Application of a simple binary genetic algorithm to a noiseless testbed benchmark

Miguel Nicolau

Application of a simple binary genetic algorithm to a noiseless testbed benchmark

Miguel Nicolau
INRIA Saclay - Île-de-France
LRI - Université Paris Sud
Paris, France
miguel.nicolau@lri.fr

ABSTRACT

One of the earliest evolutionary computation algorithms, the genetic algorithm, is applied to the noise-free BBOB 2009 testbed. It is adapted to the continuous domain by increasing the number of bits encoding each variable, until a desired resolution is possible to achieve. Good results and scaling are obtained for separable functions, but poor performance is achieved on the other functions, particularly ill-conditioned functions. Overall running times remain fast throughout.

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, Genetic algorithms

1. INTRODUCTION

One of the earliest and simplest evolutionary algorithms ever designed is the genetic algorithm [9, 3]. It is a stochastic, population-based algorithm, in which binary strings are modified through the use of genetic operators, and a fitness score dictates the worthiness of an individual.

Genetic algorithms (GAs), in their simplest form (as presented here), have been applied to a multitude of problem domains [10]. However, their continued application has highlighted one of their major drawbacks: the inability to scale up, as the problem dimension increases [12]. In answer to this problem, a new breed of genetic algorithms, the so-called Messy-GAs [5] have been developed. These GAs are certainly better suited for higher order problems; however, they are quite complex and hard to deploy. As a result, the simplicity of developing and applying a simple GA to a variety of problems remains one of its biggest strengths.

The current paper therefore adapts the original, simple binary GA to a continuous domain problem, and reports on its performance. Although not achieving stellar performance, particularly in comparison with more recent and performing algorithms, the results obtained are nevertheless remarkable, particularly in separable functions.

2. THE ALGORITHM

The typical cycle of a genetic algorithm is as follows:

1. Create a population of individuals, each representing a fully contained potential solution to the problem;
2. Apply each of the potential solutions to the problem, thus associating a fitness score to each of them;
3. Apply a selection operator (usually fitness-based), to choose two individuals;
4. Apply a crossover operator (with probability P_c, exchanging genetic material between the two parents, and thus creating a new pair of individuals (offspring);
5. Slightly mutate the binary encoding of each offspring with probability P_m;
6. Evaluate the solutions encoded by each of the new offspring individuals;
7. Repeat steps 3-6 until a full offspring population is created;
8. Replace the parent population based on the offspring population and on a chosen replacement strategy.

The actual code used is based on the freely available IlliGAL sga code [11], which was modified and adjusted prior to application to the BBO-Benchmark suite.

2.1 Encoding

Being based on binary strings, GAs are typically applied to binary problem domains. It is not uncommon to see them applied to discrete integer domains, but their application to continuous domains (as in the BBOB suite of functions [7]) is not widespread.

The choice for this implementation was to encode each variable as a n-bit sequence, which is then scaled, such that it represents a floating point value in the range $[-5, 5]^D$, where D denotes the dimension of the problem. The choice of n depends on the resolution required; as success on a particular function is defined as having a distance to the optimum smaller than 10^{-8} [7], n must be set such that

$$2^n \approx 10^{-8}$$

This wields a choice of $n \geq 30$.

Another matter is the choice of encoding. A typical binary encoding has the disadvantage of introducing local optima to the fitness landscape. For example, a value of 01111111 for an 8-bit encoded variable is extremely close to an optimum of 10000000, but it is also at the maximum hamming-distance of that optimum value, making it very hard to reach (any single binary change to 01111111 will reduce its fitness).

A potential solution to this problem is to use alternative encodings, such as Gray-encoding [10], where a small change to a binary string corresponds to a small change to its decimal equivalent. This however takes away one of the major advantages of using a binary encoding, which is the uniform probability that a single bit-flip makes either a large or small change to a value (depending on which bit is flipped). A gray-encoded variable is much more dependent on its initial starting value.

The final choice therefore relied on using a standard binary encoding, but with a higher value of $n = 32$. While this slightly increases the search space, it also introduces redundancy on the encoding, making it less likely to encounter representation-generated local optima.

2.2 Selection

There are several selection techniques available to the GA practitioner [4]. One of the most commonly used is the fitness-based roulette-wheel, in which an individual's chance of being selected is proportional to its fitness. It is however a highly elitist method, and has been shown to lead to premature convergence [1].

For harder problem domains, such as the ones on the BBO-Benchmark suite, the tournament selection scheme is more appropriate, and has been used in this work. Its actual implementation is the original one from the sga code [11]:

1. Start with a selection pool containing all individuals;
2. Select t individuals from the selection pool (t being the tournament size);
3. From these, pick the one that has the best fitness;
4. Remove all t individuals from the selection pool;
5. When the selection pool becomes empty, reset it to contain all individuals again.

This implementation has the advantage of ensuring that each individual gets at least one chance of participating in a selection tournament.

2.3 Replacement

Typically, there are two replacement strategies in GAs, generational and steady-state. Generational replacement simply replaces the current population with the offspring population, whereas steady-state replacement ensures that only the best individuals get through to the next generation. Given the nature of the BBO-Benchmark function suites, a generational replacement scheme seemed more adequate, and was thus used in the current work.

2.4 Genetic Operators

There are two main genetic operators in GAs, crossover and mutation. Crossover consists in exchanging genetic material between two individuals, whereas mutation is an operator that slightly alters the genetic code of a single individual.

In the current work, a 2-point crossover was implemented and used. This consists in selecting two individuals, choosing two random points in each (corresponding points in each individual), and exchanging the genetic material that lies between the two points. As the boundaries of each variable are known (every set of n bits corresponds to a variable), the choice of crossover points was limited to variable boundaries.

The mutation operator used was a simple bit-flipping mutation: each bit in the whole genetic sequence of an individual is flipped with a probability P_m.

2.5 Parameters

The same parameters and settings were used for all functions, such that the crafting effort [6] computes to $CrE = 0$.

2.5.1 *Population size and number of generations*

The maximum number of function evaluations per function was set to $F_{max} = 10^5 * D$, where D is the dimension of the problem, so the population size P and number of generations G must be set such that:

$$P \times G \approx 10^5 * D$$

Choosing the population size in a GA has largely been an experimental parameter. A few population-sizing models have been proposed, such as the Gambler's Ruin Problem model [8], but these are usually based on the distribution of small binary building blocks in the original population, which is infeasible given the current setup ($n = 32$).

After some (small) experimentation, the following formulae were used:

$$P = \lfloor \sqrt{F_{max}} * 5 \rfloor$$

$$G = (F_{max}/P) + (F_{max}\%P \neq 0)$$

where $(F_{max}\%P \neq 0)$ returns 0 if F_{max} is divisible by P, and 1 otherwise.

2.5.2 *Crossover and mutation probabilities*

The probability of crossover was set to 1.0. The mutation probability is slightly harder to set; a high value favours early exploration, whereas a low value is required to fine tune good values found earlier. As the population size is quite large, favouring early diversity, the mutation probability was set to a low value (after a few experimental runs):

$$P_c = 1.0$$

$$P_m = \frac{2.0}{L}$$

where L is the length of the binary strings ($L = n * D$).

2.5.3 Tournament size

The tournament size to use during selection is another non-obvious parameter to set. A low value helps to keep diversity in the population, but a higher value favours the spreading of good allele throughout the population. Quick experimentation with a subset of the target functions yielded the following (experimental) value:

$$t = \frac{P}{500}$$

3. RESULTS

Results from experiments according to [6] on the benchmark functions given in [2, 7] are presented in Figures 1 and 2 and in Table 1. The whole experiment (all dimensions, all functions, all instances) took less than a day to execute.

The results obtained show that the GA performs quite well on separable functions 1-4, both in terms of results and scaling, and, although not solving it to the resolution required (10^{-8}), scales fairly well for function 5. It seems that the application of the crossover operator is quite useful for this kind of functions.

Some functions are very hard for the GA; the Ellipsoid (function 10) and Lunacek bi-Rastrigin (function 24) are obvious examples. In any case, apart from the separable functions, there is an obvious difficulty for the GA to scale-up to the dimension of the problem.

4. CPU TIMING EXPERIMENT

For the timing experiment the GA was run with a maximum of $10^5 \times D$ function evaluations and restarted until 30 seconds has passed (according to Figure 2 in [6]). The experiments have been conducted with a Dual Core AMD Opteron processor, running at 1.8GHz, under Linux, using a C-implementation (gcc version 4.1.2). The time per function evaluation was 2.9; 4.2; 6.7; 12; 24; 47 times 10^{-6} seconds in dimensions 2; 3; 5; 10; 20; 40 respectively.

5. CONCLUSION

The application of the simple GA algorithm to a testbed of noiseless continuous functions has been presented. The results obtained show that its adaptation to continuous domains is possible and easy to achieve. However, apart from separable functions, where the exchange of genetic material through the crossover operator boosts performance considerably, the lack of scaling of the GA is obvious. It remains however a simple to implement and quite fast approach. Further tuning of parameters could potentially wield better results.

6. REFERENCES

[1] J. E. Baker. Reducing bias and inefficiency in the selection algorithm. In J. J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms, Cambridge, MA, USA, July 1987*, pages 14–21. Lawrence Erlbaum Associates, 1987.

[2] 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.

[3] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison Wesley, 1989.

[4] D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. In G. J. E. Rawlins, editor, *Proceedings of the First Workshop on Foundations of Genetic Algorithms. Bloomington Campus, Indiana, USA, July 15-18 1990*, pages 69–93. Morgan Kaufmann, 1991.

[5] D. E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3(5):493–530, 1989.

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

[7] 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.

[8] G. R. Harik, E. Cantú-Paz, D. E. Goldberg, and B. L. Miller. The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation*, 7(3):231–253, 1999.

[9] J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

[10] M. Mitchell. *An Introduction to Genetic Algorithms.* MIT Press, Cambridge, MA, USA, 1996.

[11] R. E. Smith and D. E. Goldberg. Sga-c: A c-language implementation of a simple genetic algorithm. Technical Report 91002, Illinois Genetic Algorithms Laboratory, Urbana, IL, USA, 1991.

[12] D. Thierens. Scalability problems of simple genetic algorithms. *Evolutionary Computation*, 7(4):331–352, 1999.

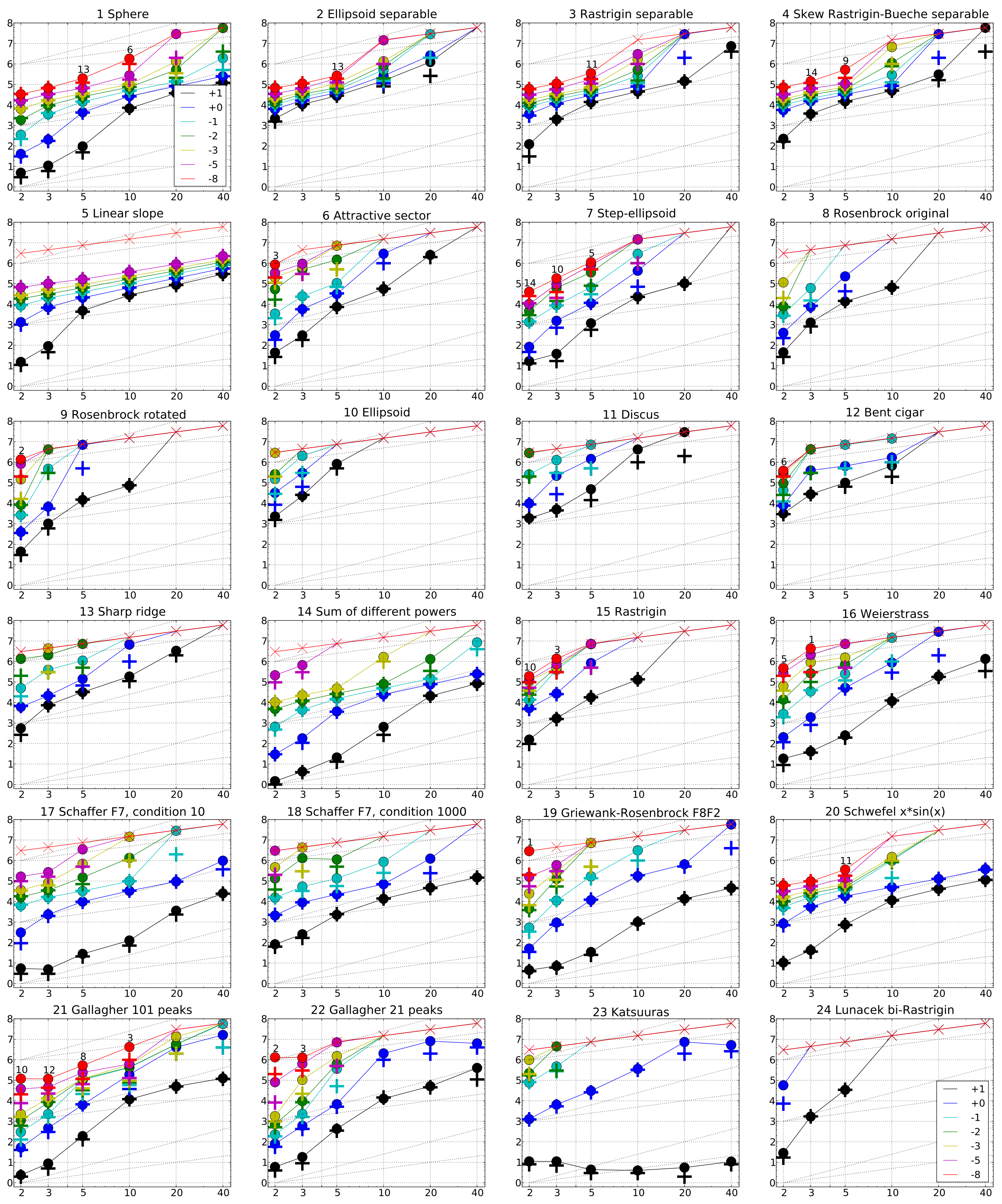

Figure 1: Expected Running Time (ERT, ●) to reach $f_{\rm opt} + \Delta f$ and median number of function evaluations of successful trials (+), shown for $\Delta f = 10, 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-5}, 10^{-8}$ (the exponent is given in the legend of f_1 and f_{24}) versus dimension in log-log presentation. The ERT(Δf) equals to #FEs(Δf) divided by the number of successful trials, where a trial is successful if $f_{\rm opt} + \Delta f$ was surpassed during the trial. The #FEs(Δf) are the total number of function evaluations while $f_{\rm opt} + \Delta f$ was not surpassed during the trial from all respective trials (successful and unsuccessful), and $f_{\rm opt}$ denotes the optimal function value. Crosses (×) indicate the total number of function evaluations #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.

	f_1 in 5-D, N=15, mFE=502112					**f_1 in 20-D**, N=15, mFE=2.00e6				
Δf	#	ERT	10%	90%	RT$_{\text{succ}}$	#	ERT	10%	90%	RT$_{\text{succ}}$
10	15	9.6e1	5.7e1	1.4e2	9.6e1	15	3.8e4	3.7e4	3.8e4	3.8e4
1	15	4.4e3	3.9e3	5.0e3	4.4e3	15	8.2e4	8.0e4	8.3e4	8.2e4
1e−1	15	1.4e4	1.4e4	1.5e4	1.4e4	15	1.4e5	1.3e5	1.4e5	1.4e5
1e−3	15	3.6e4	3.5e4	3.7e4	3.6e4	10	1.3e6	8.7e5	1.9e6	8.1e5
1e−5	15	6.6e4	6.4e4	6.7e4	6.6e4	1	2.9e7	1.4e7	>3e7	2.0e6
1e−8	13	2.0e5	1.5e5	2.6e5	1.8e5	0	*74e−5*	*21e−6*	*11e−3*	1.4e6

	f_2 in 5-D, N=15, mFE=502112					**f_2 in 20-D**, N=15, mFE=2.00e6				
Δf	#	ERT	10%	90%	RT$_{\text{succ}}$	#	ERT	10%	90%	RT$_{\text{succ}}$
10	15	2.8e4	2.7e4	2.9e4	2.8e4	10	1.2e6	8.1e5	1.8e6	9.3e5
1	15	4.0e4	3.9e4	4.1e4	4.0e4	7	2.6e6	1.7e6	4.4e6	1.0e6
1e−1	15	5.4e4	5.2e4	5.5e4	5.4e4	1	2.8e7	1.3e7	>3e7	2.0e6
1e−3	14	1.2e5	8.2e4	1.6e5	1.1e5	0	*29e−1*	*11e−2*	*42e+0*	1.4e6
1e−5	13	2.0e5	1.5e5	2.6e5	1.8e5	.	.	.	.	.
1e−8	13	2.7e5	2.2e5	3.2e5	2.4e5	.	.	.	.	.

	f_3 in 5-D, N=15, mFE=502112					**f_3 in 20-D**, N=15, mFE=2.00e6				
Δf	#	ERT	10%	90%	RT$_{\text{succ}}$	#	ERT	10%	90%	RT$_{\text{succ}}$
10	15	1.3e4	1.3e4	1.4e4	1.3e4	15	1.4e5	1.4e5	1.5e5	1.4e5
1	15	3.0e4	2.9e4	3.0e4	3.0e4	1	2.8e7	1.3e7	>3e7	2.0e6
1e−1	15	4.1e4	4.1e4	4.2e4	4.1e4	0	*21e−1*	*10e−1*	*31e−1*	1.3e6
1e−3	15	7.1e4	6.9e4	7.3e4	7.1e4	.	.	.	.	.
1e−5	13	1.8e5	1.3e5	2.5e5	1.7e5	.	.	.	.	.
1e−8	11	3.6e5	2.7e5	4.7e5	2.6e5	.	.	.	.	.

	f_4 in 5-D, N=15, mFE=502112					**f_4 in 20-D**, N=15, mFE=2.00e6				
Δf	#	ERT	10%	90%	RT$_{\text{succ}}$	#	ERT	10%	90%	RT$_{\text{succ}}$
10	15	1.5e4	1.4e4	1.6e4	1.5e4	14	3.1e5	1.7e5	4.7e5	3.0e5
1	15	3.2e4	3.1e4	3.3e4	3.2e4	1	2.9e7	1.4e7	>3e7	2.0e6
1e−1	15	4.4e4	4.2e4	4.5e4	4.4e4	0	*34e−1*	*13e−1*	*74e−1*	1.3e6
1e−3	15	7.5e4	7.2e4	7.8e4	7.5e4	.	.	.	.	.
1e−5	15	1.1e5	1.1e5	1.1e5	1.1e5	.	.	.	.	.
1e−8	9	5.2e5	3.8e5	7.6e5	3.0e5	.	.	.	.	.

	f_5 in 5-D, N=15, mFE=502112					**f_5 in 20-D**, N=15, mFE=2.00e6				
Δf	#	ERT	10%	90%	RT$_{\text{succ}}$	#	ERT	10%	90%	RT$_{\text{succ}}$
10	15	4.8e3	4.2e3	5.5e3	4.8e3	15	8.8e4	8.6e4	8.9e4	8.8e4
1	15	2.1e4	2.0e4	2.2e4	2.1e4	15	1.9e5	1.8e5	1.9e5	1.9e5
1e−1	15	4.0e4	3.8e4	4.1e4	4.0e4	15	3.1e5	3.0e5	3.1e5	3.1e5
1e−3	15	9.2e4	9.0e4	9.4e4	9.2e4	15	5.7e5	5.7e5	5.8e5	5.7e5
1e−5	15	1.7e5	1.6e5	1.7e5	1.7e5	15	9.0e5	8.9e5	9.1e5	9.0e5
1e−8	0	*35e−9*	*22e−9*	*45e−9*	4.5e5	0	*11e−8*	*62e−9*	*12e−8*	1.4e6

	f_6 in 5-D, N=15, mFE=502112					**f_6 in 20-D**, N=15, mFE=2.00e6				
Δf	#	ERT	10%	90%	RT$_{\text{succ}}$	#	ERT	10%	90%	RT$_{\text{succ}}$
10	15	7.6e3	6.2e3	9.0e3	7.6e3	7	2.5e6	1.8e6	4.0e6	1.3e6
1	15	3.2e4	2.9e4	3.5e4	3.2e4	0	*11e+0*	*33e−1*	*22e+0*	1.8e6
1e−1	14	1.1e5	7.3e4	1.5e5	1.0e5	.	.	.	.	.
1e−3	1	7.1e6	3.4e6	>7e6	5.0e5	.	.	.	.	.
1e−5	0	*24e−3*	*25e−4*	*78e−3*	4.5e5	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_7 in 5-D, N=15, mFE=502112					**f_7 in 20-D**, N=15, mFE=2.00e6				
Δf	#	ERT	10%	90%	RT$_{\text{succ}}$	#	ERT	10%	90%	RT$_{\text{succ}}$
10	15	1.2e3	8.1e2	1.6e3	1.2e3	15	1.0e5	9.9e4	1.1e5	1.0e5
1	15	1.1e4	1.0e4	1.2e4	1.1e4	0	*32e−1*	*12e−1*	*61e−1*	5.0e5
1e−1	14	6.7e4	3.1e4	1.1e5	6.5e4	.	.	.	.	.
1e−3	6	8.2e5	5.7e5	1.3e6	4.3e5	.	.	.	.	.
1e−5	6	8.2e5	5.9e5	1.3e6	4.3e5	.	.	.	.	.
1e−8	5	1.1e6	7.5e5	2.1e6	4.2e5	.	.	.	.	.

	f_8 in 5-D, N=15, mFE=502112					**f_8 in 20-D**, N=15, mFE=2.00e6				
Δf	#	ERT	10%	90%	RT$_{\text{succ}}$	#	ERT	10%	90%	RT$_{\text{succ}}$
10	15	1.4e4	1.2e4	1.5e4	1.4e4	0	*17e+0*	*12e+0*	*22e+0*	1.3e6
1	11	2.3e5	1.3e5	3.7e5	1.3e5	.	.	.	.	.
1e−1	0	*78e−2*	*31e−2*	*12e−1*	5.0e4	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_9 in 5-D, N=15, mFE=502112					**f_9 in 20-D**, N=15, mFE=2.00e6				
Δf	#	ERT	10%	90%	RT$_{\text{succ}}$	#	ERT	10%	90%	RT$_{\text{succ}}$
10	15	1.5e4	1.3e4	1.6e4	1.5e4	0	*19e+0*	*17e+0*	*20e+0*	1.4e6
1	1	7.1e6	3.3e6	>7e6	5.9e4	.	.	.	.	.
1e−1	0	*17e−1*	*11e−1*	*29e−1*	2.5e5	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_{10} in 5-D, N=15, mFE=502112					**f_{10} in 20-D**, N=15, mFE=2.00e6				
Δf	#	ERT	10%	90%	RT$_{\text{succ}}$	#	ERT	10%	90%	RT$_{\text{succ}}$
10	6	8.3e5	5.3e5	1.5e6	3.0e5	0	*15e+3*	*75e+2*	*32e+3*	1.6e6
1	0	*15e+0*	*38e−1*	*77e+0*	1.0e5	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_{11} in 5-D, N=15, mFE=502112					**f_{11} in 20-D**, N=15, mFE=2.00e6				
Δf	#	ERT	10%	90%	RT$_{\text{succ}}$	#	ERT	10%	90%	RT$_{\text{succ}}$
10	14	4.8e4	1.2e4	9.0e4	4.7e4	1	2.9e7	1.4e7	>3e7	1.3e6
1	4	1.4e6	8.2e5	3.1e6	2.7e5	0	*20e+0*	*16e+0*	*40e+0*	2.0e6
1e−1	1	7.1e6	3.4e6	>7e6	5.0e5	.	.	.	.	.
1e−3	0	*21e−1*	*68e−2*	*66e−1*	7.9e4	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_{12} in 5-D, N=15, mFE=502112					**f_{12} in 20-D**, N=15, mFE=2.00e6				
Δf	#	ERT	10%	90%	RT$_{\text{succ}}$	#	ERT	10%	90%	RT$_{\text{succ}}$
10	14	1.0e5	6.3e4	1.5e5	6.5e4	0	*14e+2*	*45e+0*	*60e+2*	1.4e6
1	7	6.6e5	4.2e5	1.1e6	2.6e5	.	.	.	.	.
1e−1	1	7.1e6	3.4e6	>7e6	5.0e5	.	.	.	.	.
1e−3	0	*11e−1*	*15e−2*	*83e−1*	4.5e5	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_{13} in 5-D, N=15, mFE=502112					**f_{13} in 20-D**, N=15, mFE=2.00e6				
Δf	#	ERT	10%	90%	RT$_{\text{succ}}$	#	ERT	10%	90%	RT$_{\text{succ}}$
10	15	3.2e4	3.1e4	3.3e4	3.2e4	6	3.3e6	2.1e6	6.2e6	8.7e5
1	13	1.4e5	8.6e4	2.1e5	1.3e5	0	*10e+0*	*59e−1*	*25e+0*	1.4e6
1e−1	5	1.1e6	7.8e5	1.8e6	5.0e5	.	.	.	.	.
1e−3	0	*20e−2*	*18e−3*	*10e−1*	4.5e5	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_{14} in 5-D, N=15, mFE=502112					**f_{14} in 20-D**, N=15, mFE=2.00e6				
Δf	#	ERT	10%	90%	RT$_{\text{succ}}$	#	ERT	10%	90%	RT$_{\text{succ}}$
10	15	2.1e1	1.4e1	2.8e1	2.1e1	15	2.1e4	1.9e4	2.2e4	2.1e4
1	15	3.7e3	3.1e3	4.3e3	3.7e3	15	7.6e4	7.4e4	7.9e4	7.6e4
1e−1	15	1.5e4	1.5e4	1.6e4	1.5e4	15	1.5e5	1.4e5	1.5e5	1.5e5
1e−3	15	4.9e4	4.6e4	5.1e4	4.9e4	0	*72e−4*	*32e−4*	*17e−3*	1.6e6
1e−5	0	*13e−5*	*27e−6*	*49e−5*	1.8e5	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_{15} in 5-D, N=15, mFE=502112					**f_{15} in 20-D**, N=15, mFE=2.00e6				
Δf	#	ERT	10%	90%	RT$_{\text{succ}}$	#	ERT	10%	90%	RT$_{\text{succ}}$
10	15	1.8e4	1.7e4	1.9e4	1.8e4	0	*25e+0*	*21e+0*	*37e+0*	1.6e6
1	6	8.4e5	5.8e5	1.4e6	3.7e5	.	.	.	.	.
1e−1	1	7.1e6	3.3e6	>7e6	5.0e5	.	.	.	.	.
1e−3	1	7.1e6	3.4e6	>7e6	5.0e5	.	.	.	.	.
1e−5	1	7.2e6	3.4e6	>7e6	5.0e5	.	.	.	.	.
1e−8	0	*15e−1*	*99e−2*	*20e−1*	3.5e5	.	.	.	.	.

	f_{16} in 5-D, N=15, mFE=502112					**f_{16} in 20-D**, N=15, mFE=2.00e6				
Δf	#	ERT	10%	90%	RT$_{\text{succ}}$	#	ERT	10%	90%	RT$_{\text{succ}}$
10	15	2.5e2	1.8e2	3.2e2	2.5e2	15	1.9e5	1.8e5	2.1e5	1.9e5
1	15	5.2e4	4.3e4	6.1e4	5.2e4	1	2.8e7	1.3e7	>3e7	2.0e6
1e−1	12	2.5e5	1.7e5	3.5e5	1.8e5	0	*24e−1*	*13e−1*	*51e−1*	1.8e6
1e−3	4	1.6e6	9.8e5	3.2e6	4.2e5	.	.	.	.	.
1e−5	1	7.2e6	3.5e6	>7e6	5.0e5	.	.	.	.	.
1e−8	0	*39e−3*	*13e−5*	*14e−2*	3.5e5	.	.	.	.	.

	f_{17} in 5-D, N=15, mFE=502112					**f_{17} in 20-D**, N=15, mFE=2.00e6				
Δf	#	ERT	10%	90%	RT$_{\text{succ}}$	#	ERT	10%	90%	RT$_{\text{succ}}$
10	15	2.8e1	2.1e1	3.6e1	2.8e1	15	3.6e3	2.7e3	4.5e3	3.6e3
1	15	9.9e3	9.2e3	1.1e4	9.9e3	15	9.4e4	9.2e4	9.7e4	9.4e4
1e−1	15	3.3e4	3.2e4	3.3e4	3.3e4	1	2.8e7	1.3e7	>3e7	2.0e6
1e−3	7	6.9e5	4.6e5	1.2e6	2.7e5	0	*21e−2*	*11e−2*	*30e−2*	1.6e6
1e−5	2	3.5e6	1.8e6	>7e6	5.0e5	.	.	.	.	.
1e−8	0	*13e−4*	*24e−8*	*11e−3*	4.5e5	.	.	.	.	.

	f_{18} in 5-D, N=15, mFE=502112					**f_{18} in 20-D**, N=15, mFE=2.00e6				
Δf	#	ERT	10%	90%	RT$_{\text{succ}}$	#	ERT	10%	90%	RT$_{\text{succ}}$
10	15	2.3e3	1.8e3	2.7e3	2.3e3	15	4.7e4	4.5e4	5.0e4	4.7e4
1	15	2.2e4	2.1e4	2.3e4	2.2e4	10	1.2e6	7.8e5	1.8e6	7.6e5
1e−1	13	1.3e5	7.9e4	2.0e5	1.2e5	0	*73e−2*	*55e−2*	*20e−1*	1.6e6
1e−3	0	*15e−3*	*25e−4*	*10e−2*	4.5e5	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_{19} in 5-D, N=15, mFE=502112					**f_{19} in 20-D**, N=15, mFE=2.00e6				
Δf	#	ERT	10%	90%	RT$_{\text{succ}}$	#	ERT	10%	90%	RT$_{\text{succ}}$
10	15	3.5e1	2.8e1	4.1e1	3.5e1	15	1.4e4	1.3e4	1.4e4	1.4e4
1	15	1.2e4	1.0e4	1.3e4	1.2e4	14	6.5e5	5.0e5	8.4e5	5.1e5
1e−1	14	1.7e5	1.3e5	2.1e5	1.6e5	0	*44e−2*	*21e−2*	*85e−2*	1.8e6
1e−3	1	7.2e6	3.4e6	>7e6	5.0e5	.	.	.	.	.
1e−5	0	*59e−3*	*24e−3*	*92e−3*	2.8e5	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_{20} in 5-D, N=15, mFE=502112					**f_{20} in 20-D**, N=15, mFE=2.00e6				
Δf	#	ERT	10%	90%	RT$_{\text{succ}}$	#	ERT	10%	90%	RT$_{\text{succ}}$
10	15	7.5e2	5.5e2	9.4e2	7.5e2	15	4.1e4	4.0e4	4.2e4	4.1e4
1	15	1.8e4	1.7e4	1.9e4	1.8e4	15	1.3e5	1.2e5	1.3e5	1.3e5
1e−1	15	3.8e4	3.6e4	4.0e4	3.8e4	0	*32e−2*	*23e−2*	*44e−2*	1.3e6
1e−3	15	6.8e4	6.5e4	7.2e4	6.8e4	.	.	.	.	.
1e−5	14	1.4e5	1.1e5	1.8e5	1.3e5	.	.	.	.	.
1e−8	11	3.6e5	2.8e5	4.8e5	2.7e5	.	.	.	.	.

	f_{21} in 5-D, N=15, mFE=502112					**f_{21} in 20-D**, N=15, mFE=2.00e6				
Δf	#	ERT	10%	90%	RT$_{\text{succ}}$	#	ERT	10%	90%	RT$_{\text{succ}}$
10	15	1.9e2	1.4e2	2.5e2	1.9e2	15	5.0e4	4.7e4	5.4e4	5.0e4
1	15	6.4e3	5.4e3	7.4e3	6.4e3	5	4.1e6	2.5e6	8.1e6	1.2e6
1e−1	13	1.0e5	3.9e4	1.7e5	9.9e4	4	5.6e6	3.2e6	1.3e7	1.1e6
1e−3	13	1.2e5	6.0e4	1.8e5	1.1e5	2	1.3e7	7.1e6	>3e7	2.0e6
1e−5	11	2.4e5	1.6e5	3.3e5	2.2e5	0	*20e−1*	*92e−6*	*47e−1*	1.4e6
1e−8	8	5.3e5	3.7e5	7.8e5	3.0e5	.	.	.	.	.

	f_{22} in 5-D, N=15, mFE=502112					**f_{22} in 20-D**, N=15, mFE=2.00e6				
Δf	#	ERT	10%	90%	RT$_{\text{succ}}$	#	ERT	10%	90%	RT$_{\text{succ}}$
10	15	4.3e2	3.2e2	5.4e2	4.3e2	15	5.1e4	4.6e4	5.6e4	5.1e4
1	15	6.9e3	5.7e3	8.1e3	6.9e3	3	8.1e6	4.9e6	2.4e7	2.0e6
1e−1	9	3.6e5	2.1e5	5.9e5	1.9e5	0	*20e−1*	*69e−2*	*87e−1*	1.0e6
1e−3	4	1.5e6	8.6e5	3.3e6	2.8e5	.	.	.	.	.
1e−5	1	7.1e6	3.3e6	>7e6	5.0e5	.	.	.	.	.
1e−8	0	*24e−3*	*13e−5*	*51e−2*	2.8e5	.	.	.	.	.

	f_{23} in 5-D, N=15, mFE=502112					**f_{23} in 20-D**, N=15, mFE=2.00e6				
Δf	#	ERT	10%	90%	RT$_{\text{succ}}$	#	ERT	10%	90%	RT$_{\text{succ}}$
10	15	4.4e0	3.1e0	6.1e0	4.4e0	15	5.5e0	3.4e0	7.9e0	5.5e0
1	15	3.1e4	2.2e4	4.0e4	3.1e4	4	7.4e6	4.9e6	1.5e7	2.0e6
1e−1	0	*49e−2*	*40e−2*	*63e−2*	2.2e5	0	*12e−1*	*48e−2*	*16e−1*	1.3e6
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

	f_{24} in 5-D, N=15, mFE=502112					**f_{24} in 20-D**, N=15, mFE=2.00e6				
Δf	#	ERT	10%	90%	RT$_{\text{succ}}$	#	ERT	10%	90%	RT$_{\text{succ}}$
10	15	3.4e4	3.1e4	3.8e4	3.4e4	0	*42e+0*	*34e+0*	*65e+0*	1.8e6
1	0	*54e−1*	*52e−1*	*58e−1*	4.5e5	.	.	.	.	.
1e−1	.	.	.	.	.	.	.	.	.	.
1e−3	.	.	.	.	.	.	.	.	.	.
1e−5	.	.	.	.	.	.	.	.	.	.
1e−8	.	.	.	.	.	.	.	.	.	.

Table 1: Shown are, 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 1); 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$_{\text{succ}}$). 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 1 for the names of functions.

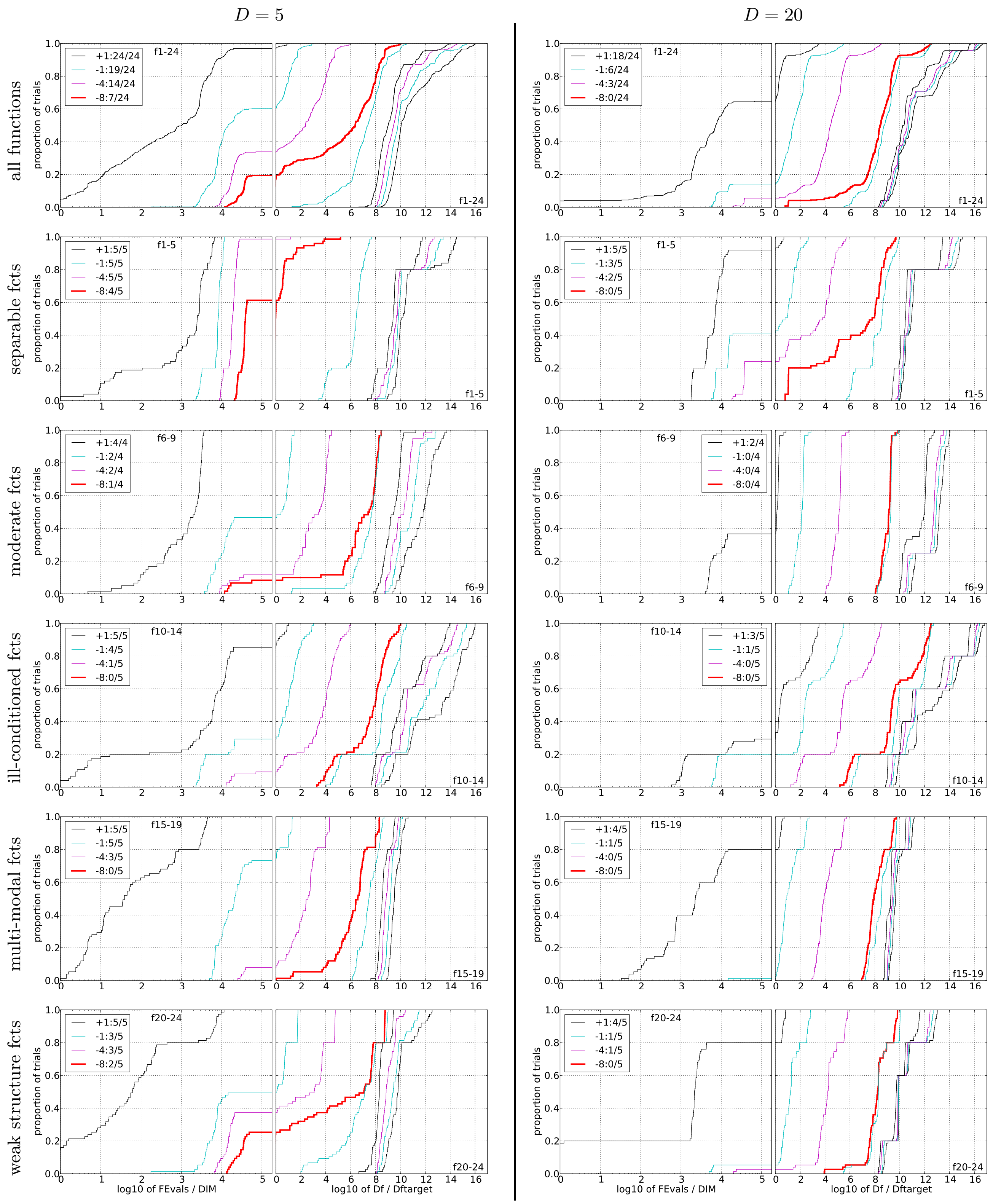

Figure 2: Empirical cumulative distribution functions (ECDFs), plotting the fraction of trials versus running time (left subplots) or versus Δf (right subplots). The thick red line represents the best achieved results. Left subplots: ECDF of the running time (number of function evaluations), divided by search space dimension D, to fall below $f_{\text{opt}} + \Delta f$ 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: separable functions; third row: misc. moderate functions; fourth row: ill-conditioned functions; fifth row: multi-modal functions with adequate structure; last row: multi-modal functions with weak structure. 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.