Comparing Basin Hopping with Differential Evolution and Particle Swarm Optimization

Marco Baioletti[1], Alfredo Milani[1], Valentino Santucci[2], and Marco Tomassini[3]

[1] Department of Mathematics and Computer Science,
University of Perugia, 06123 Perugia, Italy
{marco.baioletti,alfredo.milani}@unipg.it

[2] University for Foreigners of Perugia, 06123 Perugia, Italy
valentino.santucci@unistrapg.it

[3] Faculty of Economics, Department of Information Systems,
University of Lausanne, 1015 Lausanne, Switzerland
marco.tomassini@unil.ch

Abstract. Using a well known benchmarking and profiling environment, we compare the performances of three simple and easy to use metaheuristics for global optimization: Differential Evolution, Basin Hopping and Particle Swarm Optimization. The comparison was done on a test set of 24 functions featuring many characteristics found on real-world problems and on four different space dimensions. Our results statistically show that there is no clear winner overall. The three methods perform well in general and the actual differences are related to the different groups of functions in the benchmark with Basin Hopping being the most robust technique, and Differential Evolution and Particle Swarm Optimization excelling on highly multi-modal functions.

Keywords: Global continuous optimization · Metaheuristics · Benchmarking .

1 Introduction

Global function optimization deals with the mathematical problem of maximising or minimizing a function, possibly subject to some constraints. It is a fundamental technique that is very often needed in many fields of science, technology, and economics. Given a function f, the global minimization problem is to find the minimum value m of f over a domain χ and can be stated as follows:

$$\min_{\mathbf{x}} \{f(\mathbf{x}) \ : \ \mathbf{x} \in \chi\}$$

Usually, one also wants to know the argument, i.e., the point, or set of points, $\mathbf{x}$ that provide the minimum value m of the function:

$$\operatorname*{argmin}_{\mathbf{x}} \{\mathbf{x} \in \chi \ : \ f(\mathbf{x}) = m\}$$

Here $\mathbf{x}$ is a real column vector of scalar variables $[x_1, x_2, \ldots, x_n]^T$, $\forall x_i \in \mathbb{R}$ and $\chi \subseteq \mathbb{R}^n$ is the feasible set to which any solution $\mathbf{x}$ must belong. Maximization is obtained by replacing $f(\mathbf{x})$ with $-f(\mathbf{x})$. In this form, the definition also implicitly covers constrained optimization problems with a proper definition of χ. However, here we will only consider so-called "box constraints" that limit each variable to a segment of the real line, thus restricting the search space to the hyperrectangle $[l_1, u_1]\times, \ldots, \times[l_i, u_j]\times, \ldots, \times[l_n, u_n]$, where l_i and u_i are the lower and upper bounds of variable x_i.

Because of the importance of the problem, many algorithms have been devised to solve it by a variety of techniques, for a good recent presentation see, e.g., [8]. Most of these algorithms are very efficient for convex functions, for which a locally optimal solution is also globally optimal. However, many important applications give rise to problem formulations that are neither linear nor convex. Instead, they are often non-linear, highly multimodal, discontinuous, or even non-differentiable. In particular, it is often the case that no analytical form for the function is known and the function value is provided by a simulation or a measurement, a situation that is aptly called a "black box" and which requires algorithms that only rely on function values. To approach those more general problems, in the last few decades new heuristic methods have been introduced for global function optimization that are, in some sense, inspired by natural phenomena, the most well known being Evolution Strategies (ES) [2], Differential Evolution (DE) [14], Simulated Annealing (SA) [7], and Particle Swarm Optimization (PSO) [6]. Some methods are trajectory-based, such as simulated annealing, while others are population-based. These approaches, often dubbed *metaheuristics*, have variable and sometimes unknown convergence behavior and give no global optimality guarantee, but they all have the valuable common feature of searching the space globally by being potentially able to escape from local optima. Another advantage of metaheuristics for practitioners is that they do not require a large amount of mathematical knowledge and are generally easy to understand and to implement. However, it must be acknowledged that, to get good results, all metaheuristics require some parameters to be tuned, which is usually done by trial and error or by applying some rules of thumb. There also exist other, more mathematically based global optimization methods, either deterministic or stochastic, or both (see, e.g. [9,8]) but they are more difficult to understand, program, and parametrize for the non-specialist. In the present study we shall not consider them further and limit our investigation to metaheuristics; however, a comparison with nature-inspired techniques based on a custom benchmark function set can be found in a recent publication by Sergeyev et al. [12]. Another previous study comparing ES, PSO, Artificial Bee Colony, and the Bees Algorithm has appeared in [11]. However, it uses a different benchmark suite and does not include Basin Hopping.

A less well known trajectory-based heuristic for global optimization is *Basin Hopping* (BH). Basin Hopping has its origins in computational physical chemistry, where it has been successfully used for years to determine minimum energy configurations of atomic clusters and biological macromolecules [15,16]. Although

over the years the technique has tended to become somewhat specialized for the above tasks by taking advantage of known chemical and physical constraints, it is also a very simple but powerful general global optimization method and it is from this point of view that it will be studied here. To our knowledge, Basin Hopping has never been systematically compared to other nature-inspired algorithms before. Thus, our goal in this study is to find out how BH compares to two well established metaheuristics: Differential Evolution and Particle Swarm Optimization. General ES techniques, such as Covariance Matrix Adaptation ES, are known to perform very well on the type of functions of interest here but we did not include them in the comparison because they are more sophisticated and in general require more expertise on the part of the user. Moreover, there are several similar methods in existence and it is difficult to choose a particular one if one is not familiar with them (see, e.g., [2]). As a comparison testbed, we used the widespread BBOB benchmark test suite [5] provided in the *IOHprofiler* environment [4].

The article is structured as follows. In the next section we give an introduction to Basin Hopping, Differential Evolution, and Particle Swarm Optimization, with an emphasis on BH which is the less well known methodology. This is followed by a description of the benchmarking environment, including the function test set. The following sections describes the results obtained and discusses them. Finally, we draw our conclusions.

2 The Metaheuristics Studied

For the sake of completeness, in this section we briefly describe the metaheuristics that have been compared with an emphasis on Basin Hopping which is probably less well known among practitioners.

2.1 Basin Hopping

The outline of the BH algorithm is deceptively simple, see pseudocode 1, where solutions s, x, y, z are to be understood as n-dimensional vectors.

Algorithm 1 Basin Hopping

```
s ← generate initial solution
x ← minimize(f, s)
while termination condition not met do
    y ← perturb(x)
    z ← minimize(f, y)
    x ← acceptance(x, z)
end while
return x, f(x)
```

The algorithm starts by generating an initial solution s either randomly or heuristically. Unless the fitness landscape is flat, this solution must belong to the basin of attraction of some local optimum whose coordinates x are found by using a local search procedure starting at s. After that the algorithm iterates three stages. First, the current solution x is perturbed by some kind of coordinates change yielding the new solution y. Next, starting at y, a local minimizer finds the new local minimum z. There are two possibilities: either z is different from x or it is the same. In the first case, the algorithm has successfully jumped out of the basin of attraction of x. Otherwise, the perturbation has been insufficient and the point y belongs to the original basin of attraction, causing the search to find the same minimum again. Finally, the acceptance phase consists in deciding whether the new solution z is accepted as the starting point of the next cycle. If $f(z) = f(x)$ the search resumes by trying another perturbation from this point. Otherwise, it is accepted either unconditionally or subject to some condition. For example, it could be accepted only if $f(z) < f(x)$. In the original Basin Hopping algorithm the acceptance of the new solution was done conditionally using a Monte Carlo test as in simulated annealing [16]. The new solution z is always accepted if it is better than x. Otherwise, if it is worse than x, it is accepted with probability $e^{-\beta(f_z - f_x)}$, where β is a parameter inversely related to a simulated temperature. As in other metaheuristic approaches the termination condition is met after a predetermined number of iterations, when a given time has elapsed, or when the solution doesn't change within a given precision during a given number of iterations.

So, the three basic components of basin hopping are the minimization algorithm, the perturbation technique, and the acceptance criterion. A good synergy between these components is key for the efficiency of the search. Mathematical minimization techniques for well-behaved functions have been developed over several decades and are efficient and reliable. For this part, in our custom Python implementation of BH, we use a state-of-the art algorithm called "L-BFGS-B" in the SciPy library, which is an extension of the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [13]. The latter approximates the inverse Hessian by using first-derivative, i.e., gradient information. However, given that metaheuristics are often used in a black-box context in which only the function value at a given point is known, it is important to note that BH can also be used for non-continuous or non-differentiable functions since the local search phase can be performed with any working minimization algorithm. For example, one could use the BFGS algorithm above without providing derivatives, which will then approximated by finite differences, or the Nelder-Mead algorithm [10], or any other derivative-free method.

The perturbation technique is difficult to get right in a general way: if perturbations are too small with respect to the typical basin size of the problem at hand then the search will often fall back into the starting basin obviously causing a loss of efficiency. If, on the other hand, jumps are too long then there is the risk that the search degenerates into a random walk in solution space, hardly an efficient technique. On top of this, each particular function has its

own landscape which is in principle unknown a priori unless one samples the function space beforehand or during the search. We have seen above that the acceptance phase can also be implemented in various ways that impose a different intensification/diversification ratio thus influencing the speed of the search and its convergence.

2.2 Differential Evolution

Differential Evolution (DE) is a population-based metaheuristic for function optimization that was introduced by Storn and Price [14]. In the basic DE each individual $\mathbf{x}$ in the population is varied by recombining three randomly chosen but distinct individuals $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ to produce individual $\mathbf{z}$ according to the formula $\mathbf{z} = \mathbf{a} + F(\mathbf{b} - \mathbf{c})$, where F is a weight parameter usually chosen in $[0, 2]$. A random coordinate direction j in the n-dimensional space is then chosen and the new candidate individual $\mathbf{x}'$ is constructed using binary crossover between $\mathbf{x}$ and $\mathbf{z}$, with crossover probability CR as follows:

$$x'_i = \begin{cases} z_i \text{ if } i = j \text{ or with probability } CR \\ x_i \text{ otherwise} \end{cases} \tag{1}$$

Finally, the new solution $\mathbf{x}'$ replaces $\mathbf{x}$ if $f(\mathbf{x}') \leq f(\mathbf{x})$ assuming function minimization. Pseudocode 2 below summarizes the algorithm.

Algorithm 2 Differential Evolution

```
initialize the individuals in the population P with random positions
while termination condition not met do
    for each individual x_i ∈ P do
        choose three different individuals a,b,c ≠ x_i at random
        combine a,b, and c to produce z
        build candidate solution x' by binary crossover between x_i and z
        if f(x') ≤ f(x_i) then
            replace x_i with x' in the population P
        else
            keep solution x_i in the population P
        end if
    end for
end while
return best solution
```

The above pseudocode describes the standard DE algorithm but several more advanced variants have been developed both for the intermediate design of $\mathbf{z}$ as well as for crossover.

2.3 Particle Swarm Optimization

Inspired by animal behavior such as flocks of birds or a swarm of insects, Eberhart and Kennedy [6] proposed an optimization method called *Particle Swarm Optimization*. In this approach, a number of particles simultaneously explore a problem's search space with the goal of finding the globally optimum configuration. Here we describe the canonical version of the algorithm. PSO is a population-based metaheuristic in which the position $\mathbf{x}_i$ of each particle i corresponds to a possible solution to the problem with objective value $f(\mathbf{x}_i)$. In each iteration of the search algorithm the particles move as a function of their velocity $\mathbf{v}_i$ within the specified continuous search space. Two quantities, $\mathbf{x}_i^{best}(t)$ and $\mathbf{B}(t)$, have to be defined and updated in each iteration. The first one, $\mathbf{x}_i^{best}(t)$, which is often called *particle-best*, corresponds to the best fitness point visited by particle i since the beginning of the search. The second quantity, $\mathbf{B}(t)$, called *global-best*, is the best fitness point reached by the population as a whole up to time step t.

The particles' movement in PSO is determined by three contributions. In the first place, there is a term accounting for the "inertia" of the particles: this term tends to keep them on their present trajectory. Second, they are attracted towards $\mathbf{B}(t)$, the global best. And third, they are also attracted towards their best fitness point $\mathbf{x}_i^{best}(t)$. The movement of a particle from one iteration to the next is described by the following equations for particle i's new velocity $\mathbf{v}_i(t+1)$ and new position $\mathbf{x}_i(t+1)$:

$$\begin{aligned}\mathbf{v}_i(t+1) &= \omega\mathbf{v}_i(t) + c_1 r_1(t+1)[\mathbf{x}_i^{best}(t) - \mathbf{x}_i(t)] \\ &\quad + c_2 r_2(t+1)[\mathbf{B}(t) - \mathbf{x}_i(t)] \\ \mathbf{x}_i(t+1) &= \mathbf{x}_i(t) + \mathbf{v}_i(t+1)\end{aligned}$$

where ω, c_1 and c_2 are constants to be specified, and r_1 and r_2 are pseudo-random numbers uniformly distributed in the interval $[0, 1]$. The c_1 parameter reflects the individual's own "perception," and c_2 takes into account the group's behavior. A common choice for these parameters is $c_1 \approx c_2 \approx 2$. The ω parameter is the inertia constant, whose value is in general chosen as being slightly less than one.

In the initialization phase of the algorithm the particles are distributed in a uniform manner in the search domain and are given zero initial velocity. In the algorithm loop, which stops after a given termination condition is met, at each iteration n candidate solutions are generated, one per particle, and the set of solutions is used to construct the next generation according to the dynamical update equations above.

3 The Benchmarking Environment

To meaningfully compare the performances of the algorithms described in the previous section we have used a rich and widespread testing environment called *IOHprofiler* [4] which, besides providing different collections of test functions,

also possesses on-line tools for the visualisation and statistical evaluation of the results[4]. To put this into perspective it is useful to recall some established facts. It has been proved that the performance of any black box optimization method averaged over all possible discrete functions is the same (see the no free lunch theorems [17]). This result has been extended to the continuous scenario, e.g., see [1]. Thus, one might argue that there is no point in comparing algorithms on a given finite and usually small function set because even if an algorithm, say A_1, is more efficient on that set, there will always be other functions on which it is beaten by another algorithm A_2. However, most possible functions are essentially random and do not appear in practical problems. Therefore, it is interesting to benchmark an algorithm on a test set which contains functions that are representative of problems that appear in real-world applications. Still, the results cannot straightforwardly be generalized to other functions but can nevertheless provide useful indications for practitioners.

To this end, we have used the real-parameter optimization benchmarking set called BBOB which comprises 24 noiseless scalable real-parameter single-objective test functions (fully described in [5]). The functions are designed with the goal of exposing the typical difficulties encountered in practice. They include separable and non-separable functions, functions with conditioning, multi-modal functions of various kinds, the role of symmetry and deception. To generate these function features various transformations are applied such as random shifting of the global optimum position, and linear and non-linear transformations of the search space. The functions $\{f_1, f_2, \ldots, f_{24}\}$ are classified into five groups, emphasizing different characteristics as follows:

- separable functions (f_1 to f_5);
- functions with low or moderate conditioning (f_6 to f_9);
- unimodal functions with high conditioning (f_{10} to f_{14});
- multi-modal with adequate global structure (f_{15} to f_{19});
- multi-modal with weak global structure (f_{20} to f_{24}).

The global optima (minimization is assumed) are sought in the search domain defined by the closed compact $[-5, 5]^D$ where D is the space dimension and the typical target value for all functions is $f_{opt} + 10^{-8}$, where f_{opt} is the known optimum of the function at hand and 10^{-8} is the allowed tolerance.

4 Experimental setup

For the tests we used publicly available versions of the DE and PSO algorithm while the basic BH implementation is our own. Furthermore, since all these algorithms have several parameters that have to be chosen in advance, in the following we describe the values used in this study. The DE implementation comes from the `nevergrad`[5] library with its default parametrization, i.e., the crossover rate is

[4] See https://iohprofiler.github.io/
[5] See https://github.com/FacebookResearch/Nevergrad

set to $CR = 0.5$, the mutation operator used is "curr-to-best" with $F = 0.8$ and the population size is 30. The initial population is randomly sampled from the function domain. For the PSO algorithm, we used the RealSpacePSO implementation of `nevergrad`, again with the default parametrization: $\omega = 0.5/\log(2)$, $c_1 = c_2 = 0.5 + \log(2)$ and the population size is 40. It would be useful and interesting to vary some of the important metaheuristic variables to see their influence on the results. However, for this first basic study we couldn't afford the computing time needed to do so. This aspect should certainly be investigated more fully in future work.

Finally, we have implemented the BH algorithm as it was described in Section 2.1 and by considering the standard setting which adopts the sharp acceptance criterion and a random perturbation strength sampled from the interval $[-0.5, +0.5]$.

Each metaheuristic has been executed 15 times on the first 15 instances of each function in the BBOB benchmark, using the values $5, 10, 20, 40$ for the dimension D. Hence, the overall number of runs for each metaheuristic was $15 \times 24 \times 4 \times 15 = 21,600$. Each run had a fixed budget of $200,000$ objective function evaluations to ensure that a fair comparison is performed, regardless of the computation time needed by each metaheuristic.

5 Experimental results

In this section we report and discuss the results of running the three metaheuristics previously described under the *IOHprofiler* environment. To give an overall view of the results we report the scores, i.e. the ratio between the final objective value obtained by an execution and the best objective function value obtained by the three metaheuristics on the instance of the benchmark's test function at hand. In order to improve the visual plotting, we report the logarithm of the score defined as $log(\mathit{fitness}/\mathit{best\_fitness})$, where *best_fitness* is the best objective value found on the current instance in 45 executions, 15 for each algorithm.

In Figure 1 we report a boxplot of the scores of each metaheuristic aggregated on the D value. Results depend slightly on problem dimension but BH seems to be consistently better and shows less variation with respect to DE and PSO over the four dimensionalities tested. Figure 2 again shows aggregated results for the three algorithms, this time by groups of functions in the benchmark as defined in [5] (see also sect. 3 above). The idea here is to try to understand whether there are statistical differences between the approaches according to function group. This appears to be the case for BH, which shows better and more robust performance than either DE and PSO in the the first three groups of functions, i.e. separable functions and functions with low and high conditioning. On the other hand, the results are not dramatically different on the multi-modal functions, with or without a clear global structure.

In the following Figs. 3 and 4 we show the average scores for each function in the test set for $D = 40$, the hardest set of problems. Function numbers and names are drawn from reference [5] which provides all kind of details. It is difficult to

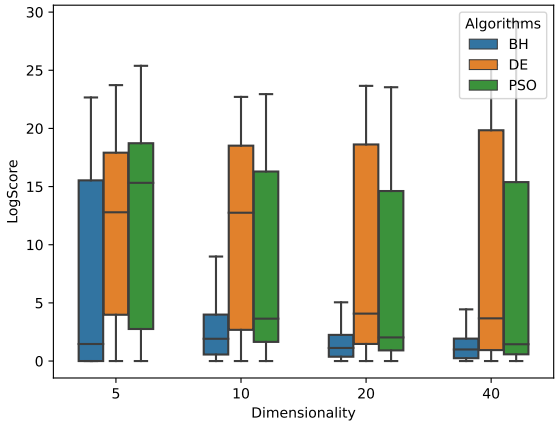

Fig. 1: Scores for BH, DE, and PSO averaged over all test functions for space dimensions $D = 5, 10, 20, 40$. See text for the interpretation of *logScore* on the ordinate axis.

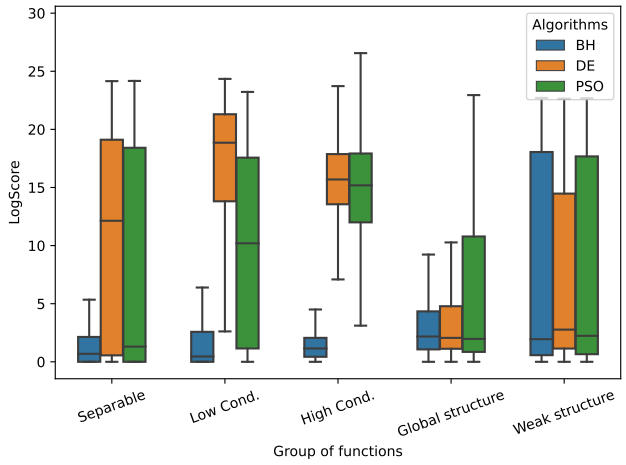

Fig. 2: Scores for BH, DE, and PSO aggregated and averaged according to the function groups as defined in the text.

draw general conclusions from the plots. However, we see that DE seems to have some trouble minimizing the relatively easy f_1 (sphere function), f_2 (ellipsoidal function), and f_5 (linear slope), while highly multimodal functions such as f_3 (Rastrigin) or f_4 (Böuche-Rastrigin) are easily optimized. In the same vein, the functions with conditioning (f_6 to f_{14}) seem also to be harder for DE and, to a lesser extent, for PSO, confirming the aggregated results previously shown in Fig. 2. On all these functions, BH seems to be the more stable method, although not always the best. In the last group of functions (multi-modal with weak global structure), f_{21} (Gallagher's Gaussian 101-me peaks function) is the function that shows the highest fluctuation in the results for all algorithms. This is a difficult highly multimodal function for which PSO obtains by far the best results.

In order to summarize the results in an easier to see and more synthetic way, we refer the reader to Table 1. In the table the first four columns contain a comparison between BH and DE for each of the 24 functions in the test set and for the four space dimensions studied. The second four columns do the same for the pair BH/PSO. For each function a Mann-Whitney test has been performed and a p-value derived [3]. According to the p-value –a standard significance level of 0.05 is considered–, differences can be statistically significant or not. In the last case the corresponding entry in the table is "=". If the differences are statistically significant and BH is better than either DE or PSO a black up-pointing triangle is drawn; otherwise, a white down-pointing triangle is drawn.

From the table, one sees that BH is better overall for the group of unimodal functions with high conditioning (f_{10} to f_{14}) and for the group $f_6 - f_9$. On the other groups results are mixed with DE being statistically better than BH on multimodal functions $f_{15} - f_{19}$ and PSO prevailing on $f_{20} - f_{23}$, while BH is always better on f_{24}. Again, these are only statistical results and more study is needed to really understand the causes for the different behavior of the three metaheuristics on functions having different structure.

To conclude our study, we show here the convergence curves for a couple of functions of dimension $D = 40$. The curves are averaged over all runs for a given function and dimension and report the behavior of the objective value as a function of time, i.e. the number of function evaluations. The shaded areas around the curves show the standard deviations. Figure 5 depicts such a plot for function f_{18} (Shaffer's function F7 moderately ill-conditioned). In agreement with the corresponding entries in Table 1, both DE and PSO converge faster and get better average results than BH. Figure 6 shows the convergence curves for function f_{24} (Lunaceck bi-Rastrigin function) and we now see that BH obtains the best results, also in agreement with Table 1. In both cases BH starts with higher variability but it recovers over time. These two examples are only meant to be illustrative: no general behavior can be drawn at this stage.

6 Conclusions

In this contribution we started from the hypothesis that, when confronted with difficult numerical optimization problems, researchers that are not well versed in

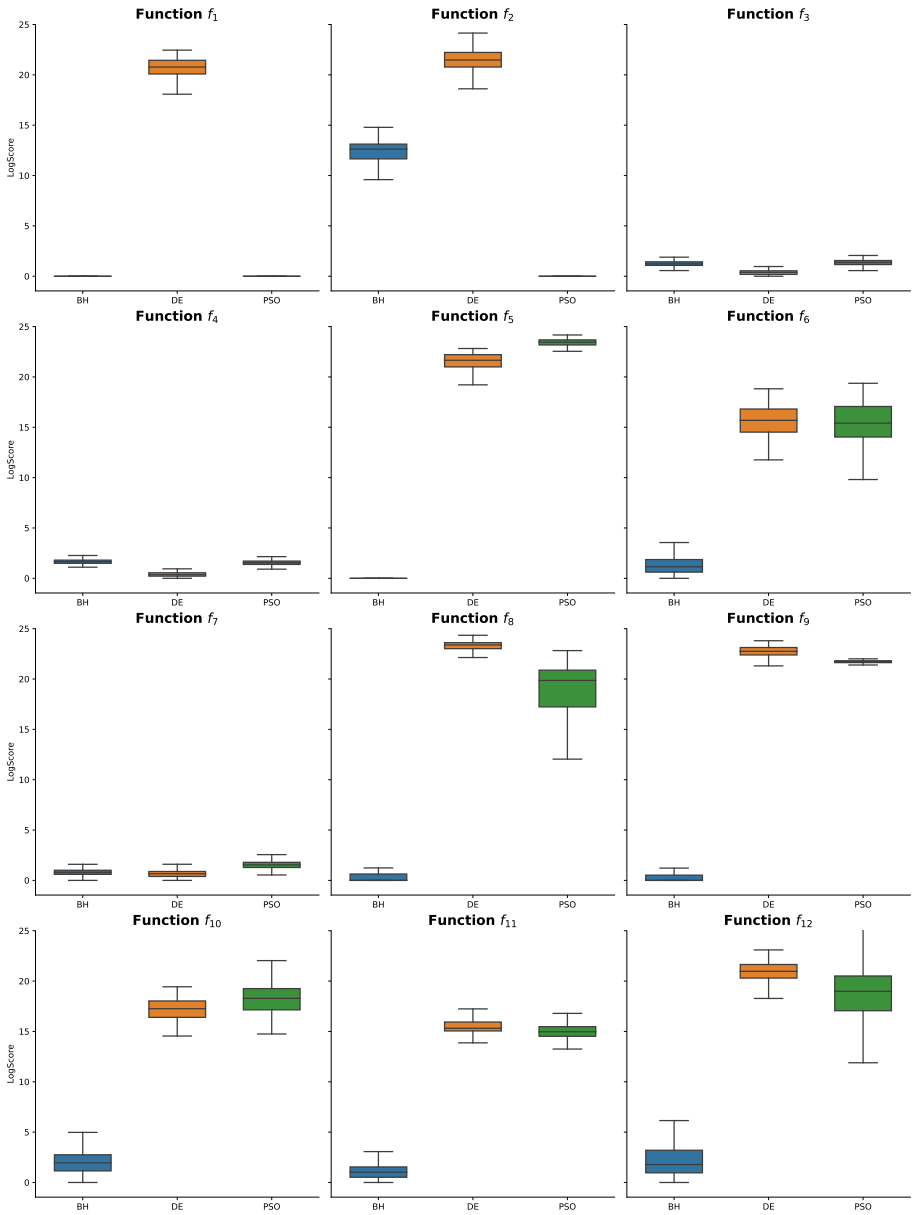

Fig. 3: Box-plots of the log-scores for the three metaheuristics for the functions from f_1 to f_{12} in the benchmark test set for dimension $D = 40$

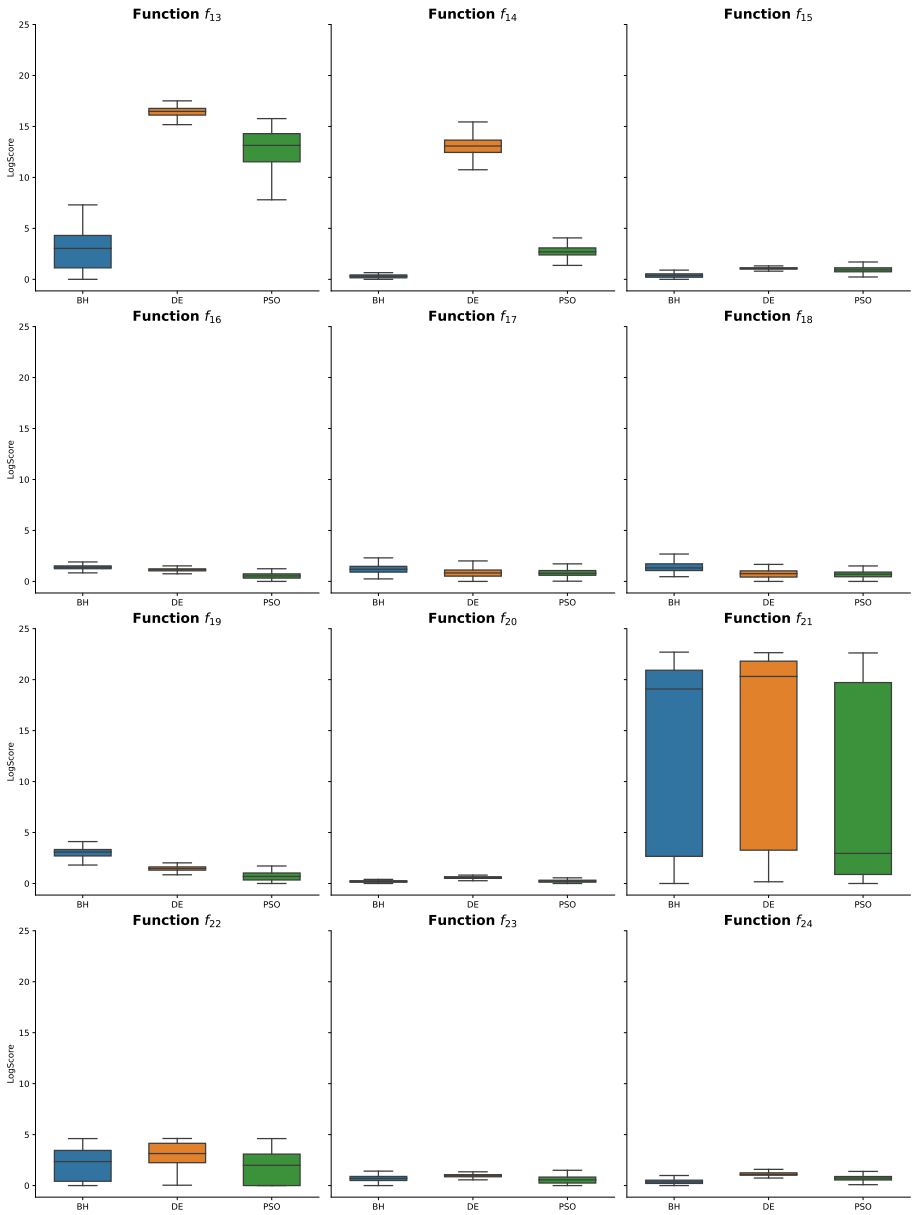

Fig. 4: Box-plots of the log-scores for the three metaheuristics for the functions from f_{13} to f_{24} in the benchmark test set for dimension $D = 40$

Function	BH vs DE				BH vs PSO			
	5D	10D	20D	40D	5D	10D	20D	40D
f_1	▲	▲	▲	▲	=	=	=	=
f_2	▲	▲	▲	▲	=	▽	▽	▽
f_3	=	▽	▽	▽	▲	▲	▲	▲
f_4	▽	▽	▽	▽	▽	▲	▲	▽
f_5	▲	▲	▲	▲	▲	▲	▲	▲
f_6	▲	▲	▲	▲	▽	▽	▲	▲
f_7	▽	=	▲	▽	▲	▲	▲	▲
f_8	▲	▲	▲	▲	▲	▲	▲	▲
f_9	▲	▲	▲	▲	▲	▲	▲	▲
f_{10}	▲	▲	▲	▲	▲	▲	▲	▲
f_{11}	▲	▲	▲	▲	▲	▲	▲	▲
f_{12}	▲	▲	▲	▲	▲	▲	▲	▲
f_{13}	▲	▲	▲	▲	▲	▲	▲	▲
f_{14}	▲	▲	▲	▲	▲	▲	▲	▲
f_{15}	▲	▲	▲	▲	▲	▲	▲	▲
f_{16}	▽	▽	▽	▽	▽	▽	▽	▽
f_{17}	▽	▽	▽	▽	▽	▽	▽	▽
f_{18}	▽	▽	▽	▽	▽	▽	▽	▽
f_{19}	▲	▲	▽	▽	▲	▲	▽	▽
f_{20}	▽	▽	▲	▲	=	▲	▲	▲
f_{21}	▽	▽	=	▲	▽	▽	▽	▽
f_{22}	▽	▽	=	▲	=	▽	▽	▽
f_{23}	▲	▲	▲	▲	=	▽	▽	▽
f_{24}	▲	▲	▲	▲	▲	▲	▲	▲
Wilcoxon	=	▲	▲	▲	▲	=	▲	=

Table 1: Results of a Mann-Whitney test for the significance of the differences between the two pairs of algorithms BH vs DE and BH vs PSO. If BH wins the entry is a black up-pointing triangle. If DE or PSO win the entry is a white down-pointing triangle. The entry "=" means that there is no statistically significant difference.

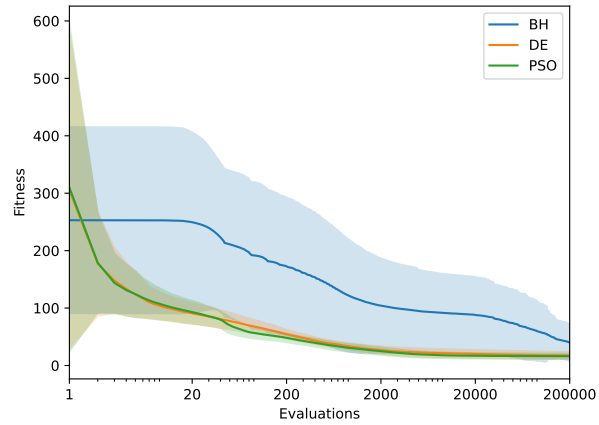

Fig. 5: Convergence curves for function f_{18} and $D = 40$.

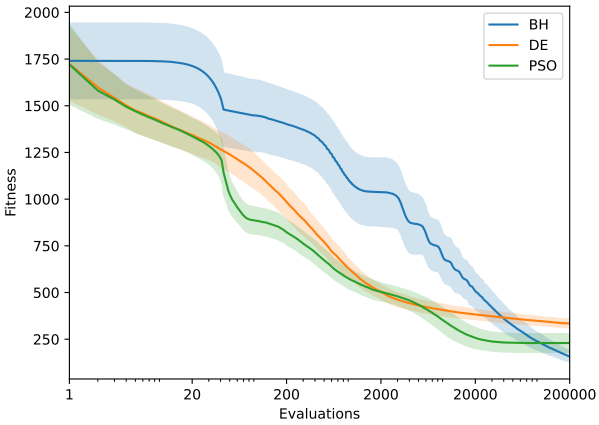

Fig. 6: Convergence curves for function f_{24} and $D = 40$.

specialized mathematical optimization algorithms, may wish to use metaheuristics which do not provide global optimization guarantees but are more intuitive and easy to use. Therefore, among the most well known metaheuristics we singled out PSO, DE, and BH as being widely available and relatively easy to parameterize, and compared them on a standard recognized function test set that is representative of real-world function features commonly encountered in optimization. The picture that emerges from the comparison, using the benchmarking profiler tools and standard statistics is one in which, averaged over all functions tested, no algorithm is really superior to the others. Depending on the group of functions, one or the other offers the best results. Overall, it can be said that BH is probably the more robust of the three metaheuristics, getting consistently good results on most functions while PSO and DE appear well suited for highly multimodal functions in the last two groups. Besides, BH offers the added advantage of being really simple to use and is thus advisable for preliminary pilot studies before embarking on more complex approaches. Future studies include investigating the effect of metaheuristic parameters on the results and the inclusion of some real-world functions arising from important applications such as machine learning.

References

1. Alabert, A., Berti, A., Caballero, R., Ferrante, M.: No-free-lunch theorems in the continuum. Theoretical Computer Science **600**, 98–106 (2015)
2. Bäck, T., Foussette, C., Krause, P.: Contemporary evolution strategies. Springer (2013)
3. Derrac, J., García, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation **1**(1), 3–18 (2011)
4. Doerr, C., Wang, H., Ye, F., van Rijn, S., Bäck, T.: IOHprofiler: A benchmarking and profiling tool for iterative optimization heuristics. arXiv preprint arXiv:1810.05281 (2018)
5. Hansen, N., Finck, S., Ros, R., Auger, A.: Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Ph.D. thesis, INRIA (2009)
6. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95-International Conference on Neural Networks. vol. 4, pp. 1942–1948. IEEE (1995)
7. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. science **220**(4598), 671–680 (1983)
8. Kochenderfer, M.J., Wheeler, T.A.: Algorithms for optimization. MIT Press (2019)
9. Liberti, L.: Introduction to global optimization. Ecole Polytechnique (2008)
10. Nelder, J.A., Mead, R.: A simplex method for function minimization. The computer journal **7**(4), 308–313 (1965)
11. Pham, D.T., Castellani, M.: Benchmarking and comparison of nature-inspired population-based continuous optimisation algorithms. Soft Computing **18**(5), 871–903 (2014)

12. Sergeyev, Y.D., Kvasov, D.E., Mukhametzhanov, M.S.: On the efficiency of nature-inspired metaheuristics in expensive global optimization with limited budget. Scientific reports **8**(1), 1–9 (2018)
13. Shanno, D.F.: On Broyden-Fletcher-Goldfarb-Shanno method. Journal of Optimization Theory and Applications, **46**, 87–94 (1985)
14. Storn, R., Price, K.: Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization **11**(4), 341–359 (1997)
15. Wales, D.J., Doye, J.P.: Global optimization by basin-hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms. The Journal of Physical Chemistry A **101**(28), 5111–5116 (1997)
16. Wales, D.J., Scheraga, H.A.: Global optimization of clusters, crystals, and biomolecules. Science **285**(5432), 1368–1372 (1999)
17. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE transactions on evolutionary computation **1**(1), 67–82 (1997)