A Memetic Algorithm using Local Search Chaining for Black-Box Optimization Benchmarking 2009 for Noisy Functions

Daniel Molina
Dept. of Computer Languages and Systems
11002 University of Cádiz
Cádiz, Spain
daniel.molina@uca.es

Manuel Lozano
Dept. of Computer Science and AI
18071 University of Granada
Granada, Spain
lozano@decsai.ugr.es

Francisco Herrera
Dept. of Computer Science and AI
18071 University of Granada
Granada, Spain
herrera@decsai.ugr.es

ABSTRACT

Memetic algorithms with continuous local search methods have arisen as effective tools to address the difficulty of obtaining reliable solutions of high precision for complex continuous optimisation problems. There exists a group of continuous search algorithms that stand out as brilliant local search optimisers. Several of them, like CMA-ES, often require a high number of evaluations to adapt its parameters. Unfortunately, this feature makes difficult to use them to create memetic algorithms.

In this work, we show a memetic algorithm that applies CMA-ES to refine the solutions, assigning to each individual a local search intensity that depends on its features, by chaining different local search applications.

Experiments are carried out on the noisy Black-Box Optimization Benchmarking BBOB'2009 test suite.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: OptimizationGlobal Optimization, Unconstrained Optimization; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems

General Terms

Algorithms

Keywords

Benchmarking, Black-box optimization, Evolutionary computation, Memetic Algorithms, Hybrid Metaheuristics

GECCO'09, July 8–12, 2009, Montréal Québec, Canada.

1. INTRODUCTION

It is well known that the hybridisation of *evolutionary algorithms* (EAs) with other techniques can greatly improve the search efficiency [3, 4]. EAs that have been hybridised with local search (LS) techniques are often called *memetic algorithms* (MAs) [17, 18, 15]. One commonly MA scheme improves the new created solutions using an LS method, with the aim of exploiting the best search regions gathered during the global sampling done by the EA. This allows us to design MAs for continuous optimization (MACO) that obtain high accurate solutions for these problems [12, 21, 13].

Nowadays, there are powerful metaheuristics available as LS algorithms that can achieve very good results by the adaptation of several explicit *strategy parameters* to guide the search. This adaptation allows them to increase the likelihood of producing more effective solutions, at the cost of requiring a substantial number of evaluations (high LS intensity). We call *intense* continuous LS algorithms to this kind of LS procedures.

In particular, we consider the *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES) model [11]. It can be seen as an intense continuous LS algorithm because, as noted by [2]: *"CMA-ES may require a substantial number of time steps for the adaptation"*.

Since CMA-ES is extremely good at detecting and exploiting local structure, it turns out to be a particularly reliable and highly competitive EA for local optimisation. In fact, very powerful *multi-start LS metaheuristics* have been created using this algorithm[19, 6]. Unfortunately, as the other intense continuous LS methods, the requirement of a high intensity make not easy to create MACO using it.

In this work, we show a MACO model, MA with LS Chains (MA-LS-Chain) [16] that employs the concept of *LS chain* to adjust the LS intensity, assigning to each individual a local search intensity that depends on its features, by chaining different local search applications. In our model, an individual resulting from an LS invocation may later become initial point of a subsequent LS application, which will adopt the final strategy parameter values achieved by the former as its initial ones. In this way, the continuous LS method may *adaptively* fit its strategy parameters to the particular features of the search zones, increasing the LS effort over the most promising solutions, and regions.

The paper is set up as follows. In Section 2, we describe in detail the presented algorithm. In Section 3, we present the experimental section, using the Black-Box Optimization Benchmarking for Noisy Function (BBOB'2009). Finally, in Section 4, we provide several conclusions.

2. MA-LS-CHAIN: MACO THAT HANDLES LS CHAINS

In this section, we describe MA-LS-Chain, MACO approach proposed in [16] that employs the concept of *LS chain* to adjust the LS intensity assigned to the intense continuous LS method. In particular, this MACO handles LS chains, throughout the evolution, with the objective of allowing the continuous LS algorithm to act more intensely in the most promising areas represented in the EA population. In this way, the LS method may adaptively fit its strategy parameters to the particular features of these zones.

2.1 Steady-State MAs

In steady-state GAs [20] usually only or two offspring are produced in each generation. Parents are selected to produce offspring and then a decision is made to select which individuals in the population will be deleted in order to make room to new offspring. Steady-state GAs are *overlapping* systems because parents and offspring compete for survival. A widely used replacement strategy is to replace the worst individual only if the new individual is better. We will call this strategy the *standard replacement strategy*.

Although steady-state GAs are less common than generational GAs, Land [14] recommended their use for the design of *steady-state MAs* (steady-state GAs plus LS method) because they may be more stable (as the best solutions do not get replaced until the newly generated solutions become superior) and they allow the results of LS to be maintained in the population.

2.2 CMA-ES Algorithm

The *covariance matrix adaptation evolution strategy* (CMA-ES) [11, 10] is an optimization algorithm originally introduced to improve the LS performances of evolution strategies. Even though CMA-ES even reveals competitive global search performances [9], it has exhibited effective abilities for the local tuning of solutions. At the *2005 Congress of Evolutionary Computation*, a *multi-start LS metaheuristic* using this method [1] was one of the winners of the real-parameter optimisation competition [19, 6]. Thus, investigating the behaviour of CMA-ES as LS component for MACOs deserves much attention.

Any evolution strategy that uses intermediate recombination can be interpreted as an LS strategy [11]. Thus, since CMA-ES is extremely good at detecting and exploiting local structure, it turns out to be a particularly reliable and highly competitive EA for local optimisation [1].

In CMA-ES, both the step size and the step direction,defined by a covariance matrix, are adjusted at each generation. To do that, this algorithm generates a population of λ offspring by sampling a multivariate normal distribution:

$$x_i \sim N\left(m, \sigma^2 C\right) = m + \sigma N_i(0, C) \text{ for } i = 1, \cdots, \lambda,$$

where the mean vector m represents the favourite solution at present, the so-called step-size σ controls the step length, and the covariance matrix C determines the shape of the distribution ellipsoid. Then, the μ best offspring are used to recalculate the mean vector, σ and m and the covariance matrix C, following equations that may be found in [11] and [9]. The default strategy parameters are given in [9]. Only the initial m and σ parameters have to be set depending on the problem.

In this paper, the initial m value is the initial solution, and we consider the initial σ as the half of distance to the most close solution. We have applied the code available in `http://www.lri.fr/~hansen/cmaes_inmatlab.html`.

2.3 Local Search Chains

In steady-state MAs, individuals improved by the LS invocations may reside in the population during a long time. This circumstance allows these individuals to become starting points of subsequent LS invocations. In [16], Molina et al. propose to *chain* an LS algorithm invocation and the next one as follows:

> *The final configuration reached by the former (strategy parameter values, internal variables, etc.) is used as initial configuration for the next application.*

In this way, the LS algorithm may continue under the same conditions achieved when the LS operation was halted, providing an *uninterrupted connection between successive LS invocations*, i.e., forming a *LS chain*.

Two important aspects that were taken into account for the management of LS chains are:

- Every time the LS algorithm is applied to refine a particular chromosome, a fixed LS intensity should be considered for it, which will be called *LS intensity stretch* (I_{str}).

  In this way, an LS chain formed throughout n_{app} LS applications and started from solution s_0 will return the same solution as the application of the continuous LS algorithm to s_0 employing $n_{app} \cdot I_{str}$ fitness function evaluations.

- After the LS operation, the parameters that define the current state of the LS processing are stored along with the final individual reached (in the steady-state GA population). When this individual is selected to be improved, the initial values for the parameters of the LS algorithm will be directly available.

2.4 MA-LS-Chain : A MACO that Handles LS Chains

MA-LS-Chain [16] is a MACO model that handles the LS chains (see Figure 1), with the following main features:

1. It is a steady-state MA model.
2. It ensures that a fixed and predetermined local/global search ratio, $r_{L/G}$, is always kept. $r_{L/G}$ is defined as the percentage of evaluations spent doing LS from the total assigned to the algorithm's run. With this policy, we easily stabilise this ratio, which has a strong influence on the final MACO behavior. Without this strategy, the application of intense continuous LS algorithms may induce the MACO to prefer super exploitation.

3. It favours the enlargement of those LS chains that are showing promising fitness improvements in the best current search areas represented in the steady-state GA population. In addition, it encourages the activation of innovative LS chains with the aim of refining unexploited zones, whenever the current best ones may not offer profitability. The criterion to choose the individuals that should undergo LS is specifically designed to manage the LS chains in this way (Steps 3 and 4).

1. Generate the `initial population`.
2. Perform the `steady-state GA` throughout n_{frec} evaluations.
3. Build the set S_{LS} with those individuals that `potentially may be refined by LS`.
4. Pick `the best individual` in S_{LS} (Let's c_{LS} to be this individual).
5. if c_{LS} belongs to an `existing LS chain` then
6. Initialise the LS operator with the `LS state stored` together with c_{LS}.
7. else
8. Initialise the LS operator with the `default` LS state.
9. Apply the LS algorithm to c_{LS} with an LS intensity of I_{str} (Let's c^r_{LS} to be the resulting individual).
10. Replace c_{LS} by c^r_{LS} in the `steady-state GA population`.
11. Store the `final LS state` along with c^r_{LS}.
12. If (*not termination-condition*) go to step 2.

Figure 1: Pseudocode algorithm for MA-LS-Chain

MA-LS-Chain defines the following relation between the steady-state GA and the LS method (Step 2): *every n_{frec} number of evaluations of the steady-state GA, apply the continuous LS algorithm to a selected chromosome, c_{LS}, in the steady-state GA population.*

$$n_{frec} = I_{str}\frac{1 - r_{L/G}}{r_{L/G}} \quad (1)$$

Since we assume a fixed $\frac{L}{G}$ ratio, $r_{L/G}$, n_{frec} must be automatically calculated. The n_{frec} value is obtained by Equation 1, where n_{str} is the LS intensity stretch (Section 2.3).

The following mechanism is performed to select c_{LS} (Steps 3 and 4):

1. Build the set of individuals in the steady-state GA population, S_{LS} that fulfils:
   (a) They have never been optimised by the LS algorithm, or
   (b) They previously underwent LS, obtaining a fitness function improvement greater than δ^{min}_{LS} (a parameter of our algorithm).
2. If $|S_{LS}| \neq 0$, then apply the continuous LS algorithm to the best individual in this set. On other case, the population of the steady-state MA is restarted randomly (keeping the best individual).

With this mechanism, when the steady-state GA finds a new best individual, it will be refined immediately. Furthermore, the best performing individual in the steady-state GA population will always undergo LS whenever the fitness improvement obtained by a previous LS application to this individual is greater than a δ^{min}_{LS} threshold. In our experiments, δ^{min}_{LS} is the threshold value, 10^{-8}.

3. RESULTS

Results from experiments according to [7] on the benchmarks functions given in [5, 8] are presented in Figures 2 and 3 and in Tables 1 and 2.

4. CONCLUSIONS

We have presented an algorithm, MA-LS-Chain that applies the CMA-ES algorithm to refine solutions, creating *LS Chains*. In MA-LS-Chain, the same individual can be improved several times by CMA-ES. An individual resulting from an LS invocation may later become initial point of a subsequent LS application, continuing the local search. Using this idea, MA-LS-Chain focus the LS search around the most promising solutions, and regions. Experiments have been carried out on the noisy BBOB 2009 test.

5. ACKNOWLEDGMENTS

This work was supported by the Research Project TIN2008-05854 and P08-TIC-4173.

`\thebibliography`

6. REFERENCES

[1] A. Auger and N. Hansen. Performance Evaluation of an Advanced Local Search Evolutionary Algorithm. In *2005 IEEE Congress on Evolutionary Computation*, pages 1777–1784, 2005.

[2] A. Auger, M. Schoenauer, and N. Vanhaecke. LS-CMAES: a second-order algorithm for covariance matrix adaptation. In *Proc. of the Parallel problems solving for Nature - PPSN VIII, Sept. 2004, Birmingham*, 2004.

[3] L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[4] W. B. et al., editor. *Optimizing global-local search hybrids*. Morgan Kaufmann, San Mateo, California, 1999.

[5] S. Finck, N. Hansen, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Presentation of the noisy functions. Technical Report 2009/21, Research Center PPE, 2009.

[6] N. Hansen. Compilation of Results on the CEC Benchmark Function Set. In *2005 IEEE Congress on Evolutionary Computation*, 2005.

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

[8] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Noisy functions definitions. Technical Report RR-6869, INRIA, 2009.

[9] N. Hansen and S. Kern. Evaluating the CMA Evolution Strategy on Multimodal Test Functions. In

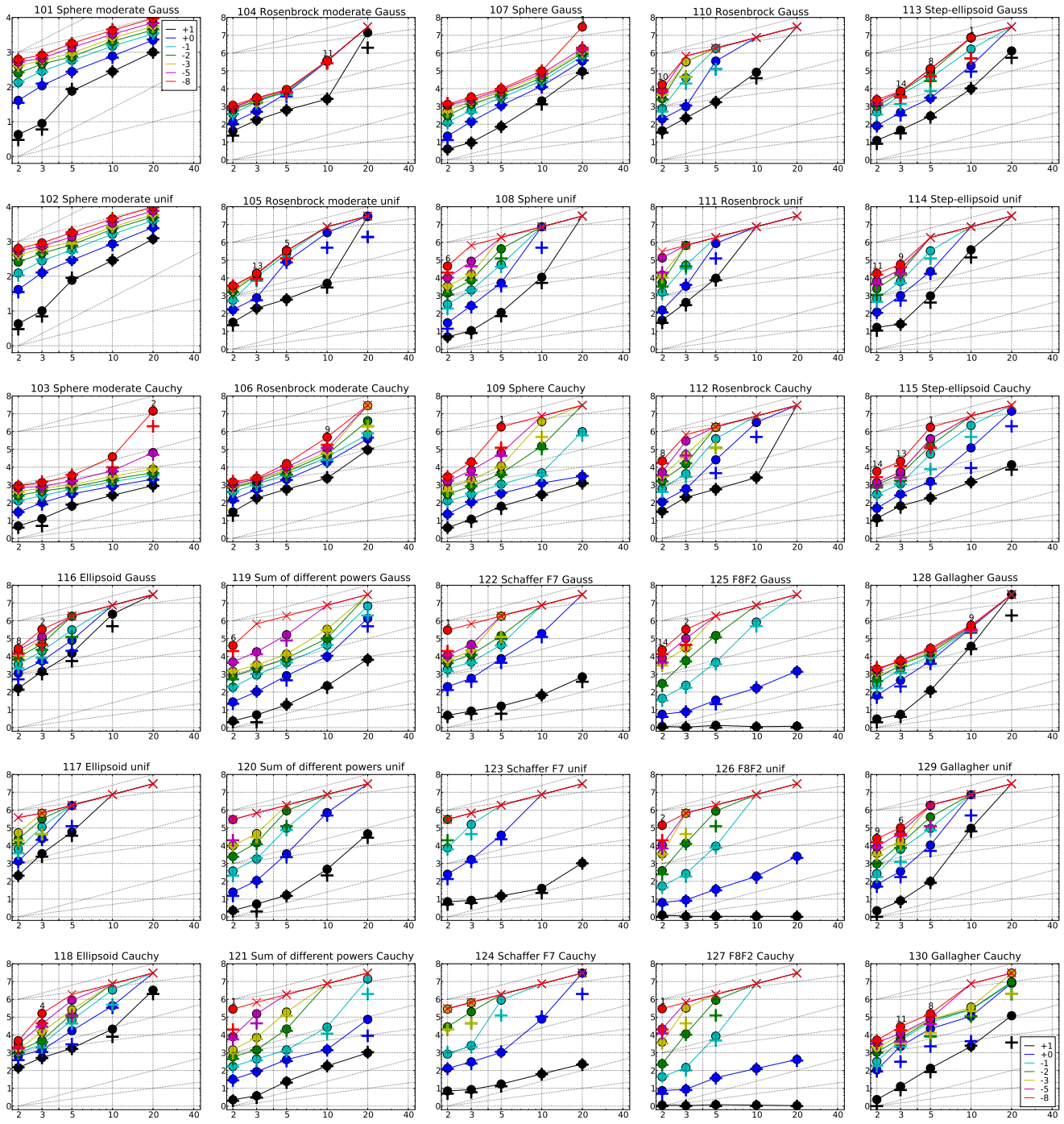

Figure 2: Expected Running Time (ERT, ●) **to reach** $f_{\mathrm{opt}} + \Delta f$ **and median number of function evaluations of successful trials** (+), **shown for** $\Delta f = 10, 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-5}, 10^{-8}$ **(the exponent is given in the legend of** f_{101} **and** f_{130}**) versus dimension in log-log presentation. The** $\mathrm{ERT}(\Delta f)$ **equals to** $\#\mathrm{FEs}(\Delta f)$ **divided by the number of successful trials, where a trial is successful if** $f_{\mathrm{opt}} + \Delta f$ **was surpassed during the trial. The** $\#\mathrm{FEs}(\Delta f)$ **are the total number of function evaluations while** $f_{\mathrm{opt}} + \Delta f$ **was not surpassed during the trial from all respective trials (successful and unsuccessful), and** f_{opt} **denotes the optimal function value. Crosses** (×) **indicate the total number of function evaluations** $\#\mathrm{FEs}(-\infty)$**. Numbers above ERT-symbols indicate the number of successful trials. Annotated numbers on the ordinate are decimal logarithms. Additional grid lines show linear and quadratic scaling.**

Δf	f_{101} in 5-D,N=15,mFE=1977 #	ERT	10%	90%	RT_{succ}	f_{101} in 20-D, N=15, mFE=10626 #	ERT	10%	90%	RT_{succ}
10	15	7.6e1	6.3e1	9.0e1	7.6e1	15	1.0e3	9.2e2	1.1e3	1.0e3
1	15	2.8e2	2.6e2	2.9e2	2.8e2	15	2.2e3	2.1e3	2.4e3	2.2e3
1e−1	15	5.8e2	5.6e2	6.2e2	5.8e2	15	3.4e3	3.3e3	3.5e3	3.4e3
1e−3	15	9.4e2	8.5e2	1.0e3	9.4e2	15	5.5e3	5.4e3	5.6e3	5.5e3
1e−5	15	1.4e3	1.2e3	1.5e3	1.4e3	15	7.1e3	7.0e3	7.2e3	7.1e3
1e−8	15	1.8e3	1.8e3	1.8e3	1.8e3	15	9.3e3	9.2e3	9.5e3	9.3e3

Δf	f_{102} in 5-D,N=15,mFE=1954 #	ERT	10%	90%	RT_{succ}	f_{102} in 20-D, N=15, mFE=11485 #	ERT	10%	90%	RT_{succ}
10	15	7.7e1	6.2e1	9.0e1	7.7e1	15	1.2e3	1.1e3	1.2e3	1.2e3
1	15	2.9e2	2.6e2	3.2e2	2.9e2	15	2.5e3	2.4e3	2.7e3	2.5e3
1e−1	15	5.8e2	5.1e2	6.4e2	5.8e2	15	4.0e3	3.9e3	4.2e3	4.0e3
1e−3	15	9.7e2	8.6e2	1.1e3	9.7e2	15	5.9e3	5.8e3	6.1e3	5.9e3
1e−5	15	1.3e3	1.2e3	1.5e3	1.3e3	15	7.6e3	7.4e3	7.8e3	7.6e3
1e−8	15	1.8e3	1.8e3	1.8e3	1.8e3	15	9.8e3	9.5e3	1.0e4	9.8e3

Δf	f_{103} in 5-D,N=15,mFE=5982 #	ERT	10%	90%	RT_{succ}	f_{103} in 20-D, N=15, mFE=2000000 #	ERT	10%	90%	RT_{succ}
10	15	6.6e1	5.1e1	8.1e1	6.6e1	15	9.3e2	8.7e2	9.9e2	9.3e2
1	15	3.1e2	3.0e2	3.3e2	3.1e2	15	2.1e3	2.0e3	2.2e3	2.1e3
1e−1	15	5.7e2	5.4e2	6.0e2	5.7e2	15	3.6e3	3.5e3	3.7e3	3.6e3
1e−3	15	1.0e3	9.5e2	1.1e3	1.0e3	15	7.9e3	7.4e3	8.3e3	7.9e3
1e−5	15	1.7e3	1.6e3	1.9e3	1.7e3	15	6.6e4	4.4e4	8.8e4	6.6e4
1e−8	15	3.3e3	2.9e3	3.7e3	3.3e3	2	1.4e7	1.4e7	1.5e7	2.0e6

Δf	f_{104} in 5-D,N=15,mFE=17794 #	ERT	10%	90%	RT_{succ}	f_{104} in 20-D, N=15, mFE=2000000 #	ERT	10%	90%	RT_{succ}
10	15	6.3e2	6.0e2	6.6e2	6.3e2	2	1.4e7	1.3e7	1.5e7	1.3e6
1	15	5.5e3	4.2e3	6.8e3	5.5e3	0	*14e+0*	*82e−1*	*15e+0*	1.8e6
1e−1	15	6.6e3	5.3e3	8.0e3	6.6e3	.	.	.	.	.
1e−3	15	7.8e3	6.6e3	9.0e3	7.8e3	.	.	.	.	.
1e−5	15	8.4e3	7.2e3	9.8e3	8.4e3	.	.	.	.	.
1e−8	15	8.9e3	7.7e3	1.0e4	8.9e3	.	.	.	.	.

Δf	f_{105} in 5-D,N=15,mFE=125000 #	ERT	10%	90%	RT_{succ}	f_{105} in 20-D, N=15, mFE=2000000 #	ERT	10%	90%	RT_{succ}
10	15	6.4e2	5.9e2	6.9e2	6.4e2	1	2.8e7	2.6e7	3.0e7	2.0e6
1	12	8.4e4	6.7e4	1.0e5	6.6e4	1	3.0e7	3.0e7	3.0e7	2.0e6
1e−1	7	2.3e5	2.1e5	2.5e5	1.1e5	0	*15e+0*	*12e+0*	*17e+0*	1.3e6
1e−3	5	3.3e5	3.1e5	3.6e5	1.1e5	.	.	.	.	.
1e−5	5	3.3e5	3.1e5	3.6e5	1.1e5	.	.	.	.	.
1e−8	5	3.3e5	3.1e5	3.6e5	1.1e5	.	.	.	.	.

Δf	f_{106} in 5-D,N=15,mFE=35488 #	ERT	10%	90%	RT_{succ}	f_{106} in 20-D, N=15, mFE=2000000 #	ERT	10%	90%	RT_{succ}
10	15	5.8e2	5.7e2	5.9e2	5.8e2	15	9.4e4	8.1e4	1.1e5	9.4e4
1	15	2.1e3	1.8e3	2.3e3	2.1e3	15	3.9e5	3.2e5	4.6e5	3.9e5
1e−1	15	3.9e3	3.3e3	4.6e3	3.9e3	15	6.8e5	5.8e5	7.9e5	6.8e5
1e−3	15	7.4e3	6.3e3	8.7e3	7.4e3	1	2.9e7	2.8e7	3.0e7	2.0e6
1e−5	15	1.0e4	8.3e3	1.2e4	1.0e4	0	*15e−3*	*11e−4*	*59e−3*	7.9e5
1e−8	15	1.6e4	1.3e4	1.8e4	1.6e4	.	.	.	.	.

Δf	f_{107} in 5-D,N=15,mFE=12747 #	ERT	10%	90%	RT_{succ}	f_{107} in 20-D, N=15, mFE=2000000 #	ERT	10%	90%	RT_{succ}
10	15	7.6e1	5.7e1	9.6e1	7.6e1	15	9.2e4	7.8e4	1.1e5	9.2e4
1	15	1.1e3	9.6e2	1.3e3	1.1e3	15	3.6e5	3.2e5	4.0e5	3.6e5
1e−1	15	3.1e3	2.9e3	3.3e3	3.1e3	15	6.7e5	6.2e5	7.3e5	6.7e5
1e−3	15	6.5e3	6.3e3	6.8e3	6.5e3	15	1.2e6	1.1e6	1.3e6	1.2e6
1e−5	15	8.3e3	7.9e3	8.8e3	8.3e3	13	1.8e6	1.7e6	1.9e6	1.5e6
1e−8	15	9.8e3	9.2e3	1.0e4	9.8e3	1	3.0e7	2.9e7	3.0e7	2.0e6

Δf	f_{108} in 5-D,N=15,mFE=125000 #	ERT	10%	90%	RT_{succ}	f_{108} in 20-D, N=15, mFE=2000000 #	ERT	10%	90%	RT_{succ}
10	15	1.1e2	8.6e1	1.4e2	1.1e2	0	*21e+0*	*15e+0*	*26e+0*	1.0e6
1	15	5.2e3	3.6e3	6.8e3	5.2e3	.	.	.	.	.
1e−1	14	5.6e4	4.4e4	6.8e4	5.2e4	.	.	.	.	.
1e−3	0	*29e−3*	*44e−4*	*81e−3*	1.0e5	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	f_{109} in 5-D,N=15,mFE=125000 #	ERT	10%	90%	RT_{succ}	f_{109} in 20-D, N=15, mFE=2000000 #	ERT	10%	90%	RT_{succ}
10	15	6.3e1	4.6e1	8.0e1	6.3e1	15	1.3e3	1.2e3	1.4e3	1.3e3
1	15	3.3e2	3.0e2	3.6e2	3.3e2	15	3.1e3	2.9e3	3.3e3	3.1e3
1e−1	15	1.1e3	1.0e3	1.2e3	1.1e3	13	9.8e5	7.3e5	1.3e6	8.0e5
1e−3	15	1.2e4	8.9e3	1.4e4	1.2e4	0	*54e−3*	*32e−3*	*10e−2*	1.3e6
1e−5	15	6.2e4	5.1e4	7.3e4	6.2e4	.	.	.	.	.
1e−8	1	1.8e6	1.8e6	1.9e6	1.2e5	.	.	.	.	.

Δf	f_{110} in 5-D,N=15,mFE=125000 #	ERT	10%	90%	RT_{succ}	f_{110} in 20-D, N=15, mFE=2000000 #	ERT	10%	90%	RT_{succ}
10	15	1.8e3	1.6e3	2.0e3	1.8e3	0	*28e+0*	*24e+0*	*31e+0*	1.6e6
1	4	3.5e5	3.0e5	4.1e5	9.6e4	.	.	.	.	.
1e−1	1	1.8e6	1.8e6	1.9e6	1.2e5	.	.	.	.	.
1e−3	0	*15e−1*	*11e−2*	*21e−1*	5.6e4	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	f_{111} in 5-D,N=15,mFE=125000 #	ERT	10%	90%	RT_{succ}	f_{111} in 20-D, N=15, mFE=2000000 #	ERT	10%	90%	RT_{succ}
10	15	9.6e3	7.6e3	1.2e4	9.6e3	0	*17e+2*	*11e+2*	*24e+2*	1.4e6
1	2	8.6e5	7.8e5	9.4e5	7.4e4	.	.	.	.	.
1e−1	0	*19e−1*	*98e−2*	*33e−1*	7.1e4	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	f_{112} in 5-D,N=15,mFE=125000 #	ERT	10%	90%	RT_{succ}	f_{112} in 20-D, N=15, mFE=2000000 #	ERT	10%	90%	RT_{succ}
10	15	5.8e2	5.6e2	6.0e2	5.8e2	0	*16e+0*	*12e+0*	*17e+0*	7.9e5
1	15	2.6e4	1.3e4	3.8e4	2.6e4	.	.	.	.	.
1e−1	4	3.9e5	3.4e5	4.4e5	1.0e5	.	.	.	.	.
1e−3	1	1.8e6	1.6e6	1.9e6	1.2e5	.	.	.	.	.
1e−5	0	*28e−2*	*13e−3*	*68e−2*	8.9e4	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	f_{113} in 5-D,N=15,mFE=125000 #	ERT	10%	90%	RT_{succ}	f_{113} in 20-D, N=15, mFE=2000000 #	ERT	10%	90%	RT_{succ}
10	15	2.9e2	2.1e2	3.8e2	2.9e2	11	1.3e6	9.6e5	1.7e6	1.0e6
1	15	2.9e3	2.5e3	3.3e3	2.9e3	0	*62e−1*	*29e−1*	*17e+0*	1.1e6
1e−1	13	4.0e4	2.2e4	5.9e4	3.0e4	.	.	.	.	.
1e−3	8	1.3e5	9.7e4	1.7e5	7.0e4	.	.	.	.	.
1e−5	8	1.3e5	9.7e4	1.6e5	7.0e4	.	.	.	.	.
1e−8	8	1.3e5	1.0e5	1.7e5	7.0e4	.	.	.	.	.

Δf	f_{114} in 5-D,N=15,mFE=125000 #	ERT	10%	90%	RT_{succ}	f_{114} in 20-D, N=15, mFE=2000000 #	ERT	10%	90%	RT_{succ}
10	15	9.7e2	6.0e2	1.4e3	9.7e2	0	*67e+0*	*57e+0*	*79e+0*	1.0e6
1	15	2.3e4	1.6e4	3.0e4	2.3e4	.	.	.	.	.
1e−1	5	3.3e5	3.1e5	3.5e5	1.1e5	.	.	.	.	.
1e−3	0	*29e−2*	*47e−3*	*69e−2*	7.9e4	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	f_{115} in 5-D,N=15,mFE=125000 #	ERT	10%	90%	RT_{succ}	f_{115} in 20-D, N=15, mFE=2000000 #	ERT	10%	90%	RT_{succ}
10	15	1.9e2	1.6e2	2.2e2	1.9e2	15	1.4e4	6.4e3	2.1e4	1.4e4
1	15	1.6e3	1.1e3	2.1e3	1.6e3	2	1.4e7	1.3e7	1.5e7	1.6e6
1e−1	12	5.5e4	3.5e4	7.7e4	3.2e4	0	*15e−1*	*99e−2*	*33e−1*	1.0e6
1e−3	4	3.9e5	3.4e5	4.4e5	1.2e5	.	.	.	.	.
1e−5	4	3.9e5	3.3e5	4.4e5	1.2e5	.	.	.	.	.
1e−8	1	1.8e6	1.6e6	1.9e6	1.2e5	.	.	.	.	.

Δf	f_{116} in 5-D,N=15,mFE=125000 #	ERT	10%	90%	RT_{succ}	f_{116} in 20-D, N=15, mFE=2000000 #	ERT	10%	90%	RT_{succ}
10	14	1.5e4	5.7e3	2.5e4	1.4e4	0	*57e+1*	*39e+1*	*11e+2*	1.6e6
1	10	8.0e4	5.6e4	1.1e5	5.1e4	.	.	.	.	.
1e−1	5	3.1e5	2.6e5	3.4e5	1.2e5	.	.	.	.	.
1e−3	0	*37e−2*	*14e−3*	*38e−1*	7.1e4	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	f_{117} in 5-D,N=15,mFE=125000 #	ERT	10%	90%	RT_{succ}	f_{117} in 20-D, N=15, mFE=2000000 #	ERT	10%	90%	RT_{succ}
10	13	5.8e4	4.4e4	7.2e4	5.3e4	0	*36e+2*	*26e+2*	*59e+2*	8.9e5
1	1	1.8e6	1.7e6	1.9e6	1.2e5	.	.	.	.	.
1e−1	0	*27e−1*	*12e−1*	*12e+0*	8.9e4	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Δf	f_{118} in 5-D,N=15,mFE=125000 #	ERT	10%	90%	RT_{succ}	f_{118} in 20-D, N=15, mFE=2000000 #	ERT	10%	90%	RT_{succ}
10	15	1.8e3	1.5e3	2.1e3	1.8e3	7	3.3e6	2.9e6	3.7e6	1.2e6
1	15	1.7e4	6.0e3	2.9e4	1.7e4	0	*10e+0*	*42e−1*	*29e+0*	1.0e6
1e−1	13	6.8e4	5.1e4	8.7e4	5.5e4	.	.	.	.	.
1e−3	6	2.6e5	2.3e5	2.9e5	9.8e4	.	.	.	.	.
1e−5	2	8.9e5	8.4e5	9.4e5	1.2e5	.	.	.	.	.
1e−8	0	*33e−4*	*43e−7*	*16e−2*	8.9e4	.	.	.	.	.

Δf	f_{119} in 5-D,N=15,mFE=125000 #	ERT	10%	90%	RT_{succ}	f_{119} in 20-D, N=15, mFE=2000000 #	ERT	10%	90%	RT_{succ}
10	15	1.9e1	1.5e1	2.3e1	1.9e1	15	7.0e3	5.7e3	8.5e3	7.0e3
1	15	8.3e2	6.3e2	1.0e3	8.3e2	10	1.4e6	9.9e5	1.8e6	8.6e5
1e−1	15	4.5e3	4.0e3	5.0e3	4.5e3	4	6.8e6	6.3e6	7.3e6	1.8e6
1e−3	15	1.3e4	1.2e4	1.5e4	1.3e4	0	*13e−2*	*60e−3*	*33e−1*	1.6e6
1e−5	8	1.6e5	1.4e5	1.9e5	1.0e5	.	.	.	.	.
1e−8	0	*61e−7*	*12e−7*	*42e−6*	7.1e4	.	.	.	.	.

Δf	f_{120} in 5-D,N=15,mFE=125000 #	ERT	10%	90%	RT_{succ}	f_{120} in 20-D, N=15, mFE=2000000 #	ERT	10%	90%	RT_{succ}
10	15	1.7e1	1.1e1	2.1e1	1.7e1	15	4.6e4	3.0e4	6.5e4	4.6e4
1	15	3.4e3	1.8e3	5.1e3	3.4e3	0	*56e−1*	*41e−1*	*77e−1*	1.0e6
1e−1	11	9.6e4	7.6e4	1.1e5	6.4e4	.	.	.	.	.
1e−3	0	*34e−3*	*89e−4*	*40e−2*	7.9e4	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Table 1: Shown are, for functions f_{101}-f_{120} and for a given target difference to the optimal function value Δf: the number of successful trials (#); the expected running time to surpass $f_{\text{opt}} + \Delta f$ (ERT, see Figure 2); the 10%-tile and 90%-tile of the bootstrap distribution of ERT; the average number of function evaluations in successful trials or, if none was successful, as last entry the median number of function evaluations to reach the best function value (RT_{succ}). If $f_{\text{opt}} + \Delta f$ was never reached, figures in *italics* denote the best achieved Δf-value of the median trial and the 10% and 90%-tile trial. Furthermore, N denotes the number of trials, and mFE denotes the maximum of number of function evaluations executed in one trial. See Figure 2 for the names of functions.

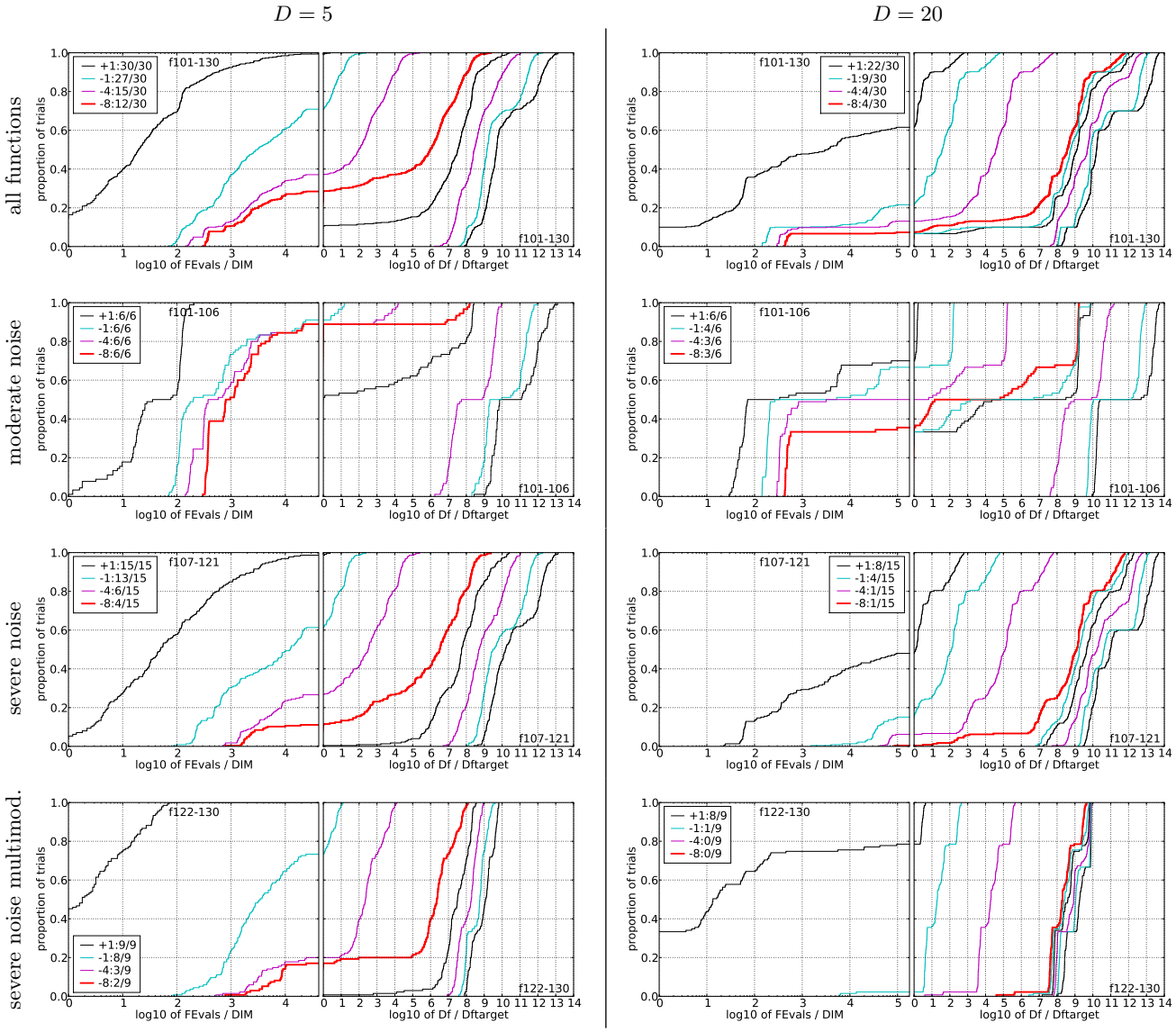

Figure 3: Empirical cumulative distribution functions (ECDFs), plotting the fraction of trials versus running time (left) or Δf. Left subplots: ECDF of the running time (number of function evaluations), divided by search space dimension D, to fall below $f_{\mathrm{opt}} + \Delta f$ with $\Delta f = 10^k$, where k is the first value in the legend. Right subplots: ECDF of the best achieved Δf divided by 10^k (upper left lines in continuation of the left subplot), and best achieved Δf divided by 10^{-8} for running times of $D, 10\,D, 100\,D \ldots$ function evaluations (from right to left cycling black-cyan-magenta). Top row: all results from all functions; second row: moderate noise functions; third row: severe noise functions; fourth row: severe noise and highly-multimodal functions. The legends indicate the number of functions that were solved in at least one trial. FEvals denotes number of function evaluations, D and DIM denote search space dimension, and Δf and Df denote the difference to the optimal function value.

Δf	#	ERT	10%	90%	RT_{succ}
f_{121} in 5-D,N=15,mFE=125000					
10	15	2.4e1	2.0e1	2.9e1	2.4e1
1	15	4.1e2	3.3e2	5.0e2	4.1e2
1e−1	15	1.4e3	1.2e3	1.7e3	1.4e3
1e−3	8	1.9e5	1.7e5	2.1e5	1.0e5
1e−5	0	*99e−5*	*18e−5*	*29e−4*	8.9e4
1e−8	.	.	.	.	.
f_{121} in 20-D, N=15, mFE=2000000					
10	15	9.7e2	8.9e2	1.1e3	9.7e2
1	15	7.4e4	4.1e4	1.1e5	7.4e4
1e−1	2	1.4e7	1.3e7	1.5e7	2.0e6
1e−3	0	*18e−2*	*88e−3*	*25e−2*	8.9e5
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{122} in 5-D,N=15,mFE=125000					
10	15	1.6e1	8.9e0	2.4e1	1.6e1
1	15	7.5e3	4.9e3	1.0e4	7.5e3
1e−1	15	4.5e4	4.0e4	5.0e4	4.5e4
1e−3	1	1.8e6	1.8e6	1.9e6	1.2e5
1e−5	0	*76e−4*	*15e−4*	*46e−3*	1.1e5
1e−8	.	.	.	.	.
f_{122} in 20-D, N=15, mFE=2000000					
10	15	7.1e2	4.2e2	1.0e3	7.1e2
1	0	*27e−1*	*17e−1*	*36e−1*	1.1e6
1e−1	.	.	.	.	.
1e−3	.	.	.	.	.
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{123} in 5-D,N=15,mFE=125000					
10	15	1.5e1	1.2e1	1.8e1	1.5e1
1	14	3.9e4	2.6e4	5.3e4	3.8e4
1e−1	0	*64e−2*	*52e−2*	*97e−2*	3.5e4
1e−3	.	.	.	.	.
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{123} in 20-D, N=15, mFE=2000000					
10	15	1.1e3	7.1e2	1.5e3	1.1e3
1	0	*45e−1*	*36e−1*	*55e−1*	1.3e6
1e−1	.	.	.	.	.
1e−3	.	.	.	.	.
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{124} in 5-D,N=15,mFE=125000					
10	15	1.7e1	1.1e1	2.5e1	1.7e1
1	15	1.0e3	8.5e2	1.2e3	1.0e3
1e−1	2	9.0e5	8.7e5	9.4e5	1.2e5
1e−3	0	*20e−2*	*87e−3*	*37e−2*	8.9e4
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{124} in 20-D, N=15, mFE=2000000					
10	15	2.3e2	2.0e2	2.6e2	2.3e2
1	1	3.0e7	3.0e7	3.0e7	2.0e6
1e−1	0	*16e−1*	*12e−1*	*19e−1*	1.3e6
1e−3	.	.	.	.	.
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{125} in 5-D,N=15,mFE=125000					
10	15	1.3e0	1.1e0	1.5e0	1.3e0
1	15	3.5e1	2.6e1	4.5e1	3.5e1
1e−1	15	4.7e3	3.3e3	6.5e3	4.7e3
1e−3	0	*80e−4*	*33e−4*	*19e−3*	8.9e4
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{125} in 20-D, N=15, mFE=2000000					
10	15	1.2e0	1.1e0	1.3e0	1.2e0
1	15	1.5e3	1.3e3	1.7e3	1.5e3
1e−1	0	*39e−2*	*36e−2*	*42e−2*	1.0e6
1e−3	.	.	.	.	.
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{126} in 5-D,N=15,mFE=125000					
10	15	1.1e0	1.0e0	1.1e0	1.1e0
1	15	3.5e1	2.6e1	4.5e1	3.5e1
1e−1	15	9.4e3	7.6e3	1.1e4	9.4e3
1e−3	0	*27e−3*	*89e−4*	*45e−3*	7.1e4
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{126} in 20-D, N=15, mFE=2000000					
10	15	1.1e0	1.0e0	1.1e0	1.1e0
1	15	2.5e3	2.1e3	2.9e3	2.5e3
1e−1	0	*52e−2*	*38e−2*	*54e−2*	1.0e6
1e−3	.	.	.	.	.
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{127} in 5-D,N=15,mFE=125000					
10	15	1.2e0	1.1e0	1.3e0	1.2e0
1	15	3.9e1	3.0e1	4.8e1	3.9e1
1e−1	15	8.6e3	5.6e3	1.2e4	8.6e3
1e−3	0	*32e−3*	*86e−4*	*51e−3*	6.3e4
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{127} in 20-D, N=15, mFE=2000000					
10	15	1.1e0	1.0e0	1.1e0	1.1e0
1	15	4.2e2	3.3e2	5.1e2	4.2e2
1e−1	0	*43e−2*	*34e−2*	*49e−2*	1.3e6
1e−3	.	.	.	.	.
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{128} in 5-D,N=15,mFE=51751					
10	15	1.2e2	9.6e1	1.5e2	1.2e2
1	15	5.5e3	4.2e3	6.9e3	5.5e3
1e−1	15	9.9e3	7.7e3	1.2e4	9.9e3
1e−3	15	1.9e4	1.5e4	2.4e4	1.9e4
1e−5	15	2.4e4	1.9e4	2.8e4	2.4e4
1e−8	15	2.9e4	2.4e4	3.3e4	2.9e4
f_{128} in 20-D, N=15, mFE=2000000					
10	1	3.0e7	3.0e7	3.0e7	2.0e6
1	0	*30e+0*	*22e+0*	*36e+0*	1.4e6
1e−1	.	.	.	.	.
1e−3	.	.	.	.	.
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{129} in 5-D,N=15,mFE=125000					
10	15	9.8e1	7.2e1	1.2e2	9.8e1
1	15	1.1e4	6.0e3	1.6e4	1.1e4
1e−1	11	9.3e4	7.2e4	1.1e5	7.9e4
1e−3	1	1.8e6	1.8e6	1.9e6	1.2e5
1e−5	1	1.8e6	1.8e6	1.9e6	1.2e5
1e−8	0	*16e−3*	*10e−4*	*26e−2*	6.3e4
f_{129} in 20-D, N=15, mFE=2000000					
10	0	*27e+0*	*24e+0*	*42e+0*	1.3e6
1	.	.	.	.	.
1e−1	.	.	.	.	.
1e−3	.	.	.	.	.
1e−5	.	.	.	.	.
1e−8	.	.	.	.	.
f_{130} in 5-D,N=15,mFE=125000					
10	15	1.3e2	1.0e2	1.7e2	1.3e2
1	14	2.2e4	9.2e3	3.6e4	2.2e4
1e−1	11	5.2e4	3.0e4	7.4e4	4.9e4
1e−3	11	7.0e4	4.5e4	9.6e4	5.7e4
1e−5	11	8.4e4	6.2e4	1.1e5	6.7e4
1e−8	8	1.6e5	1.3e5	1.8e5	1.0e5
f_{130} in 20-D, N=15, mFE=2000000					
10	15	1.2e5	5.5e4	2.0e5	1.2e5
1	3	8.1e6	6.8e6	9.4e6	1.3e6
1e−1	3	8.2e6	6.9e6	9.4e6	1.4e6
1e−3	1	3.0e7	3.0e7	3.0e7	2.0e6
1e−5	0	*21e−1*	*40e−4*	*49e−1*	1.4e6
1e−8	.	.	.	.	.

Table 2: Shown are, for functions f_{121}-f_{130} and for a given target difference to the optimal function value Δf: the number of successful trials (#); the expected running time to surpass $f_{\text{opt}} + \Delta f$ (ERT, see Figure 2); the 10%-tile and 90%-tile of the bootstrap distribution of ERT; the average number of function evaluations in successful trials or, if none was successful, as last entry the median number of function evaluations to reach the best function value (RT_{succ}). If $f_{\text{opt}} + \Delta f$ was never reached, figures in *italics* denote the best achieved Δf-value of the median trial and the 10% and 90%-tile trial. Furthermore, N denotes the number of trials, and mFE denotes the maximum of number of function evaluations executed in one trial. See Figure 2 for the names of functions.

X. Y. at al., editor, *Parallel Problem Solving for Nature - PPSN VIII, LNCS 3242*, pages 282–291. Springer, 2004.

[10] N. Hansen, S. Müller, and P. Koumoutsakos. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation*, 1(11):1–18, 2003.

[11] N. Hansen and A. Ostermeier. Adapting Arbitrary Normal Mutation Distributions in Evolution Strategies: The Covariance Matrix Adaptation. In *Proceeding of the IEEE International Conference on Evolutionary Computation (ICEC '96)*, pages 312–317, 1996.

[12] W. Hart. *Adaptive Global Optimization With Local Search*. PhD thesis, Univ. California, San Diego, CA., 1994.

[13] N. Krasnogor and J. Smith. A Tutorial for Competent Memetic Algorithms: Model, Taxonomy, and Design Issue. *IEEE Transactions on Evolutionary Computation*, 9(5):474–488, 2005.

[14] M. S. Land. *Evolutionary Algorithms with Local Search for Combinational Optimization*. PhD thesis, Univ. California, San Diego, CA., 1998.

[15] P. Merz. *Memetic Algorithms for Combinational Optimization Problems: Fitness Landscapes and Effective Search Strategies*. PhD thesis, Gesamthochschule Siegen, University of Siegen, Germany, 2000.

[16] D. Molina, M. Lozano, C. García-Martínez, and F. Herrera. Memetic algorithms for continuous optimization based on local search chains. *Evolutionary Computation. In press*, 2009.

[17] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical report, Technical Report Caltech Concurrent Computation Program Report 826, Caltech, Pasadena, California, 1989.

[18] P. Moscato. *Memetic algorithms: a short introduction*, pages 219–234. McGraw-Hill, London, 1999.

[19] P. Suganthan, N. Hansen, J. Liang, K. Deb, Y. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the CEC 2005 special session on real parameter optimization. Technical report, Nanyang Technical University, 2005.

[20] G. Syswerda. Uniform Crossover in Genetic Algorithms. In J. D. Schaffer, editor, *Proc. of the Thrid Int. Conf. on Genetic Algorithms*, pages 2–9. Morgan Kaufmann Publishers, San Mateo, 1989.

[21] E. Talbi. A Taxonomy of Hybrid Metaheuristics. *Journal of Heuristics*, 8:541–564, 2002.