On the program implementation of a Markov homogeneous random search algorithm of an extremum with normal distributions

IOP Conf. Series: Journal of Physics: Conf. Series **1352** (2019) 012052 doi:10.1088/1742-6596/1352/1/012052

On the program implementation of a Markov homogeneous random search algorithm of an extremum with normal distributions

A S Tikhomirov

Yaroslav-the-Wise Novgorod State University, ul. B. St. Petersburgskaya, 41
173003 Veliky Novgorod, Russia

E-mail: Alexey.Tikhomirov@novsu.ru

Abstract. A program that implements a Markov homogeneous monotonous random search algorithm of an extremum with normal distributions is presented. This program allows to solve a fairly wide class of problems of finding the global extremum of an objective function with a high accuracy.

1. Introduction

Let the *objective function* $f\colon \mathbb{R}^d \mapsto \mathbb{R}$ take its minimum value at a single point x_*. Let us consider the problem of search for the point of global minimum x_* with the given accuracy $\varepsilon > 0$. A way to solve the problem is to apply algorithms of random search for function extremum (q.v. [1–18]). Such methods have been long and successfully used at solving complex problems of optimization. Theoretical investigations of convergence rate of some algorithms of Markov search have been presented in the works [3, 11–17]. The given work is continuation of the works [11–14, 18] and is devoted to a computer program implementing an algorithm of homogeneous Markov monotonous search for extremum which uses the normal distribution of probabilities. The given computer program complements the program [19], in which the algorithm of homogeneous random search using different (from normal) distribution of probabilities has been implemented as well.

2. Problem Statement

The optimization space is defined as the optimization set X equipped with a metric ρ. In this paper, we consider the case $X = \mathbb{R}^d$ and the Euclidean metric

$$\rho(x, y) = \left(\sum_{n=1}^{d} (x_n - y_n)^2 \right)^{1/2},$$

where $x = (x_1, \dots, x_d)$ and $y = (y_1, \dots, y_d)$. The closed ball of radius r centered at the point x is denoted by $B_r(x) = \{y \in \mathbb{R}^d \colon \rho(x, y) \le r\}$. Let $\mathcal{F}$ be σ-algebra of Borel sets in $\mathbb{R}^d$. We'll denote Lebesgue measure on Borel subsets $\mathbb{R}^d$ with μ.

To find the point of minimum we'll use *homogeneous Markov monotonous* random search (q.v. [3, 11-14, 18]), which will be further described with the help of algorithm of modeling. Denotation "$\eta \leftarrow P(\cdot)$" is read as follows: "to obtain implementation of random vector η with distribution P". For

numbers and points in the optimization space, operations of kind $k \leftarrow 1$ and $\xi \leftarrow x$ denote the usual assignment operations.

Algorithm 1

Step 1. $\xi_0 \leftarrow x$, $k \leftarrow 1$.

Step 2. $\eta_k \leftarrow P(\xi_{k-1}, \cdot)$.

Step 3. If $f(\eta_k) \le f(\xi_{k-1})$, then $\xi_k \leftarrow \eta_k$, otherwise $\xi_k \leftarrow \xi_{k-1}$.

Step 4. If $k = N$, complete the algorithm work.

Step 5. $k \leftarrow k + 1$ and pass to step 2.

Here x — the initial search point, N — the number of search steps, $P(x, \cdot)$ — Markov transition function (q.v. [3, 11–14]).

In the first step of algorithm 1, initialization of random search is performed. An initial search point will be the point x (operator $\xi_0 \leftarrow x$), and the number of the following search step k becomes equal to one (operator $k \leftarrow 1$).

In the second step of algorithm 1, we obtain a new "trial" point η_k in the optimization space. We select a new "trial" point randomly, using distribution $P(\xi_{k-1}, \cdot)$. The distribution $P(\xi_{k-1}, \cdot)$ depends on the position of the "old" search point ξ_{k-1}. Such dependence allows to increase the effectiveness of random search. The transition function $P(x, \cdot)$ will be called a *trial transition function*. We will consider the search which has the trial transition function that does not depend explicitly on the number of step k. Such search is called *homogeneous*. The search which has trial transition functions that depend explicitly on the number of step k is called *inhomogeneous*. Certainly, search effectiveness can be improved due to inhomogeneity. But such dependence complicates the selection of search parameters, and the "correct" selection of search parameters may become a very complex problem (q.v., e.g., [7]). That is why, to make the search simpler, we limit ourselves to the consideration of homogeneous search.

In the third step of algorithm 1, we compare the new trial point η_k with the old search point ξ_{k-1}. If the new trial point η_k is not worse than the old search point ξ_{k-1} (i.e., if inequality $f(\eta_k) \le f(\xi_{k-1})$ is performed), then the search moves to a new point η_k (the operator $\xi_k \leftarrow \eta_k$ is performed), otherwise the search remains in the old point (the operator $\xi_k \leftarrow \xi_{k-1}$ is performed).

In the fourth step of algorithm 1, we verify the condition of stopping the search. In the given case, a very simple criterion of stopping search has been selected. Simply, the search makes previously the given number of steps N, and then stops.

Note that the second, third, fourth and fifth step of algorithm 1 are repeated cyclically N times. The first step of algorithm 1 is performed only once.

Note also that the introduced random search is *monotonous*, i.e., inequalities $f(\xi_k) \le f(\xi_{k-1})$ are performed at all $k \ge 1$.

3. Selection of Transition Functions of Random Search

The key question of selecting the examined search type is to select the kind of the trial transition function $P(x, \cdot)$. At selecting transition functions, two criteria are usually used. Firstly, the search should be sufficiently effective (i.e., to require not too many steps to solve the given problem). Besides, the modeling of the distribution $P(x, \cdot)$ should be sufficiently simple.

Very often, normal distribution of probabilities is used as trial transition functions (q.v., e.g., [5, 7]). In the given work we will use the mixture of normal distributions with standard deviations belonging to the interval $[\nu, \Gamma]$, where numbers ν and Γ are the method parameters. The given examples of applying the written computer program demonstrate that the use of distribution mixture allows to increase substantially the search effectiveness.

To select a trial transition function, it is left to define the rule of making the mixture of normal distributions. We'll apply the results stated in [3, 12–14]. In [12, 14], the theoretical evaluations of complexity of homogeneous Markov random search have been obtained. Besides, the authors have found the trial transition function $P(x, \cdot)$, which minimizes the evaluation of complexity of random search at optimizing the elementary objective function. Such trial transition function is the mixture of uniform distributions in d-dimensional cubes with centers in the current search point. Here we will use

exactly the same distribution mixture that was obtained in the work [12], only we will not mix uniform distributions in d-dimensional cubes, but we will mix normal probability distributions instead.

4. Random Search Modeling

In this section we present the modeling algorithm selected to implement homogeneous Markov monotonous random search. The presented search has just three parameters. Parameters giving the range of alterations of standard deviations of normal distributions are positive numbers ν and Γ, for which inequalities $0 < \nu \leq \Gamma$ must be performed. The third parameter is N, the number of search steps.

To give the modeling algorithm for distribution mixture, we will use the results [12]. The modeling of a new trial search point takes two stages. At the first stage, the standard deviation σ is being modeled by a special way. At the second stage, d of independent normal random values with zero mean and obtained standard deviation σ is being modeled.

These random values will be added to the current search point ξ_{k-1}, to obtain a new "trial" point η_k in the second step of algorithm 1.

Let $\gamma = \Gamma/2^{1/d}$ and $0 < \nu < \gamma \leq \Gamma$. Suppose $\lambda = d\ln(\gamma/\nu) + 2, p = d\ln(\gamma/\nu)/\lambda$ and $q = \lambda/d$. Let α be a random variable uniformly distributed on the interval $[0, 1]$.

Algorithm of modeling standard deviation

Step 1. Get α.

Step 2. If $\alpha \geq p$, then $\sigma \leftarrow \Gamma$, otherwis $\sigma \leftarrow \nu\exp(\alpha q)$.

In the case when $\gamma \leq \nu$, suppose $\sigma = \Gamma$.

At the second stage, we will use a modified polar method for normal random values to be modeled. The modified polar method is intended to model two independent standard normal random variables ζ_1 and ζ_2. Let α_1 and α_2 be independent random variables uniformly distributed on the interval $[0, 1]$.

Algorithm of modeling standard normal distribution

Step 1. Get α_1 and α_2. $\beta_1 \leftarrow 2\alpha_1 - 1$, $\beta_2 \leftarrow 2\alpha_2 - 1$, $\delta \leftarrow \beta_1^2 + \beta_2^2$.

Step 2. If $\delta > 1$, then pass to step 1.

Step 3. $t \leftarrow \sqrt{-2\ln(\delta)\ /\ \delta}$, $\zeta_1 \leftarrow \beta_1 t$, $\zeta_2 \leftarrow \beta_2 t$.

Since the given algorithms of modeling are sufficiently simple, then, on the whole, algorithms of modeling random search are quite easy to program. The given algorithm of modeling is slightly more complex than the algorithm of modeling random search in the work [18] and more complex than the algorithm of modeling the simplest random search (the so-called "blind search" [3, 5]) which uses the uniform distribution in previously fixed region of the optimization space.

5. Program Description

The program is written in the language C# in the Microsoft Visual Studio Professional 2010 integrated development environment. The program has the graphical user interface written with the help of Windows Forms. You can download the program at the address www.novsu.ru/doc/study/tas1 from the folder "Random_search". The program is available both as the executable file and as the project which contains the program source code and allows for users to edit the program on their own.

To run the executable program file, Microsoft .NET Framework 4 is required. As a rule, it is installed on the computer, but, if needed, you can download it from Microsoft. To edit the project, it is necessary to install the development environment Microsoft Visual Studio. The given development environment is free of charge, so it may serve as a convenient tool for scientific computations.

For the program computations, the numeric type double is used which can provide 15–16 – digit precision. Note that this numeric format limits possible precision of solving a problem. Speculating somewhat primitively, we'll conclude the following. If the objective function behaves approximately as a quadratic function in the neighborhood of global minimum, then at argument approximation accuracy in the order of 10^{-8} we obtain the approximation accuracy by the function value in the order of 10^{-16}. If minimal value of the objective function belongs to the interval $(1, 10)$, then the numeric type double

giving 15–16 – digit precision does not allow to calculate the function value with more than 10^{-16}. Thus, typical accuracy of solving a problem will be in the order of 10^{-7} at argument approximation, and in the order of 10^{-14} at function value approximation. As a rule, this accuracy will be enough from a practical point of view. Such accuracy of solving the problem can be obtained at using the program of random search under consideration, while solving not too complex optimization tasks. Certainly, if minimal value of the objective function equals zero, and the minimum point is also at zero, then the problem can be solved much more precisely.

To apply the search, it is necessary to give an objective function, search parameters and an initial search point. The search results will be a search end point (approximating the point of global minimum) and the objective function value at the search end point.

Search parameters and an initial search point are easy to give in the main program window.

It is more difficult to give an objective function. An objective function can be given by two ways. Firstly, you can write the function code directly in the program code (in the language C#). The given way has been described in more detail in [18]. While writing the code calculating the objective function value, you need usually minimal information about any programming language like C, C++, C#, Java.

Secondly, the objective function can be given in the search program itself (without using Microsoft Visual Studio). To do this, you need to press the button "Set formula", select the box "Use formula" in the dialogue that opens, indicate dimensionality of the optimization space, and write the formula defining the objective function. Rules of writing formulas are typical for mathematical programs and have been given in [18]. The way of setting an objective function is shown in the text field with inscriptions "Use function code" or "Use formula".

Optimization space dimensionality is indicated at setting the function.

In the program you can set the formats of outputting the value of the objective function and point coordinates (q.v. [18]).

You can write comments to the problem being solved. Comments can be written in text format and saved in a file together with search parameters and results.

To perform the search, the program uses the pseudo random number generator. It can be initialized either with the value depending on computer system time, or the value set.

The program can save data in the format XML, and export key search characteristics in text format.

If the value of the parameter "Number of search steps N" is set equal to zero, the program will calculate the value of the function in the initial search point. This can be used to calculate the value of the objective function in the given point.

Note that, while writing the program, users of the computer operating system Windows XP can work with the project even if they use the old development environment Microsoft Visual Studio 2010.

6. Search Parameters Selection

It is important to note that selection of search parameters may influence greatly the efficacy of a random search method [3, 5, 7]. Withal, many search algorithms include a variety of heuristic parameters, and the user of such algorithms can hardly find "good" values of parameters matching the function optimized. Let us quote [7] about the method of very fast annealing supposed by L. Ingber: "Among the disadvantages of this method we can name the fact that, due to a great variety of parameters, sometimes several months are required to tune it well for solving an individual problem". Withal, if parameters have been selected correctly, the very fast annealing method can demonstrate good results [6, 7].

The given search has only three parameters. The parameters setting the range of altering standard deviations of normal distributions are positive numbers ν and Γ for which inequalities $0 < \nu \leq \Gamma$ must be performed. The third parameter is N, the number of search steps.

The value ν may be selected as close to the required accuracy of solving the problem at argument approximation. So, the value Γ may be selected as close to either the supposed accuracy of initial approximation (the distance between the initial search point and the minimum point), or the diameter of the region examined in the optimization space. While selecting Γ, you can use the upper estimate.

The number of search steps N is recommended to be taken sufficiently large. Solving a singular problem, it is possible, for example, to perform billion of search steps, even if the problem is simple enough and can be solved much faster. Modern personal computers can perform similar amounts of computations quite easily, at least if objective functions are not very complex. Indeed, for similar amounts of computations the code of objective function is required to be set programmatically in the programming language of C#.

Besides three search parameters, it is necessary to select an initial search point. It is clear that an initial point is better to be placed nearer the point of global extremum.

To a large extent, the supposed search algorithm is free from irresistible difficulties of selecting parameters. Particularly, in the numerical examples of the following section the minimal selection of parameters consisting literally of some attempts to run the program with different values of parameters has been performed.

7. Examples of Using Program

We will give some examples of using the presented program for solving optimization tasks. Personal computer with processor Intel Core i5-4460S has been used for computations.

7.1. Example 1

Let us take the example from [5]. Here the optimization space $X = \mathbb{R}^2$, $x = (x_1, x_2)$,

$$f(x) = f(x_1, x_2) = x_1^4 + x_1^2 + x_1 x_2 + x_2^2.$$

Function f takes the minimal value in the unit point $x_* = (0, 0)$ and $f(x_*) = 0$. $x = (1, 1)$ is selected as the initial search point, and $f(x) = 4$. Here, the number of search steps N takes the value 10^4.

Algorithm B in the book [5] takes the minimal value of the objective function 2.7×10^{-6}. Algorithm B matches the search of algorithm 1 at using normal distribution of probabilities as the trial transition function.

Algorithm C in the book [5] takes the minimal value of the objective function 2.5×10^{-7}. Algorithm C also uses the normal distribution of probabilities as the trial transition function, but it presents more complex search variant, in which, at constructing a new search point, the movement performed in the previous algorithm step is being considered.

Algorithm of homogeneous search in the article [18] takes the minimal value of the objective function 9.9×10^{-49}.

Algorithm 1 of the given work with parameters $\nu = 10^{-24}$ and $\Gamma = 1$ gets the minimal value of the objective function 7.8×10^{-50}.

In this example, the search of algorithm 1 has proved to be much more accurate than algorithms B and C in the book [5] which use normal distribution of probabilities. The results of algorithm 1 are similar to those in the article [18].

Algorithm 1 of the given work with parameters $\nu = 10^{-163}$,$\Gamma = 1$and $N = 10^6$ gets the minimal value of the objective function equal to zero (i.e., less than the value 5×10^{-324}, which defines a range of values of the double type of the C# programming language) and the point of minimum $(-2.7 \times 10^{-163}, 4.0 \times 10^{-163})$. Note that in this case the extreme accuracy has been achieved, which allows to make computations in the C# at using the numerical format double (because it is impossible to calculate the value of the objective function more accurately).

Search algorithm in the article [18] has also required 10^6 steps to obtain the value of the objective function equal to zero. Here, the results of algorithm 1 are like those in the article [18].

7.2. Example 2

Here, the optimization space $X = [-8, 8]^2$, $x = (x_1, x_2)$,

$$f(x) = f(x_1, x_2) = \frac{1}{2}\Big((x_1^4 - 16x_1^2 + 5x_1) + (x_2^4 - 16x_2^2 + 5x_2)\Big).$$

Function f has four local minimums, one of which is global. $x = (4.0, 6.4)$ has been selected as an initial search point, and $f(x) = 537.18$. The search of algorithm 1 with parameters $\nu = 10^{-7}$, $\Gamma = 10$ and $N = 20000$ finds the minimal value of the objective function -78.3323314075428 and the point of minimum $(-2.903534, -2.903534)$. Let us note that here the extreme accuracy has been achieved that allows to compute in the C# at using the number format double (because you cannot calculate the value of objective function more accurately).

The results obtained are close to the results of homogeneous search in the article [18].

7.3. Example 3

Here, the space $X = [-4, 4]^{10}$, $x = (x_1, x_2, \dots, x_{10})$,

$$f(x) = f(x_1, x_2, \dots, x_{10}) = \sum_{n=1}^{5} (100(x_{2n} - x_{2n-1}^2)^2 + (1 - x_{2n-1})^2).$$

Function *f* is a known test function of Rosenbrock used for local optimization methods. The function f gets the minimal value $f(x_*) = 0$ in the point $x_* = (1, 1, \dots, 1)$. $x = (-1.2, 1, -1.2, 1, \dots, 1)$ has been selected as an initial search point and $f(x) = 121$. The search of algorithm 1 with parameters $\nu = 10^{-17}$, $\Gamma = 4$ and $N = 10^7$ finds the minimal value of the objective function 2.8×10^{-28}. The search execution time was 3.5 seconds.

The results obtained are close to the results of homogeneous search in the article [18]. But there the search execution time was 1.8 seconds. Normal distribution of probabilities used in the examined search is somewhat more difficult to model than uniform distributions in d-dimensional cubes.

7.4. Example 4

Let us consider the example with a very simple objective function, but in the optimization space of a very large dimensionality for random search methods. Here, the space $X = \mathbb{R}^{1000}$, $x = (x_1, x_2, \dots, x_{1000})$, $f(x) = \sum_{n=1}^{1000} x_n^2$. Function f takes the minimal value $f(x_*) = 0$ in unit point. $x = (1, 1, \dots, 1)$ has been selected as an initial search point. The search of algorithm 1 with parameters $\nu = 10^{-10}$, $\Gamma = 10$ and $N = 10^6$ finds the minimal value of the objective function 2.3×10^{-14}. The search execution time was 30 seconds.

Homogeneous search in the article [18] at $N = 10^6$ obtained the minimal value of the objective function 1.6×10^{-14}. The search execution time was 13 seconds. The obtained results are close to the results of homogeneous search in the article [18], but the search of algorithm 1 is somewhat slower due to the complexity of modeling normal distribution.

8. Conclusion

The obtained results demonstrate that the presented algorithm of homogeneous search based on the use of normal distribution of probabilities is sufficiently effective. The presented random search program can be successfully used to solve optimization tasks. The program itself is easy to apply, and search parameters selection is not a complex task. Herewith, the program allows solving problems with extreme accuracy, which can be obtained by using the numerical format double of the C# programming language.

References

[1] Ermakov S M and Zhiglyavsky A A 1983 On random search of global extremum *Probability theory and its applications* **1** 129–136
[2] Ermakov S M, Zhiglyavsky A A and Kondratovich M V 1989 On comparison of some procedures of random search of global extremum *Journal of Computational Mathematics and Mathematical Physics*. Vol **29** 2 163–170
[3] Zhigljavsky A and Zilinskas A 2008 *Stochastic global optimization* Springer-Verlag Berlin
[4] Zhigljavsky A and Zilinskas A 2016 Stochastic global optimization: a review on the occasion of 25 years of Informatica *Informatica* Vol **27 2** 229–256
[5] Spall J C 2003 *Introduction to stochastic search and optimization: estimation, simulation, and*

control Wiley New Jersey
[6] Ingber L 1989 Very fast simulated re-annealing *Mathl. Comput. Modelling* Vol **12** 967–973
[7] Lopatin A S 2005 Method of annealing *Stochastic optimization in informatics* **1** 133–149
[8] Granichin O N and Polyak B T 2003 *Randomized estimation and optimization algorithms under almost arbitrary noise* Nauka Moscow
[9] Sushkov Yu A 1972 On one way of organizing random search *Operations research and statistical modeling* Publishing House LSU Leningrad **1** 180–186
[10] Tarlowski D 2017 On the convergence rate issues of general Markov search for global minimum *Journal of Global Optimization* Vol **69 4** 869–888
[11] Nekrutkin V V and Tikhomirov A S 1993 Speed of convergence as a function of given accuracy for random search methods *Acta Applicandae Mathematicae* **33** 89–108
[12] Tikhomirov A S 2006 On the Markov homogeneous optimization method *Computational Mathematics and Mathematical Physics* Vol **46 3** 361–375
[13] Tikhomirov A S 2007 On the convergence rate of the Markov homogeneous monotone optimization method *Computational Mathematics and Mathematical Physics* Vol **47 5** 780–790
[14] Tikhomirov A, Stojunina T and Nekrutkin V 2007 Monotonous random search on a torus: integral upper bounds for the complexity *Journal of Statistical Planning and Inference* Vol **137 12** 4031–4047
[15] Tikhomirov A S 2010 On the convergence rate of the simulated annealing algorithm *Computational Mathematics and Mathematical Physics* Vol **50 1** 19–31
[16] Tikhomirov A S 2011 Lower bounds on the convergence rate of the Markov symmetric random search *Computational Mathematics and Mathematical Physics*. Vol **51 9** 1524–1538
[17] Tikhomirov A S 2011 On the rate of convergence of one inhomogeneous Markov algorithm of search for extremum *Vestnik St. Petersburg University. Mathematics* Vol **44 4** 309–316
[18] Tikhomirov A S 2018 On the program implementation of a Markov homogeneous monotonous random search algorithm of an extremum *IOP Conference Series: Materials Science and Engineering* Vol **441 012055** 1–8
[19] Tikhomirov A S 2016 Computer program "MarkovMonotonousSearch" *Certificate of state registration of computer programs* № 2016663645 (2016)