithms are run 51 times on each problem and with a budget of $1E3$ and $1E4$ evaluations, respectively. The reported median values indicate that most algorithms cope relatively well with the test problems, with the exception of the Nelder-Mead algorithm, which performs considerably worse. The ALT-BFGS method obviously needs more than $1E3$ evaluations to reach good areas on problems 6 and 12, but with $1E4$ evaluations it also performs well.

According to some preliminary tests, we fixed the parameter intervals for the adaptable methods, given in table 3. Especially for the $C_{cov}$ parameter of the CMA-ES, it was found that values of more than 1 usually led to failure of the algorithm, thus the chosen interval was set to the range from 0 to 1. The restart base parameter of the CMA-ES is a generalization of the restart with increasing population size as introduced in [1]. Each time a restart is scheduled, the population size is not just doubled, but multiplied with this factor. The $X \cdot C_s$ and $X \cdot C_c$ parameters are simply multipliers for the internally precomputed original $C_s$ and $C_c$ values. For the generic EA, we introduce the $p_{reco}$ parameter as probability of recombination. As the treated problems are all multimodal, employing mutation only to realize independent searches may make sense.

**Task.** We cannot demand a concrete adaptability value (ETP) for any algorithm as there is no comparison data on these values available. However, we shall expect that a) if and only if the median and tuned configuration are significantly different (5% random permutation test), than the ETP should have a large value ($> 1$), and b) that there is a negative rank correlation between the p-values of the significance test between median and tuned, and the ETP, when cross-tabled. We require a correlation of at most around $-0.5$ to state that the measure makes sense. Concerning the reviewed adaptable algorithms, the task is to beat the classic methods by changing parameter configurations according to the problem. So at least one of the adatable methods shall be significantly better than the best non-adaptable algorithm (see table 2), detected by a random permutation test at the 5% level. Problems which are solved optimally by the non-adaptable algorithms are excluded here.

**Setup.** For each of the four adaptable algorithms (Nelder-Mead, ALT-BFGS, CMA-ES, and generic-ES) and each run-length ($1E3$ and $1E4$), an SPO run is executed with a budget of 1000 runs, starting from an LHS of size 100 with 4 runs per algorithm. The allowed parameter intervals are given in table 3. By means of eq. 1, we compute the ETP from the median configuration of the LHS and the best configuration obtained via SPO, each validated with 51 separate runs. Additionally, a random permutation test is performed between these two, to check if there is a significant difference in quality stemming from the adaptation (tuning) process. We shall state that for the CMA-ES as well as for the generic-ES, default parameter guidelines are known, and it surely makes sense also to compare with these (resulting in a different ETP definition which we defer to future work). However, here we deliberately take possible misconfigurations into account, as may happen within allowed parameter bounds. Experience shows that for most algorithms and suitable bounds, the median configuration of a large LHS already performs quite well. Algorithms that suffer dramatically from every deviation from the default/best parameters are penalized by this measure.

**Table 3.** Parameter intervals for the tuning process

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Nelder-Mead | alpha | beta | gamma | | | | |
| | 0.1:5 | 0.1:5 | 0.1:5 | | | | |
| ALT-BFGS | initial stepsize | ftol | gtol | | | | |
| | domain·(0.002:1) | $-6$:$-2$ | 0.1:0.99 | | | | |
| self-adaptive EA | initial stepsize | population | selection pressure | $\tau$ | $p_{reco}$ | | |
| | domain·(0.002:1) | 2:100 | 2:10 | 0.01:1 | 0.01:1 | | |
| CMA-ES | initial stepsize | population | damp multiplier | $X \cdot C_s$ | $X \cdot C_c$ | $C_{cov}$ | restart base |
| | domain·(0.002:1) | 2:100 | 0.2:5 | 0.2:2 | 0.2:2 | 0:1 | 1:5 |

**Table 4.** CMA-ES: Median (51 runs) of the median LHS(100) and the best SPO-detected configurations, p-values (random permutation) between both and ETP

| | 1000 | | | | 10000 | | | |
|---|---|---|---|---|---|---|---|---|
| no. | LHS median | SPO best | p-value | ETP | LHS median | SPO best | p-value | ETP |
| 6 | 8.24E7 | 804.06 | 1e-04 | 17731 | 397.38 | 390.00 | 0.0189 | 110.64 |
| 10 | -270.44 | -313.09 | 1e-04 | 100.24 | -325.03 | -328.01 | 1e-04 | 5.9983 |
| 12 | 25386 | 6506.3 | 1e-04 | 8.4837 | 1096.7 | -449.99 | 1e-04 | 1.0539 |
| 13 | -118.74 | -127.41 | 1e-04 | 52.423 | -128.97 | -129.22 | 1e-04 | 1.3046 |
| 16 | 440.26 | 277.85 | 0.2499 | 13.613 | 229.30 | 213.45 | 1e-04 | 10.009 |
| 17 | 502.22 | 335.70 | 1e-04 | 23.990 | 250.80 | 219.94 | 1e-04 | 30.324 |
| 23 | 1601.0 | 1566.6 | 0.0216 | 0.0801 | 1330.5 | 919.56 | 1e-04 | 2.6853 |

**Table 5.** Generic-ES: Median (51 runs) of the median LHS(100) and the best SPO-detected configurations, p-values (random permutation) between both and ETP

| | 1000 | | | | 10000 | | | |
|---|---|---|---|---|---|---|---|---|
| no. | LHS median | SPO best | p-value | ETP | LHS median | SPO best | p-value | ETP |
| 6 | 8.86E9 | 3951.2 | 1e-04 | 1.00E6 | 1537.4 | 483.59 | 1e-04 | 14.870 |
| 10 | -206.99 | -309.82 | 1e-04 | 156.75 | -310.60 | -320.05 | 1e-04 | 19.240 |
| 12 | 93010 | 6891.3 | 1e-04 | 76.331 | 163.58 | -313.40 | 4e-04 | 1.5113 |
| 13 | 4526.5 | -126.97 | 1e-04 | 3598.5 | -128.15 | -129.47 | 1e-04 | 14.928 |
| 16 | 2572.7 | 1667.4 | 1 | 9.1604 | 1023.6 | 426.24 | 1e-04 | 32.024 |
| 17 | 2652.4 | 1887.9 | 1 | 4.6060 | 1344.1 | 482.69 | 1e-04 | 41.054 |
| 23 | 2980.4 | 1931.5 | 1e-04 | 22.579 | 1703.4 | 1440.2 | 1e-04 | 5.4864 |

**Table 6.** Nelder-Mead: Median (51 runs) of the median LHS(100) and the best SPO-detected configurations, p-values (random permutation) between both and ETP

| | 1000 | | | | 10000 | | | |
|---|---|---|---|---|---|---|---|---|
| no. | LHS median | SPO best | p-value | ETP | LHS median | SPO best | p-value | ETP |
| 6 | 3.22E8 | 9547.1 | 1e-04 | 3189.8 | 9.00E8 | 1078.6 | 1e-04 | 9.39E6 |
| 10 | -195.60 | -205.68 | 0.0834 | 0.5810 | -232.89 | -235.91 | 0.0548 | 0.1646 |
| 12 | 53157 | -23.822 | 1e-04 | 270.61 | 26886 | -459.97 | 1e-04 | 3.99E7 |
| 13 | 542.85 | -125.56 | 1e-04 | 320.52 | -38.399 | -128.54 | 1e-04 | 334.50 |
| 16 | 591.41 | 549.66 | 5e-04 | 1.6421 | 477.60 | 400.94 | 1e-04 | 5.8551 |
| 17 | 668.74 | 662.37 | 0.0524 | 0.0118 | 502.12 | 495.95 | 0.0972 | 0.0278 |
| 23 | 1776.8 | 1678.9 | 1e-04 | 10.943 | 1692.0 | 1621.0 | 1e-04 | 15.104 |

**Results/Visualization.** The results of the tuning process are depicted together with the computed ETP in tables 4, 5, 6, and 7. Kendall's tau test between the attained ETP values and the corresponding p-values discriminating between peak and median configuration from the four tables gives a correlation estimate of -0.554 with a p-value of 5.2E-8. Note that the discriminating power of the ETP is higher than the p-value as it returns intermediate values where the p-value is either near zero or 1, see e.g. table 5. Concerning the comparison of the best unparametrized method with the best adaptable method, we state that there is no need for a better method for problems 6 and 12 ($1E3$ and $1E4$ evaluations); these are solved to optimality by BFGS, CG, and L-BFGS-B. On problem 10, we have to compare SANN against the CMA-ES ($1E3$) and ALT-BFGS against the CMA-ES ($1E4$). For problem 13, it is SANN vs. CMA-ES ($1E3$) and BFGS vs. the generic ES ($1E4$). Problem 16 is dominated by BFGS and the CMA-ES ($1E3$), and CG and the CMA-ES ($1E4$). On problem 17, SANN and the CMA-ES perform best ($1E3$), and CG and the CMA-ES ($1E4$). Finally, for problem

**Table 7.** ALT-BFGS: Median (51 runs) of the median LHS(100) and the best SPO-detected configurations, p-values (random permutation) between both and ETP

| | 1000 | | | | 10000 | | | |
|---|---|---|---|---|---|---|---|---|
| no. | LHS median | SPO best | p-value | ETP | LHS median | SPO best | p-value | ETP |
| 6 | 4.44E10 | 4.82E7 | 1e-04 | 153.11 | 8.80E8 | 390.00 | 1e-04 | 9.52E15 |
| 10 | -162.85 | -175.78 | 0.1385 | 0.1093 | -247.41 | -259.35 | 0.1624 | 0.6133 |
| 12 | 3.30E6 | 1233.6 | 1e-04 | 27.228 | -163.85 | -459.99 | 1e-04 | 1.22E11 |
| 13 | 136.59 | 32.148 | 0.3409 | 0.3152 | -102.06 | -122.95 | 0.0019 | 0.5457 |
| 16 | 1026.6 | 1090.2 | 0.7604 | 0.0931 | 610.37 | 626.33 | 0.6361 | 0.0377 |
| 17 | 1110.5 | 1090.2 | 0.5835 | 0.0142 | 762.43 | 708.35 | 0.1577 | 0.5200 |
| 23 | 1911.0 | 1921.6 | 0.3728 | 0.0291 | 1810.8 | 1813.7 | 0.4411 | 0.0066 |

23, we compare SANN against the CMA-ES ($1E3$ and $1E4$). In all cases, the adaptable algorithm was significantly better than the unparametrized one, with random permuation test p-values of below $1.0E-4$. Only on problem 23 ($1E3$), the p-value was larger (0.0042), still indicating a significant difference.

**Observations.** We observe that for most algorithms, tuning works out well, so that the performance increases significantly. However, the ALT-BFGS seems to be hard to tune as indicated by best configurations in the range of median configurations. It is also striking that SANN performs remarkably well, especially on the more rugged landscapes on problems 13, 17, and 23. However, for each algorithm, the measured tuning potential is quite different for the considered problems: E.g., for Nelder-Mead, there is almost no potential on problems 10 and 17, but there is for the others. The two evolutionary algorithms (CMA-ES and generic ES) behave very similar, with the CMA-ES usually winning.

**Discussion.** First of all, the ETP measure seems to work well. It may not be ideal, but at least fulfills the criteria stated as tasks. The tuning potential itself is quite different for the tested algorithm/problem combinations. Sometimes an algorithm is brought from complete failure to robustly finding the optimum, as for the ALT-BFGS on problem 6 with $1E4$ evaluations. The opposite also happens, namely that no better solution is found by the tuning process. This also happens for the ALT-BFGS several times, so it can be assumed that this algorithm is a bit more 'difficult to adapt' than the others. The 2 evolutionary algorithms seem to have the highest tuning potential, followed by Nelder-Mead, and then ALG-BFGS. Note that for technical reasons, it is also possible that the LHS result is seemingly better than the SPO one. The reason is that the LHS table is built with only four runs per configuration, which is seemingly not enough to estimate its quality in some cases. So the LHS configuration may in fact be much better or worse than it first appeared.

## 5 Conclusions

This work documents several insights. The featured tuning potential measure (ETP) seems to work well, but may still be improved. For instance, the time aspect is not included at all yet. Concerning the comparison of adaptable and non-adaptable algorithms we can state that the potential of the adaptable algorithms is used by tuning if the problems are not already solved to optimality by the non-adaptable algorithms. Thus, it is a question of time that is needed to adapt a better method. However, the comparison on the CEC test set may be not entirely fair, as the classic methods have not been designed for such problems. Nevertheless, they cope quite well and are only overtaken by well adapted algorithms and/or on the most difficult problems.

## References

1. Auger, A., Hansen, N.: A Restart CMA Evolution Strategy With Increasing Population Size. In: McKay, B., et al. (eds.) Proc. 2005 Congress on Evolutionary Computation (CEC 2005), Piscataway NJ, pp. 1769–1776. IEEE Press, Los Alamitos (2005)
2. Bartz-Beielstein, T.: Experimental Research in Evolutionary Computation – The New Experimentalism. Natural Computing Series. Springer, Berlin (2006)
3. Bartz-Beielstein, T., Preuss, M.: Considerations of Budget Allocation for Sequential Parameter Optimization (SPO). In: Paquete, L., Chiarandini, M., Basso, D. (eds.) Empirical Methods for the Analysis of Algorithms, Workshop EMAA 2006, Proceedings, Reykjavik, Iceland, pp. 35–40 (2006)
4. Belisle, C.J.P.: Convergence theorems for a class of simulated annealing algorithms on $\mathbb{R}^d$. Annals of Applied Probability 29, 885–895 (1992)
5. Beyer, H.-G., Schwefel, H.-P.: Evolution strategies: A comprehensive introduction. Natural Computing 1(1), 3–52 (2002)
6. Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K.: A racing algorithm for configuring metaheuristics. In: Langdon, W.B., et al. (eds.) Proc. Genetic and Evolutionary Computation Conf. (GECCO 2002). Morgan Kaufmann, San Francisco (2002)
7. Byrd, R.H., Lu, P., Nocedal, J., Zhu, C.: A limited memory algorithm for bound constrained optimization. SIAM Journal on Scientific Computing 16, 1190–1208 (1995)
8. Clausen, A.: An implementation of the bfgs algorithm for smooth function minimization (2007), `http://www.econ.upenn.edu/~clausen/computing/bfgs.zip`
9. Eiben, A.E., Jelasity, M.: A critical note on experimental research methodology in EC. In: Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002), pp. 582–587. IEEE Press, Los Alamitos (2002)
10. Hansen, N., Ostermeier, A.: Completely Derandomized Self-Adaptation in Evolution Strategies. IEEE Computational Intelligence Magazine 9(2), 159–195 (2001)
11. Jong, K.D.: Parameter setting in eas: a 30 year perspective. In: Lobo, F.G., Lima, C.F., Michalewicz, Z. (eds.) Parameter Setting in Evolutionary Algorithms. Springer, Berlin (2007)
12. Nash, J.C.: Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation, 2nd edn. IOP, Bristol (1990)
13. Nelder, J.A., Mead: A simplex method for function minimization. The Computer Journal (7), 308–313 (1965)
14. Schwefel, H.-P.: Direct search for optimal parameters within simulation models. In: Conine, R.D., et al. (eds.) Proc. Twelfth Annual Simulation Symp., Tampa FL, Long Beach, CA, pp. 91–102. IEEE Computer Society, Los Alamitos (1979)
15. Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.-P., Auger, A., Tiwari, S.: Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Technical report, Nanyang Technological University, Singapore (May 2005), `http://www.ntu.edu.sg/home/EPNSugan`