On the program implementation of a Markov homogeneous random search algorithm of an extremum with Ingber’s distribution

A S Tikhomirov¹ and N Yu Kropacheva²

¹Yaroslav-the-Wise Novgorod State University, ul. B. St. Petersburgskaya, 41 173003 Veliky Novgorod, Russia
²Saint Petersburg State University, Universitetskaya Naberezhnaya, 7-9, 199034 Saint Petersburg, Russia

E-mail: Alexey.Tikhomirov@novsu.ru

Abstract. A program that implements a Markov homogeneous monotonous random search algorithm of an extremum with Ingber’s distribution 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 \( f : \mathbb{R}^d \mapsto \mathbb{R} \) take the minimum value at a single point \( x^* \). Consider the problem of finding the global minimum point \( x^* \) with a given accuracy \( \varepsilon > 0 \). One way of solving this problem is to use random search algorithms (see [1–19]). Such algorithms have long been used for solving difficult optimization problems. Theoretical studies of the convergence rate of some Markov search algorithms are given in [3, 11–18]. This work is a continuation of [11–14, 19] and is devoted to a computer program that implements one of the algorithms of homogeneous Markov monotonic extremum search using the Ingber probability distribution. The presented computer program complements the program [20], in which the algorithm of homogeneous random search was also implemented but another probability distribution was used.

2. Problem Statement

As the optimization space we will consider the space \( X = \mathbb{R}^d \) with the Euclidean metric

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

where \( x = (x_1, ..., x_d) \) and \( y = (y_1, ..., y_d) \). A closed ball of radius \( r \) with the center at point \( x \) will be denoted as \( B_r(x) = \{ y \in \mathbb{R}^d : \rho(x, y) \leq r \} \). Let \( \mathcal{F} \) be \( \sigma \)-algebra of borel sets in \( \mathbb{R}^d \). We will denote a Lebesgue \( d \)-dimensional measure on Borel subsets \( \mathbb{R}^d \) through \( \mu \).

We will use homogeneous Markov monotone random search (see [3, 11–14, 19]) to find the minimum point further described with the help of the simulation algorithm. The notation “\( \eta \leftarrow P(\cdot) \)” is read as: “to
get the realization of a random vector $\eta$ with distribution $P''$. For numbers and points in the optimization space of the operations of the form $k \leftarrow 1$ and $\xi \leftarrow x$ denote simple assignment operators.

**Algorithm 1**

1. $\xi_0 \leftarrow x, k \leftarrow 1$.
2. $\eta_k \leftarrow P(\xi_{k-1}, \cdot)$.
3. If $f(\eta_k) \leq f(\xi_{k-1})$, then $\xi_k \leftarrow \eta_k$, otherwise $\xi_k \leftarrow \xi_{k-1}$.
4. If $k = N$, then finish the algorithm.
5. $k \leftarrow k + 1$ and go to Step 2.

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

The first step of algorithm 1 initializes a random search. Point $x$ becomes the starting point of the search (operator $\xi_0 \leftarrow x$) and the number of the next step of the search is equal to one (operator $k \leftarrow 1$).

At the second step of algorithm 1 we get a new “trial” point $\eta_k$ in the optimization space. We choose the new “trial” point randomly using the distribution $P(\xi_{k-1}, \cdot)$. The distribution $P(\xi_{k-1}, \cdot)$ depends on the location of the “old” search point $\xi_{k-1}$. This dependency improves the efficiency of random search. The transition function $P(x, \cdot)$ will be called trial transition function. We will consider search whose trial transition function does not depend explicitly on the step number $k$. Such search is called homogeneous. A search whose trial transition functions explicitly depend on the step number $k$ is called inhomogeneous. Of course, due to the inhomogeneity the efficiency of the search can be improved. But this dependence complicates the choice of search parameters and the “correct” choice of search parameters can be a very difficult task (see, for example, [7]). Therefore, in order to ease the use of search we will limit ourselves by the consideration of homogeneous search.

At the third step of algorithm 1, we compare the new trial point $\eta_k$ with the old search point $\xi_{k-1}$. If the new trial point $\eta_k$ is not worse than the old search point $\xi_{k-1}$ (i.e. if the inequality $f(\eta_k) \leq f(\xi_{k-1})$ is satisfied), the search moves to the new point $\eta_k$ (the operator $\xi_k \leftarrow \eta_k$ is executed), otherwise the search remains at the old point (the operator $\xi_k \leftarrow \xi_{k-1}$ is executed).

At the fourth step of algorithm 1, we check the condition of stopping the search. In this case, a very simple criterion for stopping the search is selected. The search simply takes a predetermined number of steps $N$ and stops after that.

Let us note, that the second, the third, the fourth and the fifth steps of the algorithm 1 repeats cyclically for $N$ times. The first step of algorithm 1 is performed only once.

Let us also note that the introduced random search is monotonous, it means that the inequality $f(\xi_k) \leq f(\xi_{k-1})$ is held for all $k \geq 1$

### 3. The choice of transition functions of the random search

The key question of choice for the type of search under study is the choice of the type of the trial transition functions. When choosing transition functions two criteria are typically used. First, the search should be quite effective (it should not require too many steps to solve the problem). Moreover, the simulation of the distribution $P(x, \cdot)$ should be simple enough.

To construct the distribution $P(x, \cdot)$, we use the distribution used by L. Ingber in the method of very fast annealing, see [6, 7]. The Ingber search algorithm itself is arranged quite difficult. The fact is that L. Ingber used an annealing algorithm and not a simple monotonous search. Let us quote from [7], referring to the Ingber quick method of very fast annealing: “Among the drawbacks of this method we can mention for the large number of parameters it sometimes takes several months to properly configure it to solve a specific problem”. Moreover, with proper selection of parameters, the method of very fast annealing can show good results [6, 7]. It is clear that such great difficulties in the choice of search parameters are a serious obstacle to the application of the method. Therefore, we simplify the method of Ingber by making it more convenient to use.

In this work we will use the distribution used by L. Ingber but we will substantially simplify the search construction rules. First, instead of the annealing algorithm, we will use a monotonic search with a simple transition rule to a new search point in the third step of algorithm 1. Secondly, our search will be
homogeneous, that means we will not need to change the trial transition function during the search run (in the Ingber method the transition function changes at each step of the search). The considered examples of application of the written computer program show that the obtained simple search algorithm is quite effective. We note that a theoretical study of such a search was performed in [15].

4. Random search simulation
In this section we present a simulation algorithm chosen to implement a homogeneous Markov monotone random search. The presented search has only three parameters. Positive numbers \(\nu\) and \(\Gamma\) are the parameters that define the trial transition function for which the inequalities \(0 < \nu \leq \Gamma\) must be performed. \(N\) is the third parameter – number of search steps.

To describe the simulation algorithm for a trial transition function, we use the results of [6, 7, 15]. Let a random vector \(\Psi^* = (\psi^*_1, \ldots, \psi^*_d)\) with values in \([-1, 1]^d\) have a density \(h(x)\) given by the following formula

\[
h(x) = \prod_{n=1}^{d} \frac{1}{2(|x_n| + \nu) \ln(1 + 1/\nu)},
\]

where \(x = (x_1, \ldots, x_d), x_n \in [-1, 1]\).

Using the method of inverse functions it is not difficult to obtain a modeling formula for \(\psi^*_n\):

\[
\psi^*_n = \text{sgn}(\alpha_n - 1/2)\nu((1 + 1/\nu)^{|2\alpha_n-1|} - 1),
\]

where \(\alpha_n\) is a random variable uniformly distributed on the interval \([0, 1]\) and \(\text{sgn}\) — is the signum function.

To construct the studied search, we use a random vector \(\Psi = (\psi_1, \ldots, \psi_d)\) with a distribution similar to the vector \(\Psi^*\) and with values in \([-\Gamma, \Gamma]^d\). To obtain the vector \(\Psi\) the components of the vector \(\Psi^*\) are to be multiplied by \(\Gamma\). Thus, we come to the following modeling formula for \(\psi_n\):

\[
\psi_n = \text{sgn}(\alpha_n - 1/2)\nu\Gamma((1 + 1/\nu)^{|2\alpha_n-1|} - 1).
\]

In order to get a new “trial” point at the second step of the algorithm 1 this random vector \(\Psi = (\psi_1, \ldots, \psi_d)\) is added to the “old” search point. Thus, we believe that \(\eta_k = \xi_{k-1} + \Psi_k\).

Since the above modeling formula is quite simple, the algorithm of random search modeling is quite easy to program. The presented algorithm is a bit more complicated than the algorithm for modeling random search [19] and more complex than the algorithm for modeling the simplest random search (the so-called “blind search” [3, 5]), where a uniform distribution in a pre-fixed area of the optimization space is used.

5. Program description
The program is written in language C# in the integrated development environment Microsoft Visual Studio Professional 2010. The program has a graphical user interface written with Windows Forms. You can download the program at www.novsu.ru/doc/study/ from the “Random search” folder. The program is also available as an executable file and in the form of a project that contains the source code of the program that allows the user to edit the program at own discretion.

Microsoft. NET Framework 4 is required to run the executable program file. Usually it is already installed on the computer, but if necessary it can be downloaded from the Microsoft website. To edit the project, you need to install the Microsoft Visual Studio development environment. This development environment can be used free of charge, and therefore this development environment can serve as a convenient tool for scientific calculations.

For calculations the program uses a numeric type double providing an accuracy of 15–16 signs. Note that this numerical format limits the possible accuracy of the solution of the problem. Reasoning a bit simplistically, we will get the following conclusions. If the objective function behaves approximately as
a quadratic function in the neighborhood of the global minimum then with the accuracy of the approximating by the order argument $10^{-8}$, we obtain the accuracy of the approximation by the order the value of function $10^{-16}$. If the minimum value of the objective function belongs to the interval $(1, 10)$, the numeric type double, providing an accuracy of $15-16$ signs, will not allow to calculate the value of the function with an accuracy higher $10^{-16}$. Thus, the typical accuracy of the solution of the problem will be of the order $10^{-7}$ when approximating by the argument, and of the order $10^{-14}$ when approximating by the value of the function. This accuracy is usually enough from the practical point of view. And such accuracy of the solution of the problem can be obtained by using the considered random search program in the solution of not too complex optimization problems. Of course, if the minimum value of the objective function is zero and the minimum point is also zero then the problem can be solved with much higher accuracy.

To apply a search, you must specify a target function, search parameters and a starting search point. The search endpoint (which approximates the global minimum) and the value of the objective function at the search endpoint will be the results of the search.

The search parameters and the starting search point can be easily set in the main window of the program.

It is more difficult to set the objective function. The objective function can be set in two ways. First, you can write the function code directly in the program code (in C#). This method is described in more detail in [19]. When writing code that calculates the value of the objective function, as a rule, minimal information about some programming language such as C, C++, C#, Java is sufficient.

Secondly, the objective function can be set in the search program itself (without using Microsoft Visual Studio). To do this, click “Set formula” in the opened program dialog box, select “Use formula”, specify the dimension of the optimization space and write a formula that defines the objective function. The rules of the formula writing are typical for mathematical programs and are given in [19]. The method of setting the target function is shown in the text box with the captions “Use the function code” or “Use formula”.

The dimension of the optimization space is specified when the function is set.

In the program you can specify the output formats of the objective function value and point coordinates (see [19]).

You can write comments to the problem to be solved. Comments are recorded in the text format and saved in a file along with parameters and search results.

The program uses a pseudo-random number generator to perform the search. It can be initialized with either a value that depends on the system time of the computer or a specified value.

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

Setting the value of the parameter “Number of search steps $N$” equal to zero the program will calculate the value of the function at the starting point of the search. This can be used to calculate the value of the objective function at a given point.

Let us note that the use of the old development environment Microsoft Visual Studio 2010 when writing a program allows even users of computers with the operating system Windows XP to work with the project.

6. Search parameters selection

It is important to note that the choice of the search parameters can have a major impact on the effectiveness of the random search method [3, 5, 7]. However, many search algorithms contain a large number of heuristic parameters, and the user of such an algorithm can be very difficult to find “good” parameter values that are suitable for the optimized function (see, for example, [7]).

The presented search has only three parameters. Parameters specifying the trial transition function are the positive numbers $\nu$ and $\Gamma$ that should run inequality $0 < \nu \leq \Gamma$. The third parameter is $N$ – the number of search steps.

The value $\nu$ can be chosen close to the required accuracy of the solution of the problem when approximating the argument. The value $\Gamma$ can be chosen close to either the expected accuracy of the initial
approximation (the distance from the initial search point to the minimum point) or to the diameter of the study area in the optimization space.

The number of search steps \( N \) is desirable to be large enough. When solving a single problem, you can, for example, perform a billion of search steps, even if the task is simple enough and it can be solved much faster. Modern personal computers can easily perform similar volumes of calculations, at least for not too complex objective functions. However, for such volumes of calculations the code of the objective function must be set programmatically in the C# programming language.

In addition to the three search parameters, you need to select the starting point of the search. It is obvious that it is better to locate the starting point closer to the point of global extremum.

The proposed search algorithm is largely free from the insurmountable difficulties of choosing parameters. In particular, in the numerical examples of the next section, a minimal selection of parameters was carried out, consisting literally of several attempts to launch a program with different values of parameters.

### 7. Examples of the program use

Here are some examples of using the program to solve optimization problems. A personal computer with Intel Core i5–4460S processor was used for calculations.

#### 7.1. Example 1

Let’s use an example from [5]. Here is the optimization space \( X = \mathbb{R}^2, x = (x_1, x_2) \),

\[
f(x) = f(x_1, x_2) = x_1^4 + x_2^2 + x_1x_2 + x_2^2.
\]

The function \( f \) takes the minimum value at a single point \( x_* = (0, 0) \) and \( f(x) = 0 \). The starting point of the search is \( x = (1, 1) \) and \( f(x) = 4 \). The number of search steps \( N \) here is \( 10^4 \).

Algorithm B of the book [5] gets the minimum value of the objective function \( 2.7 \times 10^{-6} \). Algorithm B corresponds to the search for algorithm 1 using the normal probability distribution as a transition function.

Algorithm C of the book [5] gets the minimum value of the objective function \( 2.5 \times 10^{-7} \). Algorithm C also uses the normal probability distribution as a transition function, but represents a more complex search variant, in which the movement made in the previous step of the algorithm is taken into account when constructing a new search point. The homogeneous search algorithm of the article [19] gets the minimum value of the objective function \( 9.9 \times 10^{-49} \).

Algorithm 1 of this work with parameters \( \nu = 10^{-24} \) and \( \Gamma = 1 \) gets the minimum value of the objective function \( 6.0 \times 10^{-42} \).

In this example the search for algorithm 1 turned out to be much more accurate than the algorithms B and C from the book [5], which use the normal probability distribution. Meanwhile the results of the algorithm are a bit worse than article search results [19].

Algorithm 1 of this work with parameters \( \nu = 10^{-165}, \Gamma = 1 \) and \( N = 10^6 \) receives the minimum value of the objective function equal to zero (i.e. less than the value \( 5 \times 10^{-324} \) that determines the range of values of the type double of the C# programming language) and the minimum point \( (8.2 \times 10^{-163}, -1.8 \times 10^{-163}) \). We note that in this case the maximum accuracy with which you can perform calculations in C# using the numeric format double is achieved (due to the fact that it is impossible to calculate the value of the objective function more accurately).

The article search algorithm [19] also required \( 10^6 \) steps to obtain the value of the objective function equal to zero. The results of the algorithm 1 are similar to the results of the article search [19].

#### 7.2. Example 2

The optimization space is \( X = [-8, 8]^2, x = (x_1, x_2) \),

\[
f(x) = f(x_1, x_2) = \frac{1}{2} \left( (x_1^4 - 16x_1^2 + 5x_1) + (x_2^4 - 16x_2^2 + 5x_2) \right).
\]
The function $f$ has four local minima, one of them is global. The starting point of the search is $x = (4.0, 6.4)$ and $f(x) = 537.18$. Search algorithm 1 with the parameters $\nu = 10^{-7}$, $\Gamma = 10$ and $N = 20000$ finds the minimum value of the objective function $-78.3323314075428$ and the minimum point $(-2.903534, -2.903534)$. We note, that here we have reached the limit of accuracy with which you can perform calculations in C# using the numeric format double (because of the fact that it is impossible to accurately calculate the value of the objective function).

The obtained results are close to the results of a homogeneous search of the article [19].

7.3. Example 3
The space is $X = [-4, 4]^10$, $x = (x_1, x_2, ..., x_{10})$,

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

The $f$ function is a well-known Rosenbrock test function used for local optimization methods. The function takes the minimum value $f(x_*) = 0$ at the point $x_* = (1, 1, ..., 1)$. The starting point of the search is $x = (-1.2, 1, -1.2, 1, ..., 1)$ and $f(x)$ is 121. Search algorithm 1 with the parameters $\nu = 10^{-30}$, $\Gamma = 4$ and $N = 10^7$ finds the minimum value of the objective function $4.4 \times 10^{-18}$. The run time of the search was 6 seconds.

In this example algorithm 1 results are slightly worse than the article search results [19]. In addition, the run time of the search [19] was 1.8 second. It is more complicated to simulate the used in the test search probability distribution than to simulate the distribution from the article [19].

7.4. Example 4
Let us consider an example with a very simple objective function, but in the optimization space of a very large dimension for the random search methods. Here is the space $X = \mathbb{R}^{1000}$, $x = (x_1, x_2, ..., x_{1000})$, $f(x) = \sum_{n=1}^{1000} x_n^2$. The function $f$ takes the minimum value $f(x_*) = 0$ at a single point. The starting point of the search is $x = (1, 1, ..., 1)$. Search algorithm 1 with the parameters $\nu = 10^{-10}$, $\Gamma = 10^{-3}$ and $N = 10^6$ finds the minimum value of the objective function 0.00038. The run time of the search was 59 seconds.

Homogeneous search of the article [19] obtained the minimum value of the target function $1.6 \times 10^{-14}$ when $N = 10^6$. The search run time was 13 seconds. The obtained results are close to the results of a homogeneous search of the article [19] but the search for algorithm 1 is somewhat slower because of the complexity of the modeling of the Ingber distribution.

8. Conclusion
The results show that the presented algorithm of homogeneous search based on the use of the Ingber distribution is quite effective. The presented random search program can be successfully used to solve optimization problems. The program itself is easy to use and the choice of search parameters is not a difficult task. Meanwhile, the program allows you to solve problems with the utmost accuracy that can be obtained by using the double numeric format of the C# programming language.

References
[1] Ermakov S M and Zhigljavsky A A 1983 On random search of global extremum Probability theory and its applications 1 129–136
[2] Ermakov S M, Zhigljavsky A A and Kondratovich M V 1989 On comparing some of the 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 Annealing Method Stochastic optimization in computer science 1 133-149
[8] Granichin O N and Polyak B T 2003 Randomized estimation and optimization algorithms under almost arbitrary noise (Moscow: Science)
[9] Syshkov Yu A 1972 On one way of organizing a random search Operations research and statistic modeling (Leningrad: LSU) 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 A S Tikhomirov 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 2009 On fast variants of the annealing algorithm (simulated annealing) Stochastic optimization in computer science 5 65–90
[16] Tikhomirov A S 2010 On the convergence rate of the simulated annealing algorithm Computational Mathematics and Mathematical Physics Vol 50 1 19–31
[17] 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
[18] 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
[19] 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
[20] Tikhomirov A S 2016 Computer program «Markov Monotonous Search» Certificate of state registration of computer programs no. 2016663645