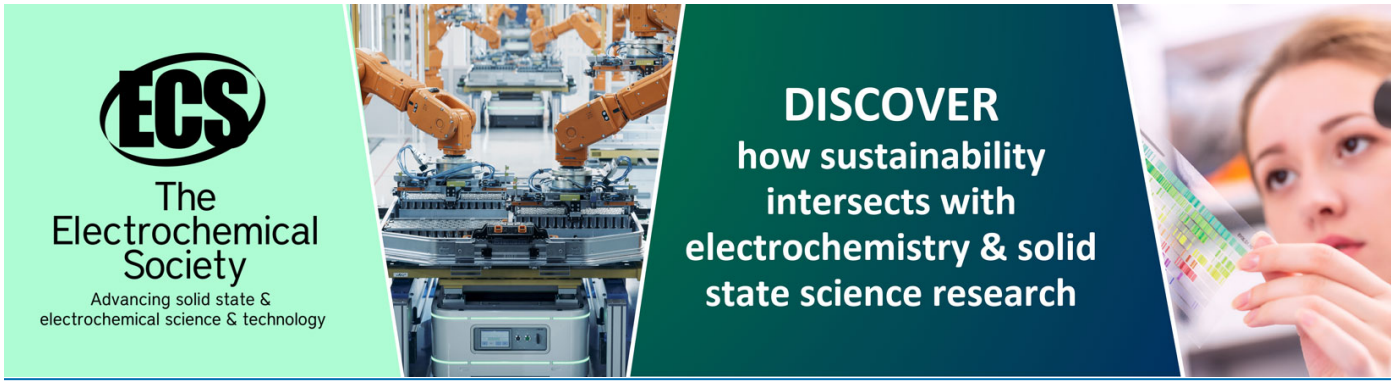
On the program implementation of one inhomogeneous Markov algorithm of search for extremum



IOP Conf. Series: Journal of Physics: Conf. Series **1352** (2019) 012054 doi:10.1088/1742-6596/1352/1/012054

On the program implementation of one inhomogeneous Markov algorithm of search for extremum

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 inhomogeneous monotonous random search algorithm of an extremum 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 the minimum value at a single point x_*. Consider the problem of finding the global minimizer x_* of this function up to a given accuracy $\varepsilon > 0$. One way of solving this problem is to use random search algorithms (see [1–16]). Such algorithms have long been successfully used for solving difficult optimization problems. Theoretical studies of the convergence rate of some Markov search algorithms are represented in [3, 10–15]. This paper is a continuation of [11, 12, 15] and is devoted to a computer program that implements one of the algorithms for the inhomogeneous Markov monotone search for an extremum. The presented computer program complements the program [16], in which one homogeneous random search algorithm was implemented.

2. Statement of the problem

As the *optimization space* we will consider the space $X = \mathbb{R}^d$ with the metric

$$\rho_\infty(x, y) = \max_{1 \le n \le d} |x_n - y_n|, \tag{1}$$

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

The metric ρ_∞ is chosen for the reasons of simplicity of the modeling of the considered random search. The fact is that the simulation of the undertaken search is based on the simulation of uniform distributions in the balls. The ball in the metric (1) is a cube. It is easy to model a uniform distribution in a multidimensional cube, while it is difficult to effectively model a uniform distribution in a "normal" ball defined by a Euclidean metric.

We will use *inhomogeneous Markov monotone random search* (see [11, 12, 15]) to find the global minimizer x_* 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 η with distribution P". For numbers and points in the optimization space, operations of the form $k \leftarrow 1$ and $\xi \leftarrow x$ denote the simple assignment operations.

Algorithm 1

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

Step 2. $\eta_k \leftarrow P_k(\xi_{k-1}, \cdot)$.
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 $k = N$, then finish the algorithm.
Step 5. $k \leftarrow k + 1$ and go to Step 2.

Here x is the starting point of search, N is the number of search steps, $P_k(x, \cdot)$ — Markov transition functions (see [11, 12, 15]).

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 k is equal to one (operator $k \leftarrow 1$).

At the second step of algorithm 1 we get a new "trial" point η_k in the optimization space. We choose the new "trial" point randomly using the distribution $P_k(\xi_{k-1}, \cdot)$. The distribution $P_k(\xi_{k-1}, \cdot)$ depends on the number of the search step k and location of the "old" search point ξ_{k-1}. This dependency improves the efficiency of random search. The transition function $P_k(x, \cdot)$ will be called *trial transition function.* The [16] considered the search, the trial transition functions of which did not depend explicitly on the step number k (i.e. had the form $P_k(x, \cdot) = P(x, \cdot)$). This search is called homogeneous. Here we will consider an inhomogeneous search whose trial transition functions $P_k(x, \cdot)$ explicitly depend on the number of the step k. Due to the inhomogeneity we can improve the efficiency of the search. But this dependence complicates the choice of search parameters and the "right" choice of search parameters is a difficult task (see, for example, [7]).

At the third step of algorithm 1, we compare the new test point η_k with the old search point ξ_{k-1}. If the new test point η_k is not worse than the old search point ξ_{k-1} (i.e. if the inequality $f(\eta_k) \le f(\xi_{k-1})$ is satisfied), the search moves to the new point η_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 $f(\xi_k) \le f(\xi_{k-1})$ is held for all $k \ge 1$.

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 $P_k(x, \cdot)$. 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 $P_k(x, \cdot)$ should be simple enough.

In [16], a homogeneous search was considered, its trial transition functions did not depend explicitly on the number of the step k (i.e. had the form $P_k(x, \cdot) = P(x, \cdot)$). So there we had to choose one transition function. Here we need to select N different transition functions, where N is the number of search steps. It is clear that this greatly complicates the task. In addition, for a homogeneous search in [13], estimates of labor intensity were obtained and studied. These results were used to select a trial transition function for homogeneous search. Unfortunately, there are no such general theoretical results for inhomogeneous search.

We consider a random search for algorithm 1 whose trial transition functions $P_k(x, \cdot)$ have symmetric densities of the form

$$p_k(x, y) = g_k\big(\rho_\infty(x, y)\big), \tag{2}$$

where ρ_∞ — metric, and g_k — non-increasing nonnegative functions defined on the half-axis $(0, +\infty)$.

Let $\mathcal{P}$ be the set of all transition functions with symmetric densities of the form (2). For $P, Q \in \mathcal{P}$ let consider that

$$\lambda(P,Q)=\sup_{x\in X}\sup_{A\in\mathcal{F}}|P(x,A)-Q(x,A)|. \tag{3}$$

We will consider the set $\mathcal{P}$ as a metric space with metric (3).

The simplest of such distributions with symmetric densities of the form (2) is the uniform distribution $U_a(x,\ \cdot\ )$ in the ball $B_a(x)$ of radius $a>0$ with the center at the point $x\in X$,

$$U_a(x,\ \cdot\ )=\mu\big(\ \cdot\cap B_a(x)\big)/\mu\big(B_a(x)\big). \tag{4}$$

Through U_a we will denote the transition function corresponding to the family of distributions $U_a(x,\ \cdot\ )$, $x\in X$. Let us consider that $\mathcal{U}=\{U_a\colon\ a>0\}$. It is clear that $\mathcal{U}\subset\mathcal{P}$. It turns out that $\mathcal{P}=\mathcal{Z}(\mathcal{U})$, where $\mathcal{Z}(\mathcal{U})$ is the closure in the metric (3) of the set of all possible convex linear combinations of transition functions from $\mathcal{U}$ (see [12]).

The search of algorithm 1, trial transition functions which belong to the set $\mathcal{P}$, will be called the inhomogeneous Markov monotone symmetric random search.

Let us consider the inhomogeneous Markov monotone symmetric random search with starting point $x\in X$ and trial transition functions $P_k\in\mathcal{P}$. We use the random search for finding some measurable subset A of the set $X=\mathbb{R}^d$. For example, we can take $A=B_\varepsilon(x_*)$ or $A=\{x\in X\colon\ f(x)\le f(x_*)+\varepsilon\}$ when $\varepsilon>0$. In this case we are limited by a fixed number N of random search steps and fixed transition functions $P_1,\dots,P_N$. Note that although the set A may not be related to the objective function f, the search itself depends on f.

Let us pose the question: is it possible to replace the transition functions $P_1,\dots,P_N$ from the set $\mathcal{P}$ with some "simple" transition functions so that for a new search (which still starts at a point $x\in X$) the probability $P_x(\xi_N\in A)$ to be at the step N in the set A does not decrease? It turns out (see [12]) that it is possible if the uniform distributions in balls of various radii with various centers are considered as "simple" transition functions. As it was already mentioned, in the space $X=\mathbb{R}^d$ with the metric ρ_∞ such distributions are easily modeled.

Emphasizing the dependence of the probability $P_x(\xi_N\in A)$ from the transition functions $P_1,\dots,P_N$ from the set $\mathcal{P}$ we will denote it $P_x(\xi_N\in A;P_1.\dots,P_N)$ .The following statement is true

Theorem 1. Let the objective function f, the set A, the number of search steps N and the starting point of the search $x\in X$ be fixed. Then

$$\begin{gathered}\sup\{P_x(\xi_N\in A;P_1,\dots,P_N)\colon P_1,\dots,P_N\in\mathcal{P}\}=\\ \sup\{P_x(\xi_N\in A;P_1,\dots,P_N)\colon P_1,\dots,P_N\in\mathcal{U}\}.\end{gathered} \tag{5}$$

Under additional constraints the existence of transition functions $P_1,\dots,P_N\in\mathcal{U}$ that realize the supremum in the right side (5) is proved in [12].

Theorem 1 proves that the optimal random search has a simple structure. To find the optimal Markov monotone symmetric random search, we do not need to search for a set of functions (a set defined by the formula (2) of densities), but rather to find a set of numbers (radii of balls for uniform distributions). However, obtaining optimal search parameters is a complex task. One of the main difficulties is that these parameters depend on the objective function and when solving practical problems they have to be chosen taking into account a priori information about the behavior of the objective function that is difficult to formalize. It is clear that such problems of optimal choice of parameters are difficult not only for the solution, but even for the formulation. Therefore, there are no theoretical results on the optimal choice of ball radii.

Taking into account the presented results, we will use the inhomogeneous Markov monotone symmetric random search whose trial transition functions are uniform distributions in balls (given by the formula (4)).

To build the search, we need to choose the radii of the balls for the uniform distributions given by the formula (4). To select these radii, we will use heuristic considerations. Consider the homogeneous monotonous random search, trial transition function $P(x,\ \cdot\ )$ which minimizes studied in [13] estimate of the complexity of random search when optimizing a simple objective function. Such search has several

advantages. This search (for non-degenerate objective functions) provides a good order of dependence of the obtained labor intensity estimates from ε (see [3, 13]). And given examples of using the program [16] show acceptable efficiency in the optimization of not too complex objective functions. The trial transition function $P(x, \cdot)$ of such a homogeneous search is close to the mixture with the same probabilities of uniform distributions in balls whose radii form a geometric progression. The radii of this geometric progression run through the values from the assumed accuracy of the initial approximation (the distance from the initial search point to the minimum point) to the required accuracy of the solution of the problem when approximating by argument. Therefore, in the inhomogeneous search that is under study, we will use uniform distributions in balls whose radii form a geometric progression. Only, in contrast to the homogeneous search, in the inhomogeneous search we will not use a mixture of uniform distributions in the balls, and we will use these uniform distributions consistently. We start naturally with the largest radius and finish with the smallest.

In addition, as the number of search steps should be large enough so that we don't have to change the radius of the ball too often, we will split the whole search into stages, and we will change the radius of the ball only after the completion of the current stage. Thus, the trial transition functions will be constant for all the steps of one stage.

Theoretical results on the choice of parameters of the considered inhomogeneous search are presented in [11, 15]. Let $N(\varepsilon, \gamma)$ be the number of search steps at which the achievement of the neighborhood of the minimum point is guaranteed with probability γ (more precisely, when $N = N(\varepsilon, \gamma)$ the inequality $P_x\big(\xi_N \in B_\varepsilon(x_*)\big) \geq \gamma$ is fulfilled). In [15] a method for selecting search parameters (radii $a_1, \dots, a_N$) is described in which $N(\varepsilon, \gamma) = O(|\ln \varepsilon| \times \ln|\ln \varepsilon|)$ is performed for a non-degenerate objective function. In [14] it is shown that the inequality $N(\varepsilon, \gamma) \geq \gamma(\ln(\rho(x, x_*)/\varepsilon) + 2)$ is true, where x is the initial point of the search, for any Markov symmetric random search of algorithm 1, whose trial transition functions have densities of the form (2). Therefore, we can assume that the search described in [15] has a good order of dependency of $N(\varepsilon, \gamma)$ from ε.

Due to everything stated in this section, we choose to implement the inhomogeneous Markov monotone symmetric random search whose trial transition functions are uniform distributions in balls. The whole search will be divided into stages and trial transition functions will be constant for all the steps of one stage. We will change the radius of the balls only after the completion of the current stage of the search and these radii form a geometric progression.

4. Random search simulation

In this section we present a simulation algorithm chosen to implement an inhomogeneous Markov monotone random search. The presented search has only four parameters. The parameters that define the range of variation of the radii of the balls are positive numbers ν and Γ for which inequalities $0 < \nu \leq \Gamma$ must be performed. The third parameter N is the number of search steps. The fourth parameter is m— the number of steps at the search stage (trial transient functions are constant for all the steps of one stage). Parameters m and N are natural numbers that satisfy the inequalities $1 \leq m \leq N$.

We will calculate two auxiliary parameters. The number of stages of the search is $\tau = \lceil N/m \rceil$. The radii of the balls at the search stages form a geometric progression with the denominator of the progression $q \in (0,1]$, where

$$q = \begin{cases} 1, & \text{if } \tau = 1, \\ (\nu/\Gamma)^{1/(\tau-1)}, & \text{if } \tau > 1. \end{cases}$$

Now we write down the algorithm chosen for implementation inhomogeneous Markov monotone symmetric random search $\xi_0, \dots, \xi_N$.

Algorithm 2

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

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

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

Step 4. If $k = N$, then finish the algorithm.

Step 5. If $k \bmod m = 0$, then $a \leftarrow a * q$.

Step 6. $k \leftarrow k + 1$ and go to the Step 2.

Here x is the starting point of the search, k— the number of the search step, N— the number of search steps. Through $U_a(\xi_{k-1}, \cdot)$ denoted the uniform distribution in the ball $B_a(\xi_{k-1})$ of radius $a > 0$ centered at the point ξ_{k-1} given by the formula (4). The radius a changes after the end of the search stage, where m— the number of steps in the search stage, and through $k \bmod m$ the remainder of the division k by m is denoted.

At the second step of algorithm 2 we obtain a new "trial" point η_k in the optimization space using a uniform distribution $U_a(\xi_{k-1}, \cdot)$.

At the third step of algorithm 2 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 the inequality $f(\eta_k) \le f(\xi_{k-1})$ is satisfied) the search moves to the new point (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 2, we check the condition for stopping the search. In this case, the search stops after performing a predetermined number of steps N, i.e. under the condition $k = N$. If the search continues (i.e., if $k < N$), then at the fifth and sixth steps of algorithm 2 we recalculate the values of the search parameters and return to the second step of algorithm 2.

Since in the space $X = \mathbb{R}^d$ with the metric ρ_∞ it is very simple to model the uniform distribution $U_a(\xi_{k-1}, \cdot)$, and in general, the simulation algorithm 2 is very easy to program. The presented simulation algorithm is only slightly more complicated than the algorithm for modeling a simplest random search (the so-called "blind search" [3, 5]), which uses a uniform distribution in a pre-fixed area of optimization space. Simulation algorithm 2 is simpler than most of the used random search simulation algorithms [3, 5–7].

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/tas1 from the "Random_search" folder. The program is also available as an executable file MarkovMonotonousSearch.exe 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 approximation by the argument of the order 10^{-8}, we obtain the accuracy of the approximation by the function value of the order 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}. Thereby, the typical accuracy of the solution of the task will be of the order of 10^{-7} in the approximation of the argument and of the order of 10^{-14} in the approximation of the function value. 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 an objective 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 [16]. 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 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 [16].

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 [16]).

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.

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 objective function. Here is a quote from [7] relating to the method of very fast annealing proposed by L. Ingber: "One of the disadvantages of this method is that due to the large number of parameters, it sometimes takes several months to set it up well for a specific task". At the same time, with proper selection of parameters, the method of very fast annealing can show very good results [6, 7].

The presented search has only four parameters. The parameters defining the range of changes of the radii of the balls are positive numbers ν and Γ for which inequalities $0 < \nu \leq \Gamma$ must be satisfied. The third parameter is N— the number of search steps. The fourth parameter is m— the number of steps at the search stage (trial transition functions are constant for all steps of one stage).

The value ν can be chosen close to the required accuracy of the solution of the problem when approximating the argument. The value Γ 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.

The parameter m (the number of steps at the search stage) can be set so that it is not necessary to change the ball radius too often. All the numerical examples in the next section use the value $m = 10$.

In addition to the four 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 $X = \mathbb{R}^2$, $x = (x_1, x_2)$,

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

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 2.7×10^{-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 2.5×10^{-7}. Algorithm C also uses the normal probability distribution as a transition function, but represents a more complex search variant, in which the displacement 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 [16] gets the minimum value of the objective function 9.9×10^{-49}.

Algorithm 2 of this work with parameters $\nu = 10^{-165}$, $\Gamma = 1$ and $m = 10$ gets the minimum value of the objective function equal to zero (i.e. less than the value 5×10^{-324} that determines the range of values of the type double of the C# programming language) and the minimum point $(-5.0 \times 10^{-163}, 1.3 \times 10^{-163})$. 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).

Algorithm 1 of homogeneous search [16] was required to take 10^6 steps to obtain the zero value of the objective function.

In this example, the search for algorithm 2 turned out to be much more accurate than the algorithms B and C of the book [5] and algorithm 1 of the article [16].

7.2. Example 2

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

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

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 2 with the parameters $\nu = 10^{-9}$, $\Gamma = 10$, $m = 10$ 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 [16].

7.3. Example 3

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

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

The f function is a well-known Rosenbrock test function used for local optimization methods. The function f takes the minimum value $f(x_*) = 0$ at the point $x_* = (1, 1, \dots, 1)$. The starting point of the search is $x = (-1.2, 1, -1.2, 1, \dots, 1)$ and $f(x) = 121$. Search algorithm 2 with the parameters $\nu = 10^{-17}$, $\Gamma = 4$, $m = 10$ and $N = 10^7$ finds the minimum value of the objective function 3.7×10^{-29}. The run time of the search was 1.5 seconds.

The obtained results are close to the results of a homogeneous search for algorithm 1 of [16].

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 $X = \mathbb{R}^{1000}$, $x = (x_1, x_2, \dots, x_{1000})$, $f(x) = \sum_{n=1}^{1000} x_n^2$. The function f takes the minimum value $f(x_*) = 0$ at a single point. The starting point of the search is $x = (1, 1, \dots, 1)$. Search algorithm 2 with the parameters $\nu = 10^{-80}$, $\Gamma = 10$, $m = 10$ and $N = 10^6$ finds the minimum value of the objective function 1.2×10^{-155}. The run time of the search was 13 seconds.

A homogeneous search of the article [16] obtained the minimum value of the objective function 1.6×10^{-14} when $N = 10^6$. The search run time was 13 seconds.

In this example, the search for algorithm 2 turned out to be much more accurate than the search for algorithm 1 of article [16].

8. Conclusion

The obtained results show that 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. At the same time, the program allows you to solve problems with the utmost precision that can be obtained using the double number 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 Simulated annealing *Stochastic optimization in computer science* **1** 133–149
[8] Granichin O N and Polyak B T 2003 *Randomized algorithms of estimation and optimization 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 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 1998 Optimal Markov monotonic symmetric random search *Computational*

Mathematics and Mathematical Physics Vol **38 12** 1894–1902
[13] Tikhomirov A S 2006 On the Markov homogeneous optimization method *Computational Mathematics and Mathematical Physics* Vol **46 3** 361–375
[14] 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
[15] 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
[16] 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