symmetry

Article

The Algorithm of Continuous Optimization Based on the Modified Cellular Automaton

Oleg Evsutin *, Alexander Shelupanov, Roman Meshcheryakov, Dmitry Bondarenko and Angelika Rashchupkina

Tomsk State University of Control Systems and Radioelectronics, 40 Lenina Prospect, Tomsk 634050, Russia; saa@keva.tusur.ru (A.S.); mrv@keva.tusur.ru (R.M.); dima030793@gmail.com (D.B.); angelinara@mail.ru (A.R.)
* Correspondence: eoo@keva.tusur.ru; Tel.: +7-3822-90-01-11

Academic Editors: Ka Lok Man, Yo-Sub Han and Hai-Ning Liang
Received: 20 March 2016; Accepted: 22 August 2016; Published: 25 August 2016

Abstract: This article is devoted to the application of the cellular automata mathematical apparatus to the problem of continuous optimization. The cellular automaton with an objective function is introduced as a new modification of the classic cellular automaton. The algorithm of continuous optimization, which is based on dynamics of the cellular automaton having the property of geometric symmetry, is obtained. The results of the simulation experiments with the obtained algorithm on standard test functions are provided, and a comparison between the analogs is shown.

Keywords: continuous optimization; metaheuristics; cellular automata

1. Introduction

The problems of optimization in different application areas often appear when choosing the best variant out of a majority of possible ones is required.

Nowadays, many methods for solving similar tasks are known. It is possible to divide them into two big classes: deterministic and stochastic. The deterministic methods let us guarantee the optimality of a solution with the given accuracy, while the stochastic methods in the common case do not give us information about the accuracy of the found solution. Nevertheless, they do show a relatively high effectiveness in optimizing complex objective functions in practice. Consequently, the methods of optimization, which appertain to the second class, are also called heuristic.

The major part of the stochastic methods is based on the search for the optimal solution which imitates the behavior of the complicated physical, biological, or social systems consisting of a considerable number of intercommunicate homogeneous elements [1–4]. This article suggests drawing upon the cellular automaton, which is an object of discrete mathematics characterized by different symmetry properties, both in geometrical and algebraic meaning. In our case, we used the cellular automaton which has the property of geometrical symmetry on the level of local rule of evolution.

It is necessary to note that there have appeared a variety of research works concerning hybridization of the cellular automata mathematical apparatus with different metaheuristics.

In papers [5,6], cellular automata are used together with genetic algorithms. Traditionally, chromosomes which are operated by genetic algorithms are represented in the form of irregular sets of binary vectors. In [5] they are recorded in cells of the two-dimensional cellular automaton, and during crossing over each chromosome takes genetic material from neighbors from the Moore neighborhood. The mutation is realized in the usual way. An additional operation is the shuffle of chromosomes in the lattice for improvement of genetic material interchange.

Paper [6] is devoted to an inverse problem. In that paper the cellular automaton serves to model the groundwater allocation, and a genetic algorithm is applied for the adjustment of the model. In this case, states of a lattice of the cellular automaton are coded by means of chromosomes of the genetic algorithm, and the ultimate goal is the deriving of an optimum state. In article [7] by the same author, the results of a similar research are presented, the research differing only in the harmony search application.

In some other papers, cellular automata are merged with the algorithms of optimization which refer to swarm intelligence. So, in paper [8], a modified ant colony algorithm is presented, in which a many-dimensional search space is considered as a cellular structure. For this purpose, a quantity of vectors is picked from the search space, and each of the vectors is called a cell. The points of the space standing apart from the given cell, when this distance does not exceed a certain value, organize a neighborhood. A cell with its neighborhood is called a region. In the given paper, cellular evolution is perceived as the motion of ants in the search space and the ants realize optimum search in each region and can pass from one region to another.

In [9] the cellular automata approach is applied to improve the particle swarm optimization algorithm. For this purpose, the authors of the paper introduce the concept of particle neighborhood which is perceived as an assemblage of particles of the swarm closest to the given particle. This concept is used for modification of the particles velocity calculation equation: instead of a randomly chosen element of a swarm, the best particle of the neighborhood is introduced. Besides, a special feature of the given research consists in specificity of the considered problem of optimization. The offered in [9] hybrid algorithm is used for optimization of truss structures.

A similar direction is presented in papers [10–12]. The solution of structural optimization problems is also considered. The cellular automaton serves for modelling of the structure which is necessary to optimize. Optimization consists in obtaining such configuration of the cellular automaton which will correspond to the best parameters of the modelled structure. Thus, in this case cellular automata are simultaneously applied as an optimization means and as a modelling environment.

The approach used in [13] for improvement of the membrane algorithm is similar to the approach used in [5] with reference to the genetic algorithm. The computing model on which the membrane algorithm is based represents a membrane system. Its principal components are membrane structure, reaction rules, and multisets. In [13], the elements of a multiset contained in the separately taken elementary membrane are represented in the form of a two-dimensional lattice, and at renovation of a state of each element, states of the neighboring elements from the Moore neighborhood are considered.

However, in the research provided, cellular automata are used indirectly only, mainly by transmitting their peculiar principle of local interaction to the basic metaheuristic model, which does not bring out the best of the given mathematical apparatus in solving optimization problems.

The principle of local interaction is one of the basic parameters of the cellular automaton: the state of all cells renovates at each step of the automaton development, depending on the state of their neighbors [14]. Consequently, in the majority of the papers indicated, the use of the cellular automaton is, in the variety of the solutions under study, limited to introducing a notion of the neighborhood area and to considering vectors from the neighborhood area while a new state of each vector containing the solution options is being calculated.

Unlike the previous papers, in paper [15] cellular learning automata are used. In the given paper, they are merged with an artificial fish swarm algorithm, imitating the behavior of a fish swarm in search of food. As a base computing model, using a one-dimensional learning cellular automaton consisting of D cells according to the dimensionality of the optimization problem being solved is suggested. Each cell of such automaton represents a separate n-dimensional learning automaton, associated with an artificial fish swarm responsible for optimization of one measurement of the objective function.

Development of the given paper is an original algorithm presented in [16]. It is based on integration of differential evolution and a computing model of learning cellular automata which does not refer to nature-inspired models. In more details, the given algorithm will be considered in the following section of this article.

Our paper develops the application of the cellular automata as the mathematical tool to the problem of continuous optimization. As distinct from the well-known approaches, it is suggested to use the dynamics of the cellular automata for the immediate optimum search in the solution space. Also added is the modification of the classic model of the cellular automaton on the basis of which the optimization algorithm is being formulated and studied.

2. Methods

2.1. Optimization on the Basis of Learning Cellular Automata

In the given section we will consider the optimization method offered in [16] in more detail. This method is based on cellular learning automata and differential evolution algorithm (CLA-DE). The CLA-DE method differs from the analogs considered in introduction in that it is not a modification of any known metaheuristics. The differential evolution declared in the title is used not as a basis, but only for improvement of the base calculation model.

The basis of the CLA-DE method is the computing model of cellular learning automata which merges cellular automata with learning automata.

A learning automaton is an adaptive decision-making system situated in an unknown random environment. This system learns the optimal action through repeated interactions with its environment. The learning is as follows: at each step, the learning automaton selects one of its actions according to its probability vector and performs it on the environment. The environment evaluates the performance of the selected action to determine its effectiveness and, then, provides a response for the learning automaton. The learning automaton uses this response as an input to update its internal action probabilities.

The learning automaton can be defined as $\langle \Theta, \alpha, \beta, A, G, \mathbf{P} \rangle$, where Θ is a set of internal states, α is a set of outputs, β is a set of input actions, A is a learning algorithm, $G : \Theta \to \alpha$ is a function that maps the current state into the current output, and $\mathbf{P}$ is a probability vector that determines the selection probability of a state at each step. The learning algorithm modifies the action probability distribution vector according to the received environmental responses. In [16], the linear reward penalty algorithm is used for automaton learning.

A learning cellular automaton represents a cellular structure in each cell of which a set of learning automata is located. The cell state is considered to be the total of states of learning automata contained in it. Each cell is evolved during the time based on its experience and the behavior of the cells in its neighborhood. The rule of the cellular learning automaton determines the reinforcement signals to the learning automata residing in its cells, and based on the received signals each automaton updates its internal probability vector.

The CLA-DE method built on the basis of the presented computing model is given below.

```
Step 1. Initialize the population: initial elements of the search space are generated randomly;
        probabilities of all actions of each learning automaton are accepted as equal.
Step 2. If the condition of stop is not reached, synchronously update each cell based on Model
        Based Evolution.
    Step 2.1. For each cell with candidate solution $CS = \{CS_1, \ldots, CS_n\}$ and solution model
              $M = \{LA_1, \ldots, LA_n\}$ do:
        Step 2.1.1. Randomly partition $M$ into $l$ mutually disjoint groups $G = \{G_1, \ldots, G_l\}$.
        Step 2.1.2. For each nonempty group $G_i$ do:
        Step 2.1.3. Create a copy $CSC^i = \{CSC^i_1, \ldots, CSC^i_n\}$ of $CS = \{CS_1, \ldots, CS_n\}$ for $G_i$.
        Step 2.1.4. For each $LA_d \in G_i$ associated with the $d$-th dimension of $CS$ do:
            Step 2.1.4.1. Select an action from the action set of $LA_d$ according to its probability.
            Step 2.1.4.2. Let this action correspond to an interval like $[s_{d,j}, e_{d,j}]$.
            Step 2.1.4.3. Create a uniform random number $r$ from the interval $[s_{d,j}, e_{d,j}]$, and
                          alter the value of $CSC^i_d$ to $r$.
        Step 2.1.5. Evaluate $CSC_i$ with objective function.
    Step 2.2. For each cell do:
        Step 2.2.1. Create a reinforcement signal for each one of its learning automata.
        Step 2.2.2. Update the action probabilities of each learning automaton based on its
                    received reinforcement signal.
        Step 2.2.3. Refine the actions of each learning automaton in cell.
Step 3. If generation number is a multiple of 5 do:
    Step 3.1. Synchronously update each cell based on DE Based Evolution.
Step 4. Return the best solution found.
```

According to the given method, a separate cell of the learning cellular automaton lattice contains a candidate solution (an n-dimensional vector from the search space) and a solution model. The solution model represents a set from n learning automata. Each of them operates the modification of one element of the vector contained in the cell. The action of a learning automaton is understood as the transition to one of the non-overlapping segments into which the corresponding measurement of the search space is divided.

During calculations for each cell some modified copies of a solution alternative are created according to the set learning algorithm. The best solution alternative of the newly created ones is taken as the new state of the cell.

As authors of the presented method note, the combination of cellular automata with learning automata allows cells to learn and thus to co-operate among themselves. The use of differential evolution allows enhancement of the influence of cells on each other thereby raising effectiveness of their learning.

The basic idea of our paper is the direct use of cellular automata for continuous optimization. Also we should note that, of those presently available, the CLA-DE method is the most significant approach to the implementation of this idea. All other methods found during the literature review take only a principle of local interacting from the concept of cellular automata, as has been noted earlier. Therefore the CLA-DE method is considered by us as a basic analog of our paper.

2.2. The Cellular Automaton with an Objective Function

Cellular automata are abstract models for systems consisting of a large number of identical simple components with local interactions.

Definition 1. *Let* $\langle Z^n, A, \mathbf{Y}, \sigma \rangle$ *be a cellular automaton, where* Z^n *is a cellular space;* A *is an set of internal states denoting a finite set of possible values of a cell;* $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k)$, $\mathbf{y}_i \in Z^n$, $i = \overline{1,k}$ *is a neighborhood pattern denoting a type of neighborhood, which is the same for every lattice cell; and* $\sigma : A^k \to A$ *is a local transition function (evolution rule).*

The local transition function is applied to all lattice cells during the evolution of a cellular automaton simultaneously. This function can be defined analytically or as a set of parallel substitutions. The cell neighborhood is regarded as a set of lattice cells, the current states of which will affect the state of a given cell in the next moment of time. A cell for which the neighborhood is constructed is called the central cell. Every neighborhood pattern element is a vector of the relative indices defining the position of a single neighborhood cell relative to the central cell, i.e., $y_i^j \in \{-r, \ldots, 0, \ldots, r\}$, $i = \overline{1,k}$, $j = \overline{1,n}$, where r is an integer, usually with a small value. For example, for classic Moore or Von Neumann neighborhoods, $r = 1$.

Also, here we should note that the neighborhoods of that type have the property of geometrical symmetry towards the central cell, as can be seen in Figure 1, and they are used in the experiments shown further down.

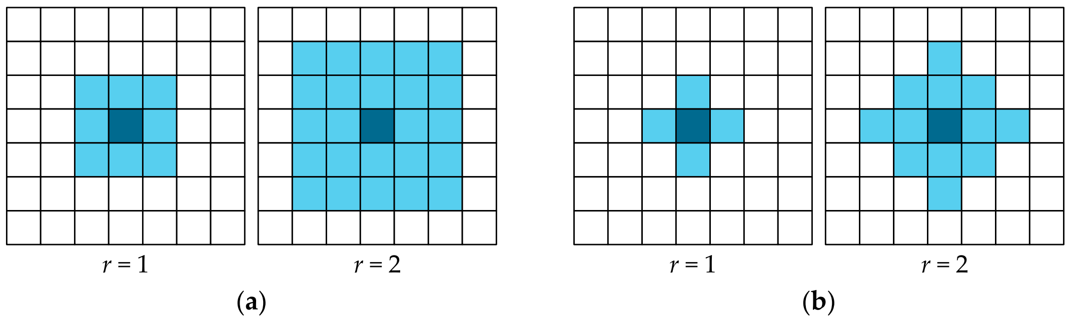


Figure 1. The symmetry of the cellular automaton neighborhood: (**a**) Moore neighborhoods; (**b**) Von Neumann neighborhoods.

Consequently, during the update of every lattice cell state, the states of k adjacent cells, whose position relative to the central cell is defined by the neighborhood pattern $\mathbf{Y}$, are taken to be σ function arguments [14,17,18].

Using a cellular automaton with a finite-sized lattice seems more natural in applied problems. In such a case, the model of a cellular automaton can be defined as $\langle Z^n, \mathbf{L}, A, \mathbf{Y}, \sigma \rangle$, where $\mathbf{L} = (l_1, \ldots, l_n)$, $l_i > 0$, $i = \overline{1, n}$ is a vector setting the lattice size. When accessing cells situated outside the lattice, their coordinates are resolved by mod l_i, $i = \overline{1, n}$, which is called "wrap-around" in terms of cellular automata.

Let us give a new extension of classic cellular automaton model.

Definition 2. *Let* $\langle Z^n, \mathbf{L}, A, \mathbf{Y}, \sigma, U, \Phi \rangle$ *be a cellular automaton with an objective function, where* Z^n, $\mathbf{L}$, *and* $\mathbf{Y}$ *correspond to analogical components of classic model; the set* A *is modified and looks like* $A = \Re^m \times D$, *where* D *is a set of labels;* $\sigma : (\Re^m \times D)^k \to \Re^m$ *is a local transition function;* $U : \Re^m \to D$ *is a cell marking-out rule; and* $\Phi = \Phi(x_1, \ldots, x_m)$ *is an objective function.*

Every lattice cell of such a cellular automaton contains the real values vector $\mathbf{x} \in \Re^m$ (which is regarded as an element of an objective function Φ domain) and an integer label which defines the quality of a solution contained in the $\mathbf{x}$ vector in terms of its proximity to the best solution acquired at this step of a cellular automaton evolution. Let us call the $\mathbf{x}$ vector the state of a cell. The local transition function can be defined only analytically, because it operates with real instead integer values.

The main features of the introduced model are shown in Figure 2. The lattice of the cellular automaton is supplemented with the separate plane containing values of an objective function for all cells.

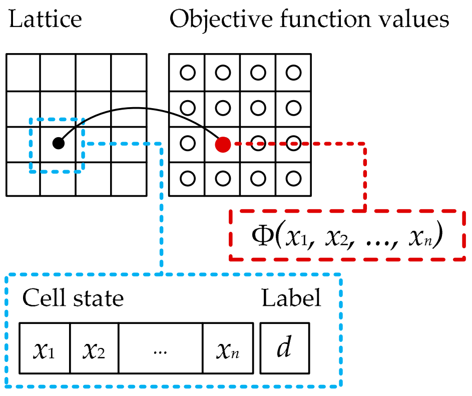
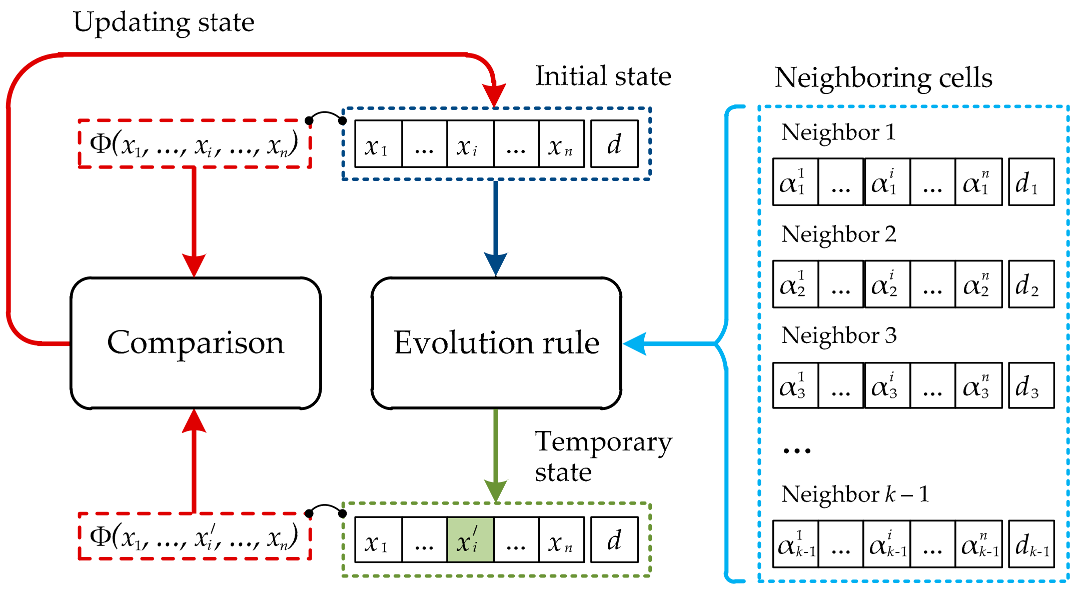


Figure 2. The cellular automaton with an objective function.

Figure 3 shows the general scheme, according to which the state of a separate cell of the lattice of the cellular automaton with an objective function is renovated.



Updating state
Initial state
Neighboring cells
$\Phi(x_1, \ldots, x_i, \ldots, x_n)$
x_1 ... x_i ... x_n d
Neighbor 1
α_1^1 ... α_1^i ... α_1^n d_1
Neighbor 2
α_2^1 ... α_2^i ... α_2^n d_2
Comparison
Evolution rule
Neighbor 3
α_3^1 ... α_3^i ... α_3^n d_3
Temporary state
Neighbor $k-1$
$\Phi(x_1, \ldots, x_i', \ldots, x_n)$
x_1 ... x_i' ... x_n d
α_{k-1}^1 ... α_{k-1}^i ... α_{k-1}^n d_{k-1}

Figure 3. Updating of the cell state.

Updating of the cell state is carried out under control of the objective function. Thus, separate measurements of the **x** vector, contained in the cell, vary sequentially and independently from each other. The new value of the *i*-th element of the **x** vector is calculated according to the chosen rule of evolution on the basis of the corresponding values contained in the neighboring cells belonging to the neighborhood of the given cell.

The most important element of the presented scheme is the rule of the cellular automaton development. Effectiveness of the offered model for solving the optimization problem depends on this rule. The rules of development introduced by us are presented in the following section of the given article.

2.3. The Procedure of the Cellular Automaton with an Objective Function Evolvement

For clarity, assume from now on that the evolution of a cellular automaton is pointed on the minimization of the objective function. The set of labels D is understood to consist of the two elements $\{0, 1\}$ and the marking-out rule for labeling cells U is defined as

$$U : (x_1, \ldots, x_m) \mapsto \begin{cases} 1, \text{if } \dfrac{\Phi(x_1, \ldots, x_m) - \Phi_{\min}}{\Phi_{\max} - \Phi_{\min}} \leq \mu; \\ 0, \text{otherwise}, \end{cases} \tag{1}$$

where $\Phi_{\min}$ and $\Phi_{\max}$ are the minimum and the maximum of the objective function under the current step of the cellular automaton evolution respectively; and $\mu \in (0, 1)$ is a preassigned parameter of the principle of cells labeling.

By means of the given rule the separate solutions contained in cells, divide into "bad" and "good" depending on their proximity to the best current solution. The extent of this proximity is defined by parameter μ. The label with the meaning "1" corresponds to the cells containing "good" solutions, and the label with the meaning "0" corresponds to the cells containing "bad" solutions. Cells with the label 0 are excluded from the neighborhood of the central cell in the following step of the evolution of the cellular automaton, or they are treated in a special way depending on a certain rule of evolution.

Let us define the two types of local transition functions of a cellular automaton with an objective function. We will use the following notation.

$[a, b]^m \subset \Re^m$ is the search space.

$\mathbf{x} \in [a,b]^m$ is the state of the central neighborhood cell in relation to which the local transition function is being recorded, and $d \in \{0,1\}$ is its label. The cell indexing from now on will be dropped for brevity.

$\alpha_1, \ldots, \alpha_k \in [a,b]^m$ is the state of the neighborhood cells, recorded in the corresponding neighborhood pattern order, and $d_1, \ldots, d_k \in \{0,1\}$ are the labels. The central cell also belongs to its neighborhood, thus for certain p, $1 \le p \le k$, $\alpha_p = \mathbf{x}$ and $d_p = d$.

$\hat{\mathbf{x}}^{\langle j \rangle} \in [a,b]^m$ is the real vector differing from $\mathbf{x}$ with j-th element value.

Then the local transition function of the first type is defined as

$$\sigma : \forall i = \overline{1,m}\ x_i \mapsto \begin{cases} \hat{x}_i^{\langle i \rangle}, \text{ if } \Phi\left(\hat{\mathbf{x}}^{\langle i \rangle}\right) < \Phi(\mathbf{x}); \\ x_i, \text{ otherwise,} \end{cases} \tag{2}$$

where $\forall\ i \neq j\ \hat{x}_i^{\langle j \rangle} = x_i$ and $\hat{x}_j^{\langle j \rangle} = \frac{\sum_{p=1}^{k} d_p W\left(x_j, \alpha_p^j, a, b, \omega\right) \alpha_p^j}{\sum_{p=1}^{k} d_p W\left(x_j, \alpha_p^j, a, b, \omega\right)}$, $\omega \in (0,1)$, $i = \overline{1,m}$, $j = \overline{1,m}$,

$$W(x_j, \alpha_p^j, a, b, \omega) = \begin{cases} 1, \text{ if } \alpha_p^j \in \left(x_j - \omega(b-a), x_j + \omega(b-a)\right); \\ 0, \text{ otherwise.} \end{cases}$$

Updating of cell states of the cellular automaton by means of the presented function of transition is carried out as follows. Every element of the vector located in the cell is updated independently from the other elements. The new value of the i-th element of a vector is calculated by neighborhood averaging. The modification is accepted only in the case that it has led to improvement of the solution contained in the cell. Otherwise, the value of the i-th element of the vector contained in the cell remains without modifications.

The new state of the central cell forms only on the part of the neighborhood cells which contains "good" solutions. The function W, in its turn, excludes from the computation the elements of the neighborhood cells which are significantly remote from the corresponding central cell elements in the search space. That is determined by the parameter ω.

Thus, the marking-out rule introduced earlier defines which cells of the neighborhood take part in the formation of a new state of the central cell and which do not. Function W serves for additional marking-out of "good" cells on separate measurements. If the value of the i-th element of the neighborhood cell that has been labelled as "good" is considerably distant from the value of the i-th element of the central cell, the given "good" cell does not participate in formation of the new value of the i-th element of the central cell.

The local transition function of the second type is defined as

$$\sigma : \forall i = \overline{1,m}\ x_i \mapsto \begin{cases} \hat{x}_i^{\langle i \rangle}, \text{ if } \Phi\left(\hat{\mathbf{x}}^{\langle i \rangle}\right) < \Phi(\mathbf{x}); \\ x_i, \text{ otherwise,} \end{cases} \tag{3}$$

where $\forall\ i \neq j\ \hat{x}_i^{\langle j \rangle} = x_i$ and $\hat{x}_j^{\langle j \rangle} = x_j + \Delta(d, \alpha_1, \ldots, \alpha_k, j)$, $i = \overline{1,m}$, $j = \overline{1,m}$.

The increment function Δ is defined as

$$\Delta(d, \alpha_1, \ldots, \alpha_k, j) = \begin{cases} (-1)^r \sqrt{\frac{1}{k-1} \sum_{p=1}^{k} \left(x_j - \alpha_p^j\right)^2}, \text{ given } d = 0, \\ s, \text{ given } d = 1, \end{cases} \tag{4}$$

where r is a random natural number, and s is a random real number from the interval [0, 1].

As in the previous case, all elements of the vector contained in a cell are processed independently from each other. The modification of the i-th element is accepted only in the event that it has led to improvement of the solution contained in the cell. For calculation of a new value of the i-th element of the vector contained in the central neighborhood cell, the increment function Δ is used. If the regenerated cell is "bad", the value Δ for the *i*-th measurement is calculated by the modified formula of the root-mean-square error where the *i*-th element of the central cell is taken instead of an average value. It allows us to stay away from the current "bad" solution contained in the cell. If the updated cell is "good", a small random value is taken as Δ. It allows us to investigate the neighboring area in the search space.

2.4. The Suggested Algorithm of Continuous Optimization on the Basis of the Cellular Automaton with an Objective Function

The general approach to finding the optimum of an objective function by means of the cellular-automata model is the following: in accordance with the given model, the parameters of the objective cellular automaton, and the number of development steps are set; the crosshatch of the cellular automaton is filled with random values from the search space; the development of the cellular automaton is launched within the set number of steps; the condition of one of the cells from the total configuration of the crosshatch, which is the best solution, is taken to be the required optimum point.

This paper studied the dynamics of the cellular automaton with an objective function, which was based on the principles of the development set by the expressions (2) and (3), in order to find out its ability to converge to the optimum of the set objective function. The standard test functions of many parameters were chosen as objective functions. Their list is given in Table 1. All the experiments were run using two-dimensional cellular automaton.

Table 1. The test functions.

Function	Dimension	Search Space	Optimum
$\Phi_1 = \sum_{i=1}^{m} \left(-x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)\right)$	30	$[-500, 500]^m$	–12,569.49
$\Phi_1^* = \sum_{i=1}^{m} \left(-x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)\right) + 418.9829m$	30	$[-500, 500]^m$	0
$\Phi_2 = \sum_{i=1}^{m} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	30	$[-5.12, 5.12]^m$	0
$\Phi_3 = -20\exp\left(-\frac{1}{5}\sqrt{\frac{1}{m}\sum_{i=1}^{m} x_i^2}\right) - -\exp\left(\frac{1}{m}\sum_{i=1}^{m}\cos(2\pi x_i)\right) + 20 + \exp(1)$	30	$[-32, 32]^m$	0
$\Phi_4 = \frac{1}{4000}\sum_{i=1}^{m} x_i^2 - \prod_{i=1}^{m}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600, 600]^m$	0
$\Phi_5 = -\sum_{i=1}^{m} \sin(x_i)\sin^{20}\left(\frac{ix_i^2}{\pi}\right)$	100	$[0, \pi]^m$	–99.2784
$\Phi_6 = \frac{1}{N}\sum_{i=1}^{m} \left(x_i^4 - 16x_i^2 + 5x_i\right)$	100	$[-5, 5]^m$	–78.33236
$\Phi_7 = \sum_{i=1}^{m-1} \left[100(x_i - x_{i+1})^2 + (x_i - 1)^2\right]$	30	$[-30, 30]^m$	0
$\Phi_8 = \sum_{i=1}^{m} x_i^2$	30	$[-500, 500]^m$	0
$\Phi_9 = \sum_{i=1}^{m} \lvert x_i \rvert + \prod_{i=1}^{m} \lvert x_i \rvert$	30	$[-10, 10]^m$	0
$\Phi_{10} = \sum_{i=1}^{m-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	10	$[-2.048, 2.048]^m$	0
$\Phi_{11} = \sum_{i=1}^{m} \left(\lfloor x_i + 0.5 \rfloor\right)^2$	30	$[-100, 100]^m$	0

Table 1. *Cont.*

Function	Dimension	Search Space	Optimum
$\Phi_{12} = -\cos\left(2\pi\sqrt{\sum_{i=1}^{m} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{m} x_i^2} + 1$	30	$[-100, 100]^m$	0
$\Phi_{13} = \sum_{j=1}^{m}\sum_{i=1}^{m}\left(\frac{y_{i,j}^2}{4000} - \cos\left(y_{i,j}\right) + 1\right)$, $y_{i,j} = 100\left(x_j - x_i^2\right)^2 + (1 - x_i)^2$	30	$[-100, 100]^m$	0
$\Phi_{14} = \frac{\pi}{m}\left(\sum_{i=1}^{m-1}(y_i - 1)^2\left[1 + 10\sin^2(\pi y_{i+1})\right] + 10\sin^2(\pi y_1) + (y_m - 1)^2\right) + \sum_{i=1}^{m} u(x_i, 10, 100, 4)$, $y_i = 1 + \frac{1}{4}(x_i + 1)$, $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & \text{if } x_i > a, \\ 0, & \text{if } -a \le x_i \le a, \\ k(-x_i - a)^m, & \text{if } x_i < -a \end{cases}$	30	$[-50, 50]^m$	0
$\Phi_{15} = \frac{1}{10}\left(\sum_{i=1}^{m-1}(x_i - 1)^2\left[1 + \sin^2(3\pi x_{i+1})\right] + \sin^2(3\pi x_1) + (x_m - 1)^2\left[1 + \sin^2(2\pi x_m)\right]\right) + \sum_{i=1}^{m} u(x_i, 5, 100, 4)$	30	$[-50, 50]^m$	0

As a result, it was discovered that, when choosing the rule of the development on the basis of increment (3), the dynamics of the cellular automaton in the majority of cases has better convergence in comparison with the rule on the basis of averaging (2). However, the usage of these rules individually allowed us to find the optimum values only of some testing objective functions from the plurality of the examined ones. Using the rule (2) we were able to get the optimum of the function Φ_9 with the error not more than 0.03, and using the rule (3) it was possible to get the optima of the functions Φ_3, Φ_4, and Φ_8 with the error not more than 0.05.

In Figure 4a, the graph presenting the process of the optimization of the Φ_2 is given. Each graph presented in the given article has been received by averaging the results of 30 experiments.

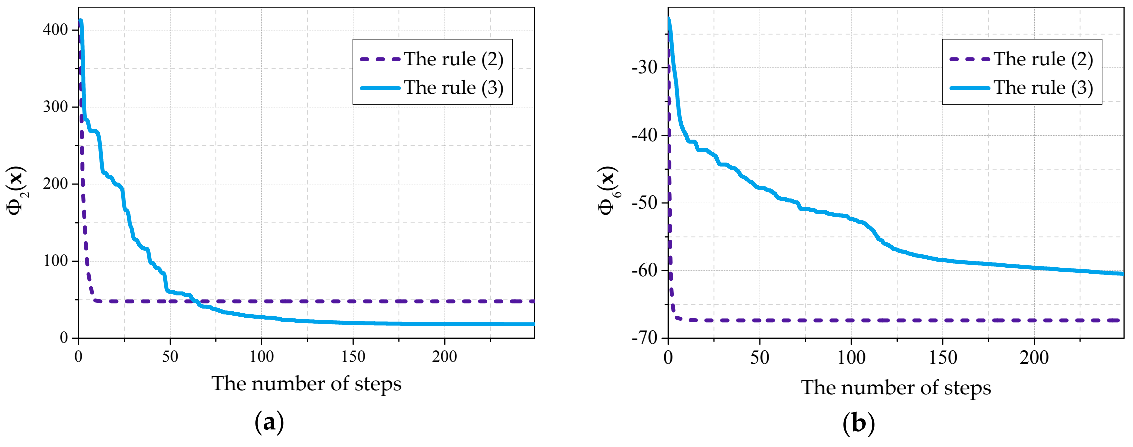


Figure 4. The results of optimization under independent usage of the rules of the cellular automaton (2) and (3): (**a**) for the objective function Φ_2; (**b**) for the objective function Φ_6.

It can be seen that, when using the rule (2) during the first 10 steps, fast movement to the optimum value side is observed, but after that the development stops. When using the rule (3) noticeable movement to optimum value side is observed in duration of the first 100–200 steps. However, the rule (2) differs by a prominently faster speed of convergence during the first effective steps of the cellular automaton development.

A similar picture was observed for the majority of objective functions. Nevertheless, in some cases a cellular automaton was able to advance to the optimum side noticeably farther using the rule (2), rather than using the rule (3), as it is shown in Figure 4b using Φ_6 as an example.

Due to this, a continuous optimization algorithm, being a general result of this paper, was based on the composition of suggested rules of evolvement of a cellular automaton with an objective function. This algorithm, formulated for a two-dimensional cellular automaton, is represented below.

```
Input:

a vector denoting sizes of a cellular automaton lattice, $\mathbf{L} = (l_1, l_2)$; a neighborhood pattern $\mathbf{Y}$; cell marking-out rule parameter $\mu$; a ratio of an acceptable remoteness of neighborhood cell states from corresponding central cell elements $\omega$; an objective function of $m$ variables $\Phi = \Phi(x_1, \ldots, x_m)$, the optimum(minimum) of which has to be found; search space borders $[a, b]^m \subset \Re^m$; a number of cellular automaton evolvement steps $T$.

Output:

a vector $\mathbf{x} \in [a, b]^m$, defining a point of the optimum being sought for.
Step 1. Fill the states of all cellular automaton lattice cells with random values from $[a, b]^m$.
Step 2. Calculate an objective function value for each lattice cell, storing these values in a $\mathbf{M} = (m_{ij})_{i=1,j=1}^{l_1,l_2}$ matrix.
Step 3. Find a minimal element in the $\mathbf{M}$ matrix. Write this element value to $\Phi_{\min}$ and its coordinates to $(p, q)$.
Step 4. Find the maximum element in the $\mathbf{M}$ matrix. Write this element value to $\Phi_{\max}$.
Step 5. Calculate the label for each cell according to the rule (1).
Step 6. For $t$ from 1 to $T$ do:
    Step 6.1. For $i$ from 1 to 3 do:
        Step 6.1.1. Update every lattice cell state according to the evolvement rule (2) and neighborhood pattern $\mathbf{Y}$.
        Step 6.1.2. Update the $\mathbf{M}$ matrix, $\Phi_{\min}$, $\Phi_{\max}$, and $(p, q)$.
        Step 6.1.3. Update every cell label according to the rule (1).
    Step 6.2. Update every cell state according to the evolvement rule (3) and neighborhood pattern $\mathbf{Y}$.
    Step 6.3. Update the $\mathbf{M}$ matrix, $\Phi_{\min}$, $\Phi_{\max}$, and $(p, q)$.
    Step 6.4. Update every cell label according to the rule (1).
Step 7. Return state of the cell with coordinates $(p, q)$ as the $\mathbf{x}$ vector and finish the algorithm.
```

3. Experiments

Figure 5 represents convergence plots built for the Φ_3 function with different optimization algorithm parameters. Using other objective functions gave similar plots. The best results are obtained for the cellular automaton with 10×10 cell lattice and Moore neighborhood with single-unit radius (3×3 cells). Further increase of the lattice and Moore neighborhood sizes leads to convergence deterioration.

Notice that not only the number of cells in a lattice of the cellular automaton influences the quality of optimization, but also the lattice form. The accuracy of the received solutions increases at the approach of the lattice form to a square as it can be seen in Figure 5c. Besides, a one-dimensional cellular automaton with the lattice of 1×100 cells shows the worst result.

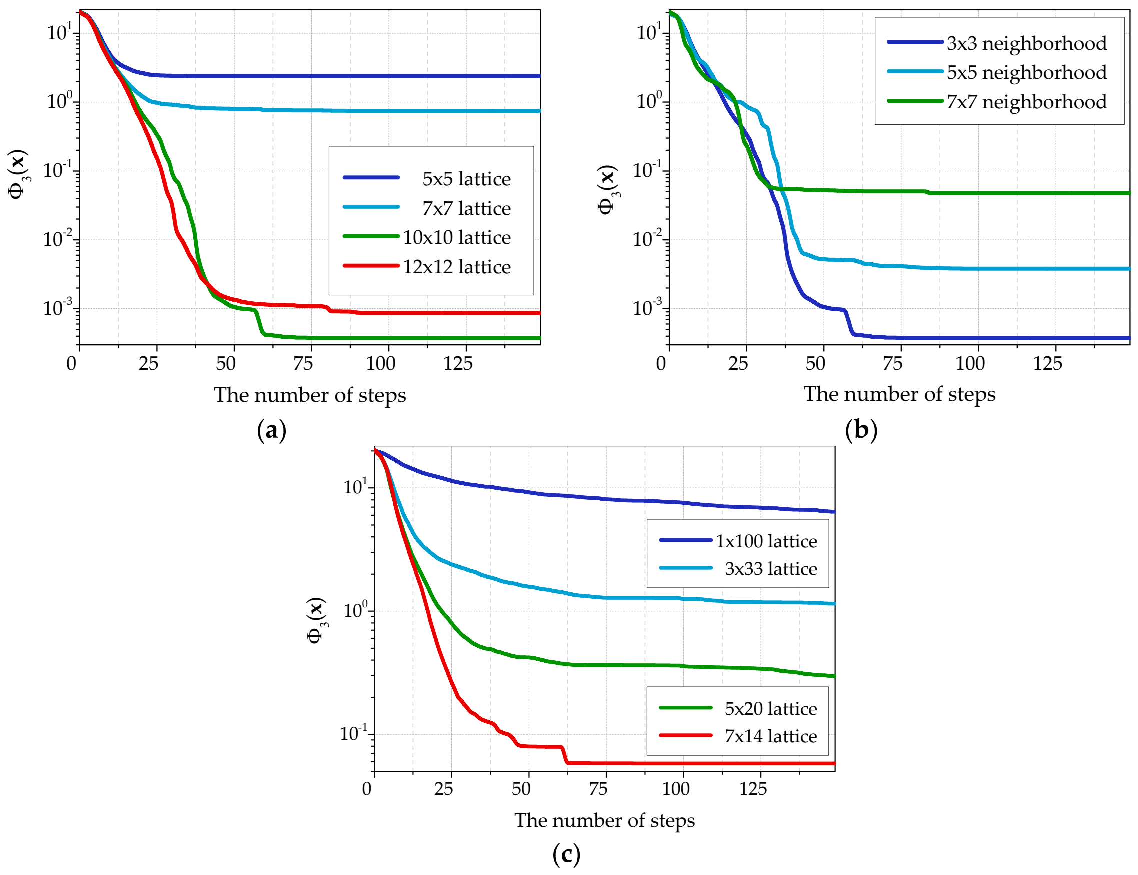
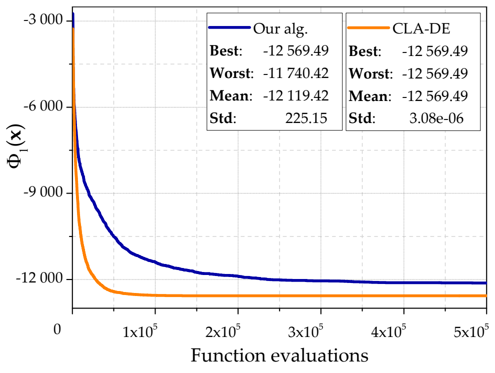


Figure 5. The results of optimization of the objective function Φ_3 under different values of the parameters of the cellular automaton: (**a**) under change of the sizes of the lattice and fixed Moore neighborhood 3×3 of the cells; (**b**) under change of the sizes of the neighborhood for lattice 10×10 of the cells; (**c**) under change of the lattice form and a comparable number of cells.

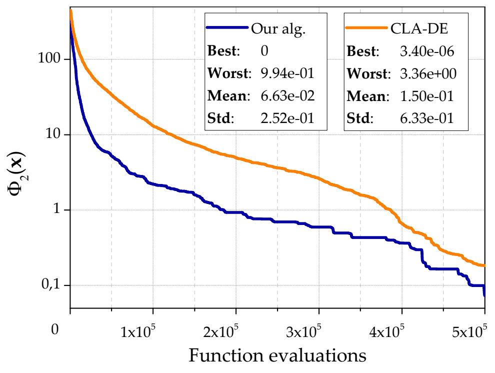
The main analog for the algorithm being described is the CLA-DE algorithm represented in [16], because among all the algorithms being considered it keeps the properties of cellular automaton model to the fullest extent. That algorithm is based on the dynamics of a cellular automaton-designed system, every cell of which contains a set of learned automata responsible for the changes of separate variables of an objective function in the search space.

Figure 6 denotes the results of the corresponding experiments for 15 test objective functions. The following algorithm parameter values were used: $\mathbf{L} = (10, 10)$, $\Upsilon = ((-1,-1), (-1,0), (-1,1), (0,-1), (0,0), (0,1), (1,-1), (1,0), (1,1))$, $\mu = 0.4$, $\omega \in [0.01, 0.025]$, $T < 500$.

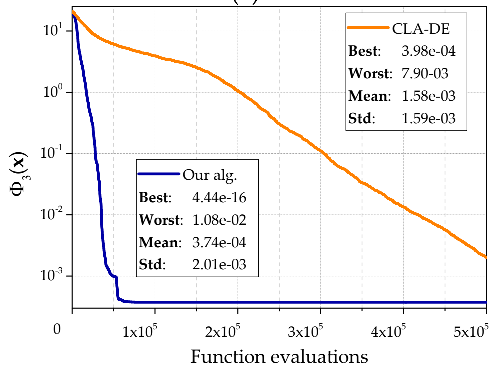
Since an amount of calculation for the objective function on first step of the learning cellular automaton and an amount of calculation for the objective function on the first step of the cellular automaton are different, the average dependence of the best found value and the amount of calculation of the objective function for 30 runs is shown. During each run, 500,000 function evaluations were performed. Each graph shows the best result, the worst result and the standard deviation.



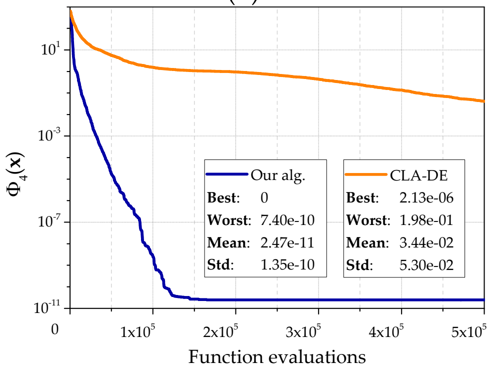
(**a**)



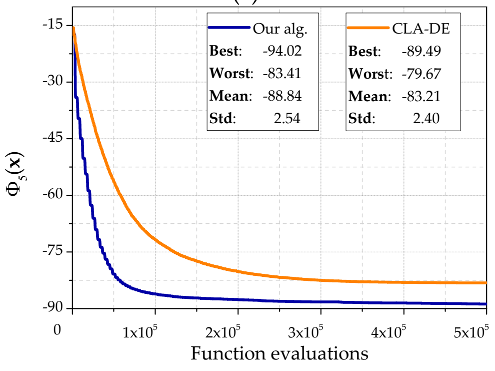
(**b**)



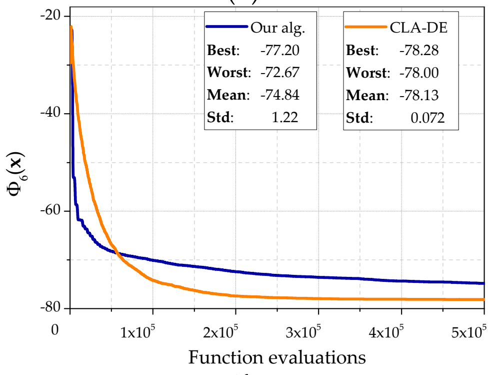
(**c**)



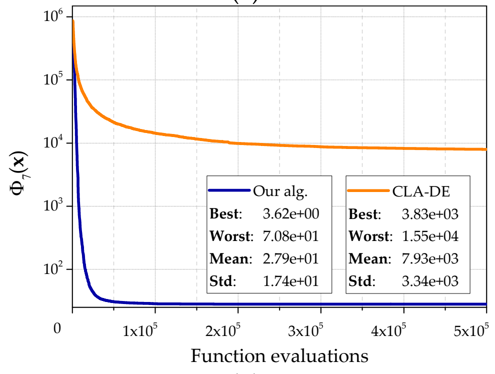
(**d**)



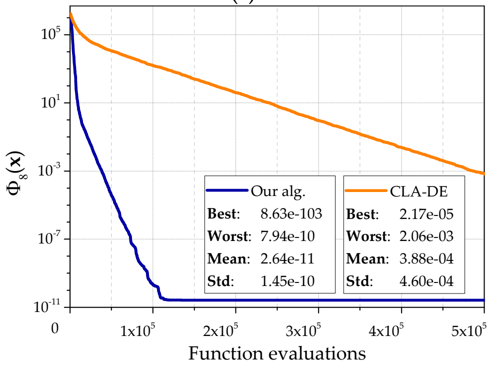
(**e**)



(**f**)

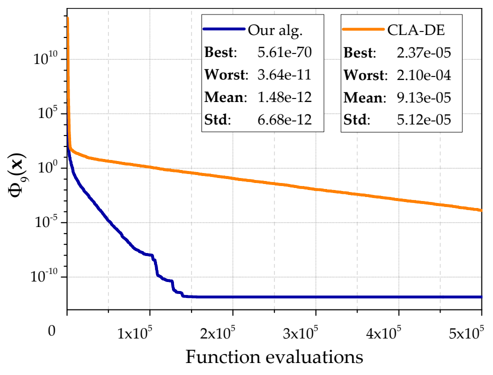


(**g**)

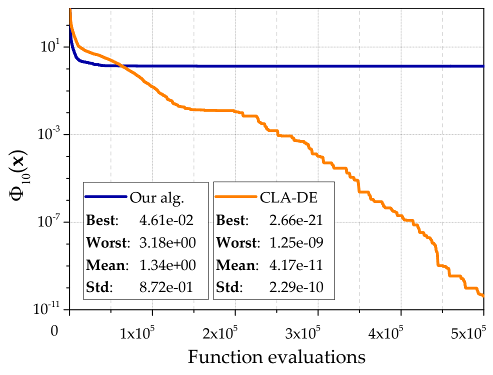


(**h**)

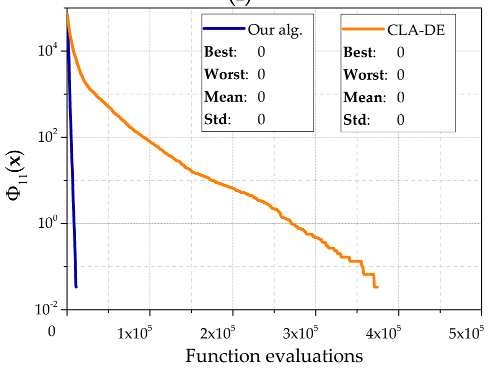
Figure 6. *Cont.*



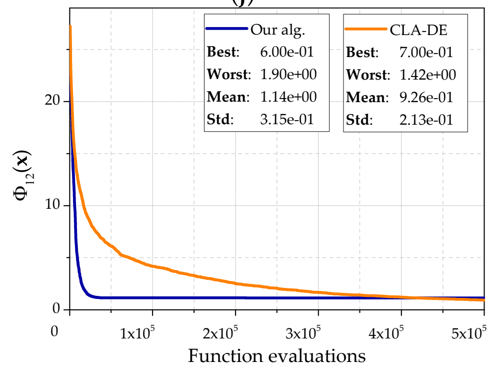
(**i**)



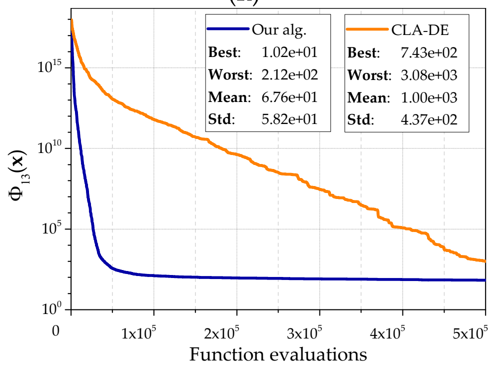
(**j**)



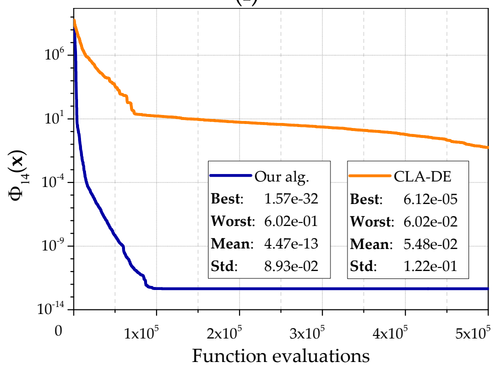
(**k**)



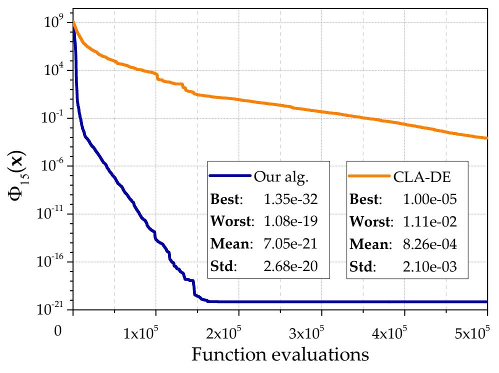
(**l**)



(**m**)



(**n**)



(**o**)

Figure 6. The comparison of the developed algorithm and algorithm CLA-DE: (**a**) for the objective function Φ_1; (**b**) for the objective functions Φ_2; (**c**) for the objective functions Φ_3; (**d**) for the objective functions Φ_4; (**e**) for the objective functions Φ_5; (**f**) for the objective functions Φ_6; (**g**) for the objective functions Φ_7; (**h**) for the objective functions Φ_8; (**i**) for the objective functions Φ_9; (**j**) for the objective functions Φ_{10}; (**k**) for the objective functions Φ_{11}; (**l**) for the objective functions Φ_{12}; (**m**) for the objective functions Φ_{13}; (**n**) for the objective functions Φ_{14}; (**o**) for the objective functions Φ_{15}.

The character of these graphs allows us to make the following conclusions. Our algorithm has a higher speed of convergence than the CLA-DE. It quickly reaches a sufficiently accurate approximation to the optimum, which is shown by a bigger angle of curves. Generally, the proposed algorithm is better than CLA-DE on both the average and the best values. CLA-DE is better only in some cases, for example, on the objective function Φ_1. This can be explained by getting into local optimum, which is abundantly presented in the function Φ_1. A similar situation is observed for the function Φ_6. It should be noted that the CLA-DE gets into local optimums less frequently. The lack of horizontal sections on the curves that sometimes appear in our algorithm also shows it. Experiments showed that in some cases CLA-DE is able to continue to move toward the optimum longer than our algorithm. This property is clearly seen, for example, on functions Φ_3, Φ_8, Φ_9. However, this increase of accuracy can be achieved only by increasing the number of calculations.

This observation is also confirmed by experiments with the objective functions Φ_7 and Φ_{10}, which are the Rosenbrock functions with dimension 30 and 10, respectively. For the function Φ_7 our algorithm returns a value close to the optimum. CLA-DE significantly does not achieve the optimum and shows the worst result. However, in most of the experiments for the function Φ_{10} our algorithm also quickly comes to a value close to the optimum, and stops the evolution, while CLA-DE continues to move toward the optimum. Our algorithm still shows a higher speed of convergence on the first iterations.

On the example of objective function Φ_{12}, CLA-DE is also ahead of our algorithm, but only by average result by virtue of smaller spread in the values received in separate experiments. CLA-DE is behind our algorithm compared to the best result which has been received for the given function. Besides, lower speed of CLA-DE convergence leads to the fact that the advantage of the average value is shown only after 400,000 function evaluations.

For an estimation of significance of the received results we will use the following nonparametric statistical tests intended for pairwise comparison algorithms: Sign test and Wilcoxon test [19].

As a value characterising productivity of algorithms we will use the error obtained for each one over the 15 benchmark functions considered. Table 2 shows the best and average values of this value which were received during a various number of function evaluations: 50,000 and 500,000. Statistical comparison of the results obtained by means of two considered algorithms for a small number of function evaluations will serve for refinement of the above mentioned observations.

Table 2. The values of an error obtained for test functions.

Function	**Our Algorithm**				**CLA-DE**			
	50,000 Function Evaluation		**500,000 Function Evaluation**		**50,000 Function Evaluation**		**500,000 Function Evaluation**	
	Mean	**Best**	**Mean**	**Best**	**Mean**	**Best**	**Mean**	**Best**
Φ_1	2.07×10^3	1.36×10^3	4.50×10^2	3.38×10^{-3}	1.49×10^2	7.96×10^1	3.39×10^{-3}	3.38×10^{-3}
Φ_2	5.50	9.96×10^{-1}	6.63×10^{-2}	0	3.39×10^1	2.52×10^1	1.50×10^{-1}	3.40×10^{-6}
Φ_3	9.94×10^{-4}	2.15×10^{-5}	3.74×10^{-4}	4.44×10^{-16}	6.09	4.68	1.58×10^{-3}	3.98×10^{-4}
Φ_4	1.88×10^{-5}	6.82×10^{-7}	2.47×10^{-11}	0	5.56	2.92	3.44×10^{-2}	2.13×10^{-6}
Φ_5	1.84×10^1	1.48×10^1	1.04×10^1	5.25	4.31×10^1	3.89×10^1	1.61×10^1	9.79
Φ_6	1.01×10^1	8.55	3.49	1.14	1.17×10^1	1.05×10^1	2.03×10^{-1}	4.92×10^{-2}
Φ_7	3.06×10^1	3.81	2.79×10^1	3.62	2.13×10^3	1.03×10^3	7.93×10^3	3.83×10^3
Φ_8	4.19×10^{-5}	3.15×10^{-7}	2.64×10^{-11}	8.63×10^{-103}	1.15×10^4	5.24×10^3	3.88×10^{-4}	2.17×10^{-5}
Φ_9	1.57×10^{-5}	3.88×10^{-7}	1.48×10^{-12}	5.61×10^{-70}	4.45	3.18	9.13×10^{-5}	2.37×10^{-5}
Φ_{10}	1.40	5.00×10^{-2}	1.34	4.61×10^{-2}	2.51	1.09	4.17×10^{-11}	2.66×10^{-21}
Φ_{11}	0	0	0	0	5.11×10^2	2.15×10^2	0	0
Φ_{12}	1.15	6.00×10^{-1}	1.14	6.00×10^{-1}	6.11	3.90	9.26×10^{-1}	7.00×10^{-1}
Φ_{13}	3.74×10^2	2.02×10^2	6.76×10^1	1.02×10^1	1.20×10^{13}	2.15×10^{11}	1.00×10^3	7.43×10^2
Φ_{14}	6.40×10^{-9}	9.40×10^{-10}	5.59×10^{-13}	1.57×10^{-32}	4.67×10^3	9.10	6.53×10^{-2}	6.12×10^{-5}
Φ_{15}	7.03×10^{-8}	1.56×10^{-9}	7.05×10^{-21}	1.35×10^{-32}	9.55×10^4	3.28×10^2	8.26×10^{-4}	1.00×10^{-5}

The results of the application of Sign test and Wilcoxon test to the given data sets are shown in Tables 3 and 4.

Table 3 shows the number of wins and losses of our algorithm in four considered cases, and the level of significance for each case is defined. Identical results are divided equally between both algorithms.

Table 3. Results of the Sign test for comparison of our algorithm and CLA-DE algorithm.

Results	**50,000 Function Evaluation**		**500,000 Function Evaluation**	
	Mean	**Best**	**Mean**	**Best**
Wins (+)	14	14	11	13
Losses (–)	1	1	4	2
Detected differences	$\alpha = 0.05$	$\alpha = 0.05$	0.1	$\alpha = 0.05$

We can see that, for 50,000 function evaluations, our algorithm shows significant improvement over CLA-DE with a level of significance $\alpha = 0.05$ both in the best and in average results. For 500,000 function evaluations our algorithm shows significant improvement over CLA-DE with the level of significance $\alpha = 0.1$ in the best result and $\alpha = 0.05$ in the average result.

Sign test is an elementary statistical test allowing one to evaluate the advantage of one algorithm over another with a series of experiments. However, it considers only the part of experiments in which the given algorithm showed the best result. Wilcoxon test is a stronger test as it also considers the difference in productivity for separate experiments.

Table 4. Results of the Wilcoxon test for comparison of our algorithm and CLA-DE algorithm.

Results	**50,000 Function Evaluation**		**500,000 Function Evaluation**	
	Mean	**Best**	**Mean**	**Best**
***T*-value**	10	12	39	20
***p*-value**	0.004514	0.006407	0.396727	0.041328

We can see that, in three out of four cases, the Wilcoxon test confirms the considerable advantage of our algorithm over the CLA-DE algorithm. Our algorithm leaves CLA-DE considerably behind at a small number of function evaluations, both by average result, and by the best result. At a higher number of function evaluations our algorithm also advances CLA-DE by the best result, but for an average result it is not possible to disprove the null hypothesis.

It is connected to a wider dispersion of output values which is characteristic for our algorithm. At the same time, its ability to quickly obtain the values close to optimum, advancing the algorithm-competitor, is proved by the Wilcoxon test for a small number of function evaluations.

Thus, the conclusions made earlier at the analysis of the graphs in Figure 6 are proved by statistical tests.

Earlier during the literature review other examples of use of cellular automata in problems of continuous optimization except CLA-DE were also considered. However, when analyzing corresponding algorithms, we determined that cellular automata are applied in these algorithm mainly indirectly. The given algorithms represent modifications of the known metaheuristics in which the cellular-automatic principle of local interactions is involved, but evolution of the cellular automaton is not defined. The CLA-DE algorithm considered by us as the main analog is possible to define as an exception. However, a cellular automaton is not exactly used but a set of trained automata united in a lattice. Also hybridization with differential evolution takes place.

Having compared the offered algorithm with CLA-DE, we showed that cellular automata can be used for the solution of problems of optimization not only as an auxiliary device but also as the basic computing model. Now we will compare our algorithm with some other metaheuristics that were considered earlier during the literature review: the artificial fish swarm algorithm based on cellular learning automata (AFSA-CLA) [15] and the evolutionary membrane algorithm (EMA) [13].

The results of additional comparison are given in Table 5. The table shows the best result of each algorithm. We have carried out the given comparison with metaheuristics of those authors who presented the results of computing experiments with standard test functions from ones given in Table 1. To provide equal conditions with the experiments described in the papers [13,15], the values given in the table were obtained after 300,000 function evaluations. A dash in a cell means that, for the given function, experimental data is absent.

Table 5. Experimental results of AFSA-CLA, EMA, and our algorithm.

Function	AFSA-CLA	EMA	Our Algorithm
Φ_1^*	—	4.56×10^{2}	1.18×10^{2}
Φ_2	0	2.06×10^{1}	0
Φ_3	1.33×10^{-14}	6.42×10^{-1}	3.99×10^{-15}
Φ_4	4.44×10^{-16}	1.73×10^{-2}	0
Φ_5	—	—	-9.33×10^{1}
Φ_6	—	—	-7.52×10^{1}
Φ_7	—	—	3.62
Φ_8	3.19×10^{-104}	7.11×10^{-103}	2.32×10^{-82}
Φ_9	—	—	1.46×10^{-36}
Φ_{10}	3.30×10^{-3}	8.34×10^{-1}	5.58×10^{-2}
Φ_{11}	0	—	0
Φ_{12}	—	8.12×10^{-2}	6.00×10^{-1}
Φ_{13}	—	9.26×10^{1}	1.02×10^{1}
Φ_{14}	—	3.73×10^{-2}	1.57×10^{-32}
Φ_{15}	—	2.19×10^{-3}	1.35×10^{-32}

Our algorithm shows the best result on the functions Φ_1^*, Φ_3, Φ_4, Φ_{12}–Φ_{15}. AFSA-CLA shows the best result on the functions Φ_8 and Φ_{10}. On the function Φ_2 and Φ_{11} our algorithm and AFSA-CLA return the real optimum. EMA is better than our algorithm on the functions Φ_8, but worse than AFSA-CLA.

The algorithm AFSA-CLA has a high performance because it is based on metaheuristics, which is effective in itself. Also, a learning cellular automaton improves these metaheuristics. This leads to higher performance than when a learning cellular automaton is used separately for optimization like it occurs in CLA-DE. AFSA-CLA is significantly ahead of our algorithm only on the function Φ_{10}, in all other cases both algorithms show similar results. At the same time, our algorithm can achieve real optimum in two cases, but AFSA-CLA only in one case.

The algorithm EMA shows the worst results in terms of accuracy. However, according to [13], this algorithm has a high speed of convergence and allows one to obtain acceptable values for a relatively small number of function evaluations. In this case, for the function Φ_8 an error is negligible for all three algorithms, which allows us to consider that the results are equivalent.

The results of paired comparison of our algorithm with AFSA-CLA and EMA algorithms by means of statistical tests are shown in Table 6. Initial data were calculated on the basis of the values given in Tables 1 and 5.

Table 6. Results of statistical tests for AFSA-CLA, EMA, and our algorithm.

Results	Our Algorithm versus AFSA-CLA	Our Algorithm versus EMA
	Sign Test	
Wins (+)	3	7
Loses (–)	3	2
Detected differences	—	$\alpha = 0.1$
	Wilcoxon Test	
***T*-value**	5	6
***p*-value**	1.0	0.050613

When comparing with the AFSA-CLA algorithm, our algorithm is the winner in two cases out of six and in two cases the results coincide. This does not allow to disprove a null hypothesis and to establish the significant advantage of one algorithm over another. However, mainly it is connected to insufficient amount of experimental data for AFSA-CLA.

In turn, comparison with EMA shows that our algorithm considerably surpasses EMA with the significance level of $\alpha < 0.1$.

Therefore, the algorithm suggested by us on the basis of the cellular automaton with an objective function shows more acceptable results in comparison with others metaheuristics.

4. Conclusions

The main objective of our research consisted in showing applicability of cellular automata for the solution of the problem of continuous optimization. The given mathematical apparatus has been applied in optimization problems for a long time. However, the analysis of known algorithms showed that cellular automata are mainly used as an auxiliary computing model. We want to show the possibility of direct application of cellular automata for optimization of arbitrary functions. For this purpose, we introduce the concept of the cellular automaton with an objective function, which is a modification of the classical cellular automaton. The introduced cellular-automata model has the following features:

- The objective function Φ is included in the model.
- Each cell of the array contains the vector real value $\mathbf{x} \in \Re^m$ which is considered to be the element of function domain of the objective function Φ.
- The development of the cellular automaton is organized in such a way that, while renovating the state of each cell of the array, its new state will be closer to the optimum.

In this paper, two rules of cellular automaton with an objective function development are introduced, and properties of their convergence are analyzed.

On the basis of the described model with the usage of the two introduced rules, we formulated a new algorithm of continuous optimization, which belongs to the stochastic algorithm in connection to the usage of random values in development of the cellular automaton.

The conducted benchmarks showed the similarity of the effectiveness between the suggested algorithm and the analogs. The usage of the cellular automaton with the array 10×10 cells with Moore neighborhood 3×3, having the property of geometrical symmetry, has led to the top results.

The advantage of the developed algorithm is the convergence of fast speed during the first steps of the computation. This advantage lets us recommend the algorithm for use in real time data problems, where the speed of getting a solution closer to the optimum is more important than accuracy.

The goal of future research will be the development of the suggested approach, the research of new rules of the cellular automaton with an objective function development that are different from these being considered in this paper, and also the synthesis of new algorithms of continuous optimization on the basis of composition of the given rules.

Acknowledgments: The work was supported by the Ministry of Education and Science of the Russian Federation in terms of the basic part of the state order for Tomsk State University of Control Systems and Radioelectronics in 2015–2016 (project No. 3657). We are very grateful to the editor and the anonymous referees for their constructive comments and helpful suggestions to improve the quality of this paper.

Author Contributions: The authors contributed equally to the work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kennedy, J.; Ebenhart, R. Particle Swarm Optimization. In Proceedings of the 1995 IEEE International Conference on Neural Networks, University of Western Australia, Perth, Australia, 27 November–1 December 1995; pp. 1942–1948.
2. Storn, R.; Price, K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359.
3. Karaboga, D.; Akay, B. A survey: Algorithms simulating bee swarm intelligence. *Artif. Intell. Rev.* **2009**, *31*, 61–85.
4. Khodashinskii, I.A.; Dudin, P.A. Identification of fuzzy systems using a continuous ant colony algorithm. *Optoelectron. Instrum. Data Process.* **2012**, *48*, 54–61.
5. Canyurt, O.E.; Hajela, P. Cellular genetic algorithm technique for the multicriterion design optimization. *Struct. Multidiscip. Optim.* **2010**, *40*, 201–214.
6. Sidiropoulos, E.; Fotakis, D. Cell-based genetic algorithm and simulated annealing for spatial groundwater allocation. *WSEAS Trans. Environ. Dev.* **2009**, *5*, 351–360.
7. Sidiropoulos, E. Harmony Search and Cellular Automata in Spatial Optimization. *Appl. Math.* **2012**, *3*, 1532–1537.
8. Du, T.S.; Fei, P.S.; Shen, Y.J. A new cellular automata-based mixed cellular ant algorithm for solving continuous system optimization programs. In Proceedings of the 4th International Conference on Natural Computation, Jinan, China, 18–20 October 2008; pp. 407–411.
9. Gholizadeh, S. Layout optimization of truss structures by hybridizing cellular automata and particle swarm optimization. *Comput. Struct.* **2013**, *125*, 86–99.
10. Tovar, A.; Patel, N.M.; Niebur, G.L.; Sen, M.; Renaud, J.E. Topology optimization using a hybrid cellular automation method with local control rules. *J. Mech. Des.* **2006**, *128*, 1205–1216.
11. Penninger, C.L.; Watson, L.T.; Tovar, A.; Renaud, J.E. Convergence analysis of hybrid cellular automata for topology optimization. *Struct. Multidiscip. Optim.* **2010**, *40*, 201–214.
12. Bochenek, B.; Tajs-Zielinska, K. Novel local rules of cellular automata applied to topology and size optimization. *Eng. Optim.* **2012**, *44*, 23–35.
13. Han, M.; Liua, C.; Xing, J. An evolutionary membrane algorithm for global numerical optimization problems. *Inf. Sci.* **2014**, *276*, 219–241.
14. Toffoli, T.; Margolus, M. *Cellular Automata Machines*; MIT Press: Cambridge, MA, USA, 1987.
15. Yazdani, D.; Golyari, S.; Meybodi, M.R. A New Hybrid Algorithm for Optimization Based on Artificial Fish Swarm Algorithm and Cellular Learning Automata. In Proceedings of the 5th International Symposium on Telecommunications, Sharif University of Technology, Tehran, Iran, 4–6 December 2010; pp. 932–937.
16. Vafashoar, R.; Meybodi, M.R.; Momeni Azandaryani, A.H. CLA-DE: A hybrid model based on cellular learning automata for numerical optimization. *Appl. Intell.* **2012**, *36*, 735–748.
17. Bandman, O.L. Discrete Models of Physicochemical Processes and Their Parallel Implementation. In Proceedings of the Second Russia-Taiwan Symposium on Methods and Tools of Parallel Programming Multicomputers, Vladivostok, Russia, 16–19 May 2010.
18. Tanaka, I. Effects of Initial Symmetry on the Global Symmetry of One-Dimensional Legal Cellular Automata. *Symmetry* **2015**, *7*, 1768–1779.
19. Derrac, J.; Garcia, S.; Molina, D.; Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **2011**, *1*, 3–18.