INFORMATICA, 2015, Vol. 26, No. 4, 649–662

DOI: http://dx.doi.org/10.15388/Informatica.2015.69

On Benchmarking Stochastic Global Optimization Algorithms

Eligius M.T. HENDRIX[1], Algirdas LANČINSKAS[2]*
[1]*Computer Architecture, Universidad de Málaga, Spain*
[2]*Institute of Mathematics and Informatics, Vilnius University, Lithuania*
e-mail: eligius@uma.es, algirdas.lancinskas@mii.vu.lt

Received: March 2015; accepted: June 2015

Abstract. A multitude of heuristic stochastic optimization algorithms have been described in literature to obtain good solutions of the box-constrained global optimization problem often with a limit on the number of used function evaluations. In the larger question of which algorithms behave well on which type of instances, our focus is here on the benchmarking of the behavior of algorithms by applying experiments on test instances. We argue that a good minimum performance benchmark is due to pure random search; i.e. algorithms should do better. We introduce the concept of the cumulative distribution function of the record value as a measure with the benchmark of pure random search and the idea of algorithms being dominated by others. The concepts are illustrated using frequently used algorithms.

Key words: stochastic global optimization, benchmark, black-box, meta-heuristic.

1. Problem Description

We consider the box-constrained global optimization problem

$$f^* = \min_{x \in X} f(x), \tag{1}$$

where $f(x)$ is a continuous function and $X \subset \mathbb{R}^n$ is a box constrained feasible region. The idea of black-box optimization is that function evaluations imply running an external (black-box) routine that may take minutes or hours to provide the evaluated objective function value. In engineering applications, often the question is how to a good, but not necessarily optimal solution within a day, several days, or a week. In terms of global optimization the question is formulated as how to obtain a good solution within a limited number (a budget) of function evaluations. We focus on the case, where a stochastic algorithm is run on an instance of (1) up to a budget N of function evaluations has been used.

Many stochastic heuristic algorithms for generating good solutions for such a problem have been described in literature; see e.g. Hendrix and Tóth (2010) for the overview of that

*Corresponding author.

algorithms. Although concepts of simulated annealing and population algorithms already existed for a long time, many algorithms have been developed under the terminology of evolutionary algorithms or meta-heuristics after the appearance of the work of Holland (1975) on genetic algorithms. Mathematical statistical analysis of the speed of convergence is difficult for complicated algorithm descriptions. Therefore, researchers rely on numerical tests with a set of test problems that have been evolved in books and on the Internet after the first set described by Dixon and Szegö (1975).

The ultimate question is which types of algorithms perform well on which type of instances; what defines the characteristics of the case to be solved such that one algorithm is more successful than the other? This question requires to investigate for which instances a specific algorithm does not perform well compared to simple benchmarks. In most published numerical results of algorithms, systematic investigation of worst case behavior is lacking. Insight is necessary for in the end advising for a given problem (and its characteristics), which algorithm is most appropriate to solve it.

We argue that there is a tendency in literature to focus on the average behavior of algorithms rather than to investigate in a systematic way the variation in their behavior or worst case performance. We provide some papers that introduced new algorithms and that are cited in literature, where the report on the performance focuses on average behavior. Chelouah and Siarry (2000) discuss a Genetic Algorithm and do numerical tests measuring only average number of function evaluations. Jelasity *et al.* (2001) illustrate performance with average record values for a budget N. Recchioni and Scoccia (2000) introduce a heuristic for constrained optimization and measure an average number of gradient evaluations and an average number of iterations. İlker Birbil and Fang (2003) discuss average number of function evaluations against the average reached function values for a new electromagnetism-like mechanism. Redondo *et al.* (2012) introduced an evolutionary algorithm for multidimensional scaling and illustrate its performance with average computing time.

The so-called performance profiles introduced by Dolan and Moré (2002) for deterministic methods make sense when comparing deterministic global optimization algorithms, like in the recent study of Misener and Floudas (2014) on MINLP software. The concept was propagated for analyzing stochastic methods by Ali *et al.* (2005), which lead to more focus on average values, although that paper as such also discussed more sophisticated measures. An example of (average based) performance profiles is due to the paper on particle swarm (Vaz and Vicente, 2007). The concept becomes even more complicated when mixing concepts of stochastic based algorithms with deterministic heuristics. An example is the paper of Müller and Schoemaker (2014) where random sampling is combined with radial basis function heuristic. For a large comparison of deterministic heuristics versus stochastic ones see the work of Rios and Sahinidis (2013) combining many performance measures. Our argument is that stochastic methods exhibit variation, which should be taken into account in benchmarking.

The research question of this paper is how to evaluate the quality of an algorithm for an individual test case. We argue that the performance of Pure Random Search (PRS) can be taken as benchmark to measure how much better (or worse) an algorithm performs.

This paper is organized as follows. Section 2 describes the concept of the cumulative density function for the best point found and the concept of domination of one algorithm by another. Section 3 illustrates the new concept on well-known test cases and several frequently used algorithms. Section 4 summarizes our findings.

2. Cumulative Density of the Best Point Found

In general, a stochastic optimization algorithm generates a series of points x_k that approximate an (or the, or all) minimum point(s). According to the generic description of Törn and Žilinskas (1989):

$$\boldsymbol{x}_{k+1} = \mathrm{Alg}(\boldsymbol{x}_k, \boldsymbol{x}_{k-1}, \ldots, \boldsymbol{x}_1, \boldsymbol{\xi}), \tag{2}$$

where $\boldsymbol{\xi}$ is a random variable and index k is the iteration counter. In this paper, random variables are denoted by boldface symbols whereas non-bold symbols stand for regular variables. Description (2) represents the idea that a next point $\boldsymbol{x}_{k+1}$ is generated based on the information in all former points $\boldsymbol{x}_k, \boldsymbol{x}_{k-1}, \ldots, \boldsymbol{x}_1$ ($\boldsymbol{x}_1$ usually being the starting point) and a random effect $\boldsymbol{\xi}$ based on generated pseudo-random numbers. The final result of running an algorithm with N function evaluations on a test function is the random record function value $\boldsymbol{Y}_N = \min_{k=1,\ldots,N} f(\boldsymbol{x}_k)$. The quality of an algorithm A with N trials is defined by the cumulative distribution function of the record $\boldsymbol{Y}_N$ denoted by $CDFR_N^{[\mathrm{A}]}(y) = P\{\boldsymbol{Y}_N \leqslant y\}$. This measure is three dimensional when we consider the probability, the level y and the budget on function evaluations N and therefore hard to capture in an analysis. If one only focuses on the expected value $E(\boldsymbol{Y}_N)$ as function of the budget N estimated by a numerical average, the variation in the result is ignored; for some run (repetition), an algorithm may fail and for another not.

In order to understand the concept, let us first consider the starting point of stochastic global optimization algorithms of sampling one trial point $\boldsymbol{x}$ uniformly drawn over the feasible region. Consider $\mu(y) = P\{f(\boldsymbol{x}) \leqslant y\}$ being the cumulative distribution function of random variable $\boldsymbol{y} = f(\boldsymbol{x})$, where $\boldsymbol{x}$ is uniform over X. So, basically $CDFR_1^{[\mathrm{PRS}]}(y)$ is the function $\mu(y)$ with domain $[f^*, \max_X f(\boldsymbol{x})]$ and characterizes completely $CDFR_N^{[\mathrm{PRS}]}(y)$ on a test instance; the probability that a level y is reached after generating N trial points is given by $P_N(y) = 1 - (1 - \mu(y))^N$.

$P_N(y)$ provides a benchmark for all stochastic algorithms. For an algorithm A, the $CDFR_N^{[\mathrm{A}]}(y)$ for function value y should at least reach the probability $P_N(y)$, i.e. what is the difference between $CDFR_N^{[\mathrm{A}]}(y)$ and $P_N(y) = 1 - (1 - \mu(y))^N$ after having generated N points?

For example, the distribution function $P_1(y) = \mu(y)$ can be approximated numerically. We illustrate this for the Six-hump camel-back function

$$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4 \tag{3}$$

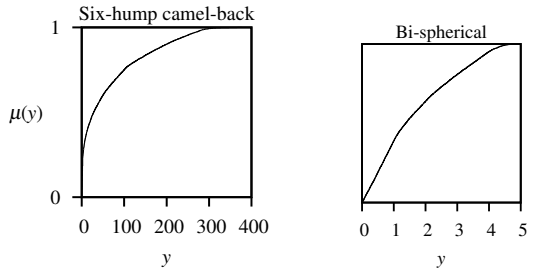

Fig. 1. Approximation of cumulative distribution function $P_1(y) = \mu(y)$ via 10 000 samples.

over $X = [-3, 3]^2$ and the bi-spherical functions (Hendrix and Tóth, 2010)

$$f(x) = \min\left\{(x_1 - 1)^2, (x_1 + 1)^2 + 0.01\right\} + \sum_{2}^{n} x_i^2 \tag{4}$$

over $[-2, 2]^2$ based on 10 000 samples. In Fig. 1 one can observe that it is relatively easy to obtain points below the level $y = 50$, corresponding to 12.5% of the value range, for the Six-hump camel-back test problem. A similar $\mu(y)$ probability for the Bi-spherical problem reaches only 40% of the value range.

Figure 1 also shows that the values of y could be scaled towards $[0, 1]$ if we would like to compare the characteristic function $P_N(y)$ over different test cases. Theoretically the value range is $[\min_X f(\boldsymbol{x}), \max_X f(\boldsymbol{x})]$. For experiments on a given test instance the minimum value is known, but the maximum value is less relevant. One can base the right end on the reached values, or alternatively on extreme order statistics considerations as discussed in Zhigljavsky and Žilinskas (2008).

Another important algorithm in stochastic optimization is Multistart (MS) (Baritompa and Hendrix, 2005); see also Algorithm 2. It requires a local (nonlinear) optimization routine $\mathrm{LS}(s, N)$ as a procedure which given a starting point s and limit N on the number of function evaluations returns a point in the domain that approximates a local minimum point. In contrast to PRS, numerical results therefore depend on the LS routine applied using N_{LS} evaluations for a local search. If the budget N is relatively low compared to the full local search cost N_{LS}, spending function evaluations on the final steps of local search makes less sense and one better adapts Algorithm 2. Let us consider what exactly happens when running the algorithm on an instance.

The function $CDFR_N^{[\mathrm{MS}]}$ can theoretically be generated by studying the characteristics of the test case in conjunction with the local routine. It consists of a set of (local) minimum values and their relative volume of the region of attraction. If N_{LS} is independent of the starting point and N is a multiple of N_{LS}, $CDFR_N^{[\mathrm{MS}]}$ follows from a multinomial distribution. An early study in literature on these characteristics is due to Zielinski (1981) focusing on estimating the volumes and values of the local minima. The consequence is that $CDFR_N^{[\mathrm{MS}]}$ has a typical step shape of the discrete distribution where the objective values of the local minima have a certain probability mass. Notice that in the practice these assumptions may not apply, i.e. $\frac{N}{N_{LS}}$ may be low given the budget N. In that case, the last local search may reach the budget N of function evaluations before convergence, so

Algorithm 1 PRS

procedure PRS(X, f, N)
  $F \leftarrow \infty$;
  for $k = 1$ **to** N **do**
    Generate uniform $\boldsymbol{x}_k$ over X;
    if $f(\boldsymbol{x}_k) < F$ **then** $F \leftarrow f(\boldsymbol{x}_k)$
  end for
  return F and $\text{argmin}_k f(\boldsymbol{x}_k)$;
end procedure

Algorithm 2 MS

procedure MS(X, f, N, LS)
  $F \leftarrow \infty$; $k \leftarrow 1$;
  while $N > 0$ **do**
    Generate $\boldsymbol{s}$ uniformly over X;
    $[\boldsymbol{x}, N_{\text{LS}}] \leftarrow \text{LS}(\boldsymbol{s}, N)$;
    if $f(\boldsymbol{x}) < F$ **then** $F \leftarrow \boldsymbol{x}_k$;
    $N \leftarrow N - N_{\text{LS}}$; $k \leftarrow k + 1$;
  end while;
  return F and $\text{argmin}_k f(\boldsymbol{x}_k)$;
end procedure

$CDFR_N^{[\text{MS}]}$ may also reveal a probability on values other than the local minima. One could argue beforehand, that for small budgets compared to the number of optima, one should adapt the search and perform less local searches, as discussed in Hendrix and Roosma (1996). However, it is not our objective to come to new schemes here, but to observe its exact behavior on test cases.

Our idea is to have a measure for comparison of two algorithms A and B, to see whether one of them is performing better on a certain problem instance. It may be clear that algorithm A is doing better than algorithm B on an instance for effort N if $\forall y$, $CDFR_N^{[\text{A}]}(y) > CDFR_N^{[\text{B}]}(y)$.

Using concepts like the Performance profile, designed to compare deterministic algorithms and promoted by Ali *et al.* (2005) for stochastic algorithms, stimulates a focus on the average behavior to determine whether $E(\boldsymbol{Y}_N^{[\text{A}]}) > E(\boldsymbol{Y}_N^{[\text{B}]})$. This is typical a necessary but not sufficient condition to determine the better performance. If test cases (instances) can be classified into problem classes, the most interesting question is whether the behavior of a particular algorithm B is dominated by that of another algorithm A. It means one can take B out of consideration to solve problems from this class. In order to investigate this, one should strictly define the concept of domination.

Definition 1. Let A and B be two stochastic algorithms run on an instance F of problem (1). Then A is said to dominate B on F if $\forall y, N$, $CDFR_N^{[\text{A}]}(y) \geqslant CDFR_N^{[\text{B}]}(y)$ and $\exists y, N$ such that $CDFR_N^{[\text{A}]}(y) > CDFR_N^{[\text{B}]}(y)$.

Notice that for population-based algorithms that initially start with a randomly generated and evaluated population, the behavior of the record value is exactly the same as that of PRS till the population is generated and the mechanism of "reproduction" (i.e. generating new trial points on the base of the current population) is started. Very low budgets N of function evaluations are therefore less interesting. On the other hand, for a high budget N in the lower dimensional cases typically used in literature, the difference between the methods diminishes. We stress this, because we observed tables in literature where two-dimensional instances were hit with tens of thousands of trial points. Ali *et al.* (2005) suggested to compare algorithms for $N \in \{10n, 10n^2, 100n^2\}$ evaluations. That does not necessarily reflect a budget for practical black box design problems. In the sequel, we will attempt to find an interesting region of budget N and test cases where well known algorithms can be distinguished.

3. Numerical Illustration of the New Concepts

3.1. *Description of Algorithms*

We generated the *CDRF* curves for many test cases that are widely available in literature and on the web. For the illustration we selected several popular instances that show an interesting difference for several frequently used population algorithms. For the choice of the maximum effort N, we take the idea that a function evaluation takes 5 minutes and one would like to have an answer in one or several days. For the illustration we consider $N = 200, 500, 1000$. Due to the popularity of evolutionary algorithms, we will use several easily accessible codes of population based algorithms as well as the basic Multistart of Algorithm 2 with the local search fmincon of MATLAB and confront their performance with the benchmark of PRS.

Price (1979) introduced a population-based algorithm called Controlled Random Search (CRS) that has been widely used and modified into many variants by himself and other researchers. Algorithm 3 describes the initial scheme modified for a budget N. No-

Algorithm 3 CRS

procedure CRS(X, f, N, M)
  Generate and evaluate a set $\boldsymbol{P}$ of M random points uniformly over X;
  $k \leftarrow M$;
  while $k < N$ **do**
    Select at random a subset $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_{n+1}\} \subset \boldsymbol{P}$;
    $\boldsymbol{x}_k \leftarrow \frac{2}{n}\sum_{i=1}^{n} \boldsymbol{p}_i - \boldsymbol{p}_{n+1}$;
    if $\boldsymbol{x}_k \in X$ **and** $f(\boldsymbol{x}_k) < \max_{\boldsymbol{p}\in\boldsymbol{P}} f(\boldsymbol{p})$ **then** replace $\operatorname{argmax}_{\boldsymbol{p}\in\boldsymbol{P}} f(\boldsymbol{p})$ by $\boldsymbol{x}_k$;
    $k \leftarrow k + 1$;
  end while
  return $\operatorname{argmin}_{\boldsymbol{p}\in\boldsymbol{P}} f(\boldsymbol{p})$ and $\min_{\boldsymbol{p}\in\boldsymbol{P}} f(\boldsymbol{p})$;
end procedure

Algorithm 4 GA

procedure GA(X, f, N, M)
  Generate and evaluate a set $\boldsymbol{P}$ of M random points uniformly over X;
  $k \leftarrow M$;
  while $k < N$ **do**
    $\boldsymbol{Q} = \emptyset$;
    for $i = 1$ **to** M **do**
      Select at random $\{\boldsymbol{p}_1, \boldsymbol{p}_2\} \subset \boldsymbol{P}$;
      Generate $\boldsymbol{q}$ by crossing $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$.
      Mutate $\boldsymbol{q}$ by changing some of its coordinates;
      $\boldsymbol{Q} \leftarrow \boldsymbol{Q} \cup \{\boldsymbol{q}\}$;
    end for
    Evaluate fitness of the generated points $\boldsymbol{q} \in \boldsymbol{Q}$;
    $k \leftarrow k + M$;
    $\boldsymbol{P} \leftarrow M$ best points from $\boldsymbol{P} \cup \boldsymbol{Q}$;
  end while
  return $\operatorname{argmin}_{\boldsymbol{p} \in \boldsymbol{P}} f(\boldsymbol{p})$;
end procedure

tice that apart from the population size M, no additional parameter is required in this basic description.

Although the concept of evolutionary algorithms already existed before, they became extremely popular after the appearance of the works of Holland (1975) and Davis (1991). Evolutionary terminology such as individuals, offspring, alleles, genome, crossover, survival of the fittest, etc., is usually mixed with the pure algorithm description. The algorithms usually suffer from an overshot of parameters to steer the search process; for instance the GA function in MATLAB has 26 parameters of which about 17 influence the performance of the algorithm, the others refer to output. For the illustration we use the MATLAB implementation with its default values for the parameters.

Kennedy and Eberhart (1995) came up with an algorithm where evolutionary terminology was replaced by "swarm intelligence" and "cognitive consistency". In each iteration of the algorithm, each member (particle) of the population, called swarm, is modified and evaluated. Classical nonlinear programming modification by direction and step size is now termed "velocity". Instead of considering $\boldsymbol{P}$ as a set, one better thinks of an ordered list of elements, $\boldsymbol{p}_i$, $i = 1, \ldots, M$. Besides its current position $\boldsymbol{p}_i$, also the best position $\boldsymbol{b}_i$ found by particle i is stored. Each particle i has a velocity $\boldsymbol{v}_i$ that is updated at each iteration containing random effects. The velocity $\boldsymbol{v}_i$ is based on the current position $\boldsymbol{p}_i$, best found position $\boldsymbol{b}_i$, and the global best point $\boldsymbol{x} = \min_i \boldsymbol{b}_i$, found so far by the whole swarm. The position $\boldsymbol{p}_i$ is updated by adding the velocity: $\boldsymbol{p}_i \leftarrow \boldsymbol{p}_i + \boldsymbol{v}_i$. For the description it is useful to use the element index j besides the particle index i and iteration index k. See Algorithm 5 for the pseudo-code of PSO.

Algorithm 5 PSO

procedure PSO(X, f, N, M, ω, c_1, c_2)
  Generate and evaluate a set $\boldsymbol{P}$, of M random points uniformly over X;
  $\boldsymbol{x} \leftarrow \text{argmin}_{\boldsymbol{p}\in\boldsymbol{P}} f(\boldsymbol{p})$; $\boldsymbol{b}_i \leftarrow \boldsymbol{p}_i$ and $\boldsymbol{v}_i \leftarrow 0$, $i = 1, 2, \ldots, M$;
  $k \leftarrow M$;
  while $k < N$ **do**
    for $i = 1$ **to** M **do**
      for $j = 1$ **to** n **do**
        Generate r uniformly over $[0, 1]^2$;
        $v_{ij} \leftarrow \omega v_{ij} + 2c_1 r_1 (b_{ij} - p_{ij}) + 2c_2 r_2 (x_{bj} - p_{ij})$;
      end for
      $\boldsymbol{p}_i \leftarrow \boldsymbol{p}_i + \boldsymbol{v}_i$;
      if $f(\boldsymbol{p}_i) < f(\boldsymbol{b}_i)$ **then** $\boldsymbol{b}_i \leftarrow \boldsymbol{p}_i$;
      if $f(\boldsymbol{p}_i) < f(\boldsymbol{x})$ **then** $\boldsymbol{x} \leftarrow \boldsymbol{p}_i$;
    end for
    $k \leftarrow k + M$;
  end while
  return $f(\boldsymbol{x})$ and $\boldsymbol{x}$;
end procedure

3.2. *Results and Discussion*

In order to illustrate the concepts of the cumulative distribution $CDFR_N(y)$ of the record, we elaborate numerical results for $N = 200, 500, 1000$ over several test instances for the three population algorithms described in Section 3.1 with a population size of $M = 50$ and the basic Multistart and confront them with the benchmark of PRS. The curves were generated by repeating each algorithm 1000 times. The x-axis of the graphs is scaled by the maximum objective function value of PRS over 1000 repetitions for N function evaluations.

The Six-hump camel-back test problem is a popular test case in 2 dimensions. There are 6 local minima points in the feasible region, of which two are global minimum points. Figure 2 shows the $CDFR_N^{[\mathrm{A}]}(y)$ curves for each algorithm A and each budget $N = 200, 500, 1000$. It is interesting to see that visually for $N = 1000$ the scale is such that the quality of the population algorithms cannot be distinguished. Results can be better distinguished for the small budget $N = 200$. For this budget, Multistart can perform only 5 local searches with the MATLAB local search solver, but this provides still more than 99% chance to reach a global minimum point. Thinking in terms of "generations", PSO and GA only refresh their population (swarm) four times. Nevertheless, the GA algorithm dominates the other population algorithms, i.e. its curves are higher for all tested budgets N. None of the population algorithms is worse than PRS.

The Bi-spherical test problem has 2 local minimum points one of which is the global one. Independent of the dimension n, the two regions of attraction are about 50% of the total volume facilitating analysis. For the numerical generation we used the feasible area

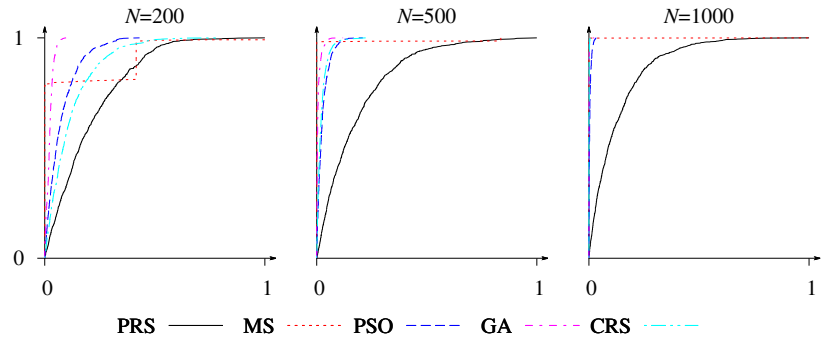

Fig. 2. Plots of $CDFR_N^{[A]}(y)$ by running algorithms $A = $ PRS, MS, PSO, GA, CRS on the Six-hump camel-back test problem, $N = 200$ (left); $N = 500$ (middle); $N = 1000$ (right).

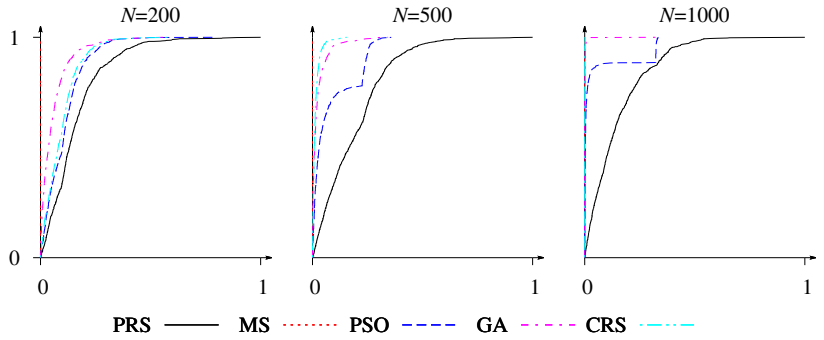

Fig. 3. Plots of $CDFR_N^{[A]}(y)$ running algorithms $A = $ PRS, MS, PSO, GA, CRS on the Bi-spherical test case, dimension $n = 2$, $N = 200$ (left); $N = 500$ (middle); $N = 1000$ (right).

$[-2, 2]^2$ and the higher dimensional $[-2, 2]^{10}$. Figure 3 shows the curves derived for $n = 2$ and Fig. 4 – for $n = 10$, which is considered much more difficult due to the dimensionality. As the region of attraction of the global minimum consists of practically 50% of the search space, the curve of MS coincides with the vertical axis independent on the number of function evaluations. For $n = 2$, the refractive point of the PSO curve with $N = 500, 1000$ function evaluations indicates that the algorithm is sensitive to the attraction of the local (non-global) minimum point.

For higher dimensions, like $n = 10$, usage of budgets as $N = 200, 500, 1000$ is perceived as hopeless. Notice that for this specific instance, Multistart is doing still excellent and dominates all population algorithms. It practically always reaches the global minimum point.

The Ackley test problem (Ackley, 1987) is another interesting instance that facilitates observation of results in several dimensions having practically the same landscape. Besides the unique global minimum point, the instance has a large number of local minima. Therefore, it is often used to test the behavior of algorithms on the existence of many optima. Figure 5 shows the derived curves of $CDFR_N^{[A]}(y)$ for the algorithms for $n = 2$ and Fig. 6 for the high dimensional case $n = 10$. An interesting characteristic of this in-

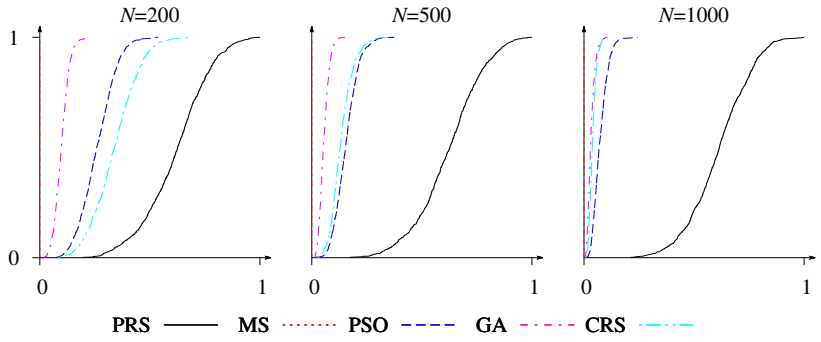

Fig. 4. Plots of $CDFR_N^{[A]}(y)$ running algorithms $A = $ PRS, MS, PSO, GA, CRS on the Bi-spherical test case, dimension $n = 10$, $N = 200$ (left); $N = 500$ (middle); $N = 1000$ (right).

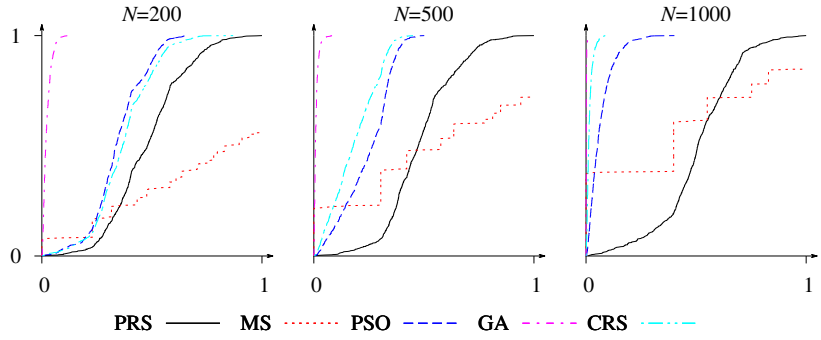

Fig. 5. Plots of $CDFR_N^{[A]}(y)$ obtained by running algorithms $A = $ PRS, MS, PSO, GA, CRS on the Ackley test problem, $n = 2$, $N = 200$ (left); $N = 500$ (middle); $N = 1000$ (right).

stance is that MS is not more effective than PRS due to the large number of local minima. For $n = 2$, it still has a positive probability of reaching the global optimum, but in higher dimension the use of local minimization on this budget looks hopeless. The population algorithms here are also doing better than PRS and again GA seems to dominate the other two population algorithms: CRS and PSO.

Another extreme (difficult) case is the Easom test problem (Easom, 1990) in 2-dimensional space. There is one global minimum point characterized by a narrow and deep indentation in a nearly horizontal plane. This means that any local search gets stuck on the plane, but population-based algorithms are able to start moving after one of its members found the hole. Figure 7 shows the derived curves of $CDFR_N^{[A]}(y)$ for each algorithm A. One can see in the figure that the algorithms behave equally poor on this instance when using $N = 200$ function evaluations. The GA algorithm at least reaches some probability to come close to the lower valley. When the budget of function evaluations is $N = 500$, the performance of PSO can be distinguished as well, though PSO is much less effective than GA. Finally, when the effort really increases towards $N = 1000$, all population algorithms seem to do better than PRS and MS. For this characteristic, local searches appear very bad.

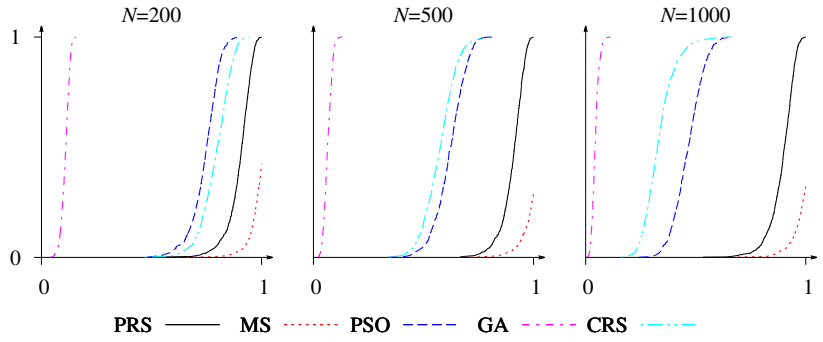

Fig. 6. Plots of $CDFR_N^{[A]}(y)$ obtained by running algorithms $A = $ PRS, MS, PSO, GA, CRS on the Ackley test problem, $n = 10$, $N = 200$ (left); $N = 500$ (middle); $N = 1000$ (right).

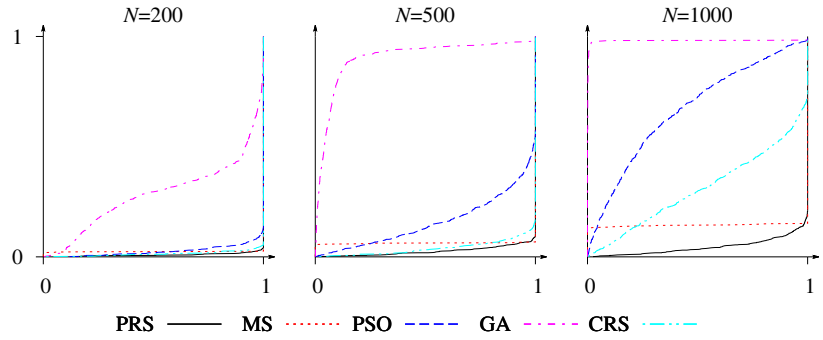

Fig. 7. Plots of $CDFR_N^{[A]}(y)$ obtained by running algorithms $A = $ PRS, MS, PSO, GA, CRS on the Easom test problem, $N = 200$ (left); $N = 500$ (middle); $N = 1000$ (right).

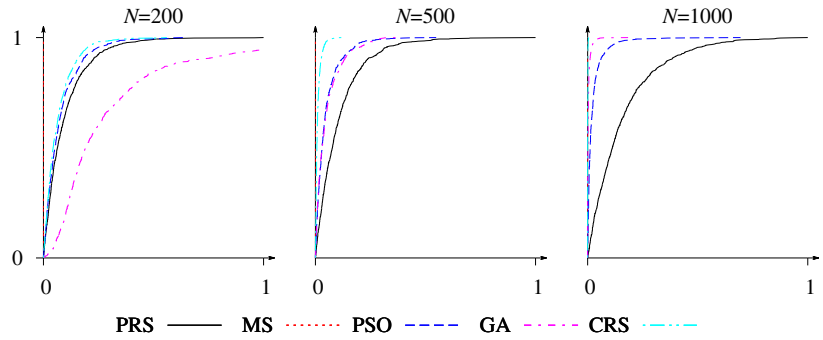

Fig. 8. Plots of $CDFR_N^{[A]}(y)$ obtained by running algorithms $A = $ PRS, MS, PSO, GA, CRS on the Branin test problem, $N = 200$ (left); $N = 500$ (middle); $N = 1000$ (right).

The Branin (Dixon and Szegö, 1978) test problem in 2-dimensional space has 3 global minimum points which are located in a shallow flat valley. The derived curves of $CDFR_N^{[A]}(y)$ are presented in Fig. 8. Since the function has no local non-global min-

ima, MS always descends to the global minimum independently of the starting point and the budget of function evaluations $N = 200, 500$, or 1000. Therefore, the probability to find the global minimum using MS is one, and the curve of $CDFR_N^{[\mathrm{MS}]}(y)$ coincides with the vertical axis. One can say that for such instances, the benchmark of MS dominates the population algorithms from at least the budget N_{LS} of performing one local search. Another interesting property of the problem instance is that the least effective algorithm when using $N = 200$ appears to be GA, whereas CRS is the most effective of all population-based algorithms. The performance of GA is very similar to the performance of PSO when $N = 500$ is used. However, both of them are worse than CRS and MS. It should be noted that if the aim was to find all global minimum points another criterion should be used.

4. Conclusions

Heuristics for the box-constrained global optimization problem are often tested on a set of test instances. We showed that the Cumulative Distribution Function of the obtained function value provides the answer (in terms of domination) to the question of which algorithms behave well on which type of instances. We argue that a good minimum performance benchmark is due to pure random search; i.e. algorithms should do better. The concepts have been illustrated for several well-known population algorithms. It shows how domination is determined by the characteristics of an individual test problem. In the selection of an algorithm, one better first studies the underlying characteristics of the black-box problem to be solved.

Acknowledgements. This research was funded by a grant (No. MIP-51/2014) from the Research Council of Lithuania. This research is supported by grants from the Spanish Ministry (TIN2012-37483-C03-01) and Junta de Andalucía (P11-TIC-7176), in part financed by the European Regional Development Fund (ERDF).

References

Ackley, D.H. (1987). *A Connectionist Machine for Genetic Hillclimbing*. Kluwer, Boston.

Ali, M.M., Khompatraporn, C., Zabinsky, Z.B. (2005). A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *Journal of Global Optimization*, 31, 635–672.

Baritompa, W.P., Hendrix, E.M.T. (2005). On the investigation of stochastic global optimization algorithms. *Journal of Global Optimization*, 31, 567–578 .

İlker Birbil, Ş., Fang, S.C. (2003). An electromagnetism-like mechanism for global optimization. *Journal of global optimization*, 25(3), 263–282.

Chelouah, R., Siarry, P. (2000). A continuous genetic algorithm designed for the global optimization of multimodal functions. *Journal of Heuristics*, 6(2), 191–213.

Davis, L. (1991). *Handbook of Genetic Algorithms*. Van Nostrand–Reinhold, New York.

Dixon, L.C.W., Szegö, (1975). *Towards Global Optimization*. North-Holland, Amsterdam.

Dixon, L.C.W., Szegö, G.P. (1978). *Towards Global Optimization 2*. North-Holland, Amsterdam.

Dolan, E.D., Moré, J.J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91, 201–2013.

Easom, E.E. (1990). *A survey of global optimization techniques*. University of Louisville, Louisville.

Hendrix, E.M.T., Roosma, J. (1996). Global optimization with a limited solution time. *Journal of Global Optimization*, 8, 413–427.
Hendrix, E.M.T., Tóth, B.G. (2010). *Introduction to Nonlinear and Global Optimization*. Springer, New York.
Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.
Jelasity, M., Ortigosa, P.M., García, I. (2001). UEGO, an abstract clustering technique for multimodal global optimization. *Journal of Heuristics*, 7(3), 215–233.
Kennedy, J., Eberhart, R.C. (1995). Particle swarm optimization. In: *Proceedings of IEEE International Conference on Neural Networks*, Piscataway, NJ, pp. 1942–1948.
Misener, R., Floudas, C.A. (2014). ANTIGONE: algorithms for continuous/integer global optimization of nonlinear equations. *Journal of Global Optimization*, 60, 123–144.
Müller, J., Schoemaker, C.A. (2014). Influence of ensemble surrogate models and sampling strategy on the solution quality of algorithms for computationally expensive black-box global optimization problems. *Journal of Global Optimization*, 59, 503–526.
Price, W.L. (1979). A controlled random search procedure for global optimization. *The Computer Journal*, 20, 367–370.
Recchioni, M.C., Scoccia, A. (2000). A stochastic algorithm for constrained global optimization. *Journal of Global Optimization*, 16(3), 257–270.
Redondo, J.R., Ortigosa, P.M., Žilinskas, J. (2012). Multimodal evolutionary algorithm for multidimensional scaling with city-block distances. *Informatica*, 23(4), 601–620.
Rios, L.M., Sahinidis, N.V. (2013). Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56, 1247–1293.
Törn, A., Žilinskas, A. (1989). *Global Optimization*. *Lecture Notes in Computer Science*, Vol. 350. Springer, Berlin.
Vaz, A.I.F., Vicente, L.N. (2007). A particle swarm pattern search method for bound constrained global optimization. *Journal of Global Optimization*, 39(2), 197–219.
Zhigljavsky, A., Žilinskas, A. (2008). *Stochastic Global Optimization*. Springer, New York.
Zielinski, R. (1981). A statistical estimate of the structure of multi-extremal problems. *Mathematical Programming*, 21, 348–356.

E.M.T. Hendrix is a European researcher in the field of optimization algorithms. He is affiliated to the universities of Wageningen and Málaga, but also teaches at other universities. His research interests are Global and Dynamic optimization and computational impacts.

A. Lančinskas received the doctoral degree in informatics from Institute of Mathematics and Informatics of Vilnius University in 2013. Currently he is a researcher at the same institute and lecturer at the department of Informatics of the Lithuanian University of Educational Sciences. His research interest is focused on development and investigation of global and multi-objective optimization algorithms and their parallelization.

Apie stochastinių globaliojo optimizavimo algoritmų lyginamąją analizę

Eligius M.T. HENDRIX, Algirdas LANČINSKAS

Literatūroje sutinkama daugybė įvairių euristinių optimizavimo algoritmų, kurių sustojimo kriterijus yra paremtas tikslo funkcijų perskaičiavimų skaičiumi, o jų našumas vertinamas remiantis vidutine tikslo funkcijos reikšme. Šiame straipsnyje nagrinėjamas bendresnis euristinių algoritmų našumo eksperimentinio vertinimo metodas, skirtas įvertinti, kurie algoritmai yra labiau tinkami konkrečios klasės optimizavimo uždaviniams spręsti. Straipsnyje siūlomas metodas yra grįstas tikslo funkcijos reikšmių pasiskirstymo funkcijos vertinimu, atsižvelgiant į paprastosios atsitiktinės paieškos (prasčiausio atvejo) našumą sprendžiant konkretų uždavinį. Siūlomo metodo efektyvumas iliustruojamas vertinant gerai žinomų euristinių algoritmų našumą įvairiems testo uždaviniams spręsti.