DOI: 10.17323/2587-814X.2023.2.7.19

# Modeling and optimization of strategies for making individual decisions in multi-agent socio-economic systems with the use of machine learning

**Andranik S. Akopov**
E-mail: akopovas@umail.ru

Central Economics and Mathematics Institute, Russian Academy of Sciences
Address: 47, Nakhimovsky Prospect, Moscow 117418, Russia

**Abstract**

This article presents a new approach to modeling and optimizing individual decision-making strategies in multi-agent socio-economic systems (MSES). This approach is based on the synthesis of agent-based modeling methods, machine learning and genetic optimization algorithms. A procedure for the synthesis and training of artificial neural networks (ANNs) that simulate the functionality of MSES and provide an approximation of the values of its objective characteristics has been developed. The feature of the two-step procedure is the combined use of particle swarm optimization methods (to determine the optimal values of hyperparameters) and the Adam machine learning algorithm (to compute weight coefficients of the ANN). The use of such ANN-based surrogate models in parallel multi-agent real-coded genetic algorithms (MA-RCGA) makes it possible to raise substantially the time-efficiency of the evolutionary search for optimal solutions. We have conducted numerical experiments that confirm a significant improvement in the performance of MA-RCGA, which periodically uses the ANN-based surrogate-model to approximate the values of the objective and fitness functions. A software framework has been designed that consists of the original (reference) agent-based model of trade interactions, the ANN-based

surrogate model and the MA-RCGA genetic algorithm. At the same time, the software libraries FLAME GPU, OpenNN (Open Neural Networks Library), etc., agent-based modeling and machine learning methods are used. The system we developed can be used by responsible managers.

**Keywords:** multi-agent socio-economic systems, particle swarm optimization, modeling random sales, machine learning, artificial neural networks, genetic optimization algorithms

**Citation:** Akopov A.S. (2023) Modeling and optimization of strategies for making individual decisions in multi-agent socio-economic systems with the use of machine learning. *Business Informatics*, vol. 17, no. 2, pp. 7–19.
DOI: 10.17323/2587-814X.2023.2.7.19

## Introduction

Currently, there is a growing interest in studying the behavior of multi-agent socio-economic systems (MSES) and developing decision support systems (DSS) using agent-based modeling (AOM), machine learning and heuristic (in particular, genetic) optimization algorithms.

Most modern DSS can be divided into two enlarged classes: effective control systems based on simulation, including optimization modeling, and expert decision support systems.

As examples of DSS belonging to the first type, we can highlight: a software package designed to manage the investment activities of a large oil company [1], a decision support system for environmental and economic planning [2], intelligent transport systems [3–5], etc.

The most well-known DSS of the second type include expert systems for making strategic decisions using the hierarchy analysis method [6, 7], systems designed to prioritize decisions in the management of IT projects [8], systems that support the ability to select the best alternatives in case of poorly structured initial data [9], etc.

In that work, the MSES management system of the first type, intended mainly for the formation of optimal strategies for making individual decisions in multiple trade interactions (concluding barter and monetary transactions), is proposed. A software implementation of a modified model of random sales [10] was performed with the use of the agent-based modeling methods [11, 12], machine learning [13, 14], genetic [1, 3, 15], and particle swarm optimization algorithms [16].

The urgency of development of such an intelligent system is mainly due to the high computational complexity of determining the optimal moments for concluding barter and monetary transactions in trading systems with random interactions of economic agents. In particular, in conditions when economic agents maximize the utility of future consumption due to the effective control of their own states, allowing or blocking paired trade interactions. The traditional approach to finding optimal strategies in such multi-agent systems is based on solving optimal control problems using the classical methods of variational calculus and dynamic programming [17]. However, due to the high dimensionality of such MSESs' models (i.e. a large number of interacting agents) the computational complexity of searching for individual solutions increases many times over. Therefore, it is necessary to develop a software package that uses machine learning methods and heuristic algorithms for an approximate solution of the optimal control problems for the strategy of trade interactions in the MSES.

The purpose of this work is to develop a new approach to modeling and forming strategies for making individual decisions in MSES using machine learning methods, particle swarm and genetic optimization

algorithms. The general methodology of this approach is to create a simulation model of MSES, perform experiments (such as the Monte Carlo type) with the model to form a training sample, synthesize an artificial neural network (ANN) with the optimal topology and integrate it into a genetic optimization algorithm for use as a surrogate model, significantly accelerating the evolutionary search for solutions within the entire ensemble of interacting economic agents. At the same time, the effectiveness of the approach we developed and the software we designed is investigated with the use of the single-objective optimization problem for the proposed agent-based model of trade interactions, implemented with the agent-based modeling system FLAME GPU [18] and the OpenNN machine learning library [19], as the case study.

## 1. Agent-based model of trade interactions

The essential feature of the proposed agent-based model of trade interactions that highlights it among the previously known ones is considering the initial spatial arrangement of sellers and buyers, which is set using various configurations, examples of which are shown in *Fig. 1*.

In the model, at each moment, between each arbitrary pair of agents mutually located within the boundaries of the trade interaction zone (*Fig. 1*), a barter or monetary transaction can be completed (i.e., the exchange of goods for goods, or the exchange of goods for its monetary equivalent), if these agents, firstly, are in the state of readiness for such transactions, and secondly, they have the necessary product, or a product that is close to the target product in terms of its consumer characteristics.

Here,

$T = \{t_0, t_1, ..., |T|\}$ is the set of time moments (by days), $|T|$ is the total number of moments;

$t_0 \in T$, $t_{|T|} \in T$ are the initial and final moments of the model;

$I = \{i_1, i_2, ..., i_{|I|}\}$ is the set of agent indices, where $|I|$ is the total number of agents, $\tilde{i} \in I$ are the sellers' indices, $\hat{i} \in I$ are the buyers' indices;

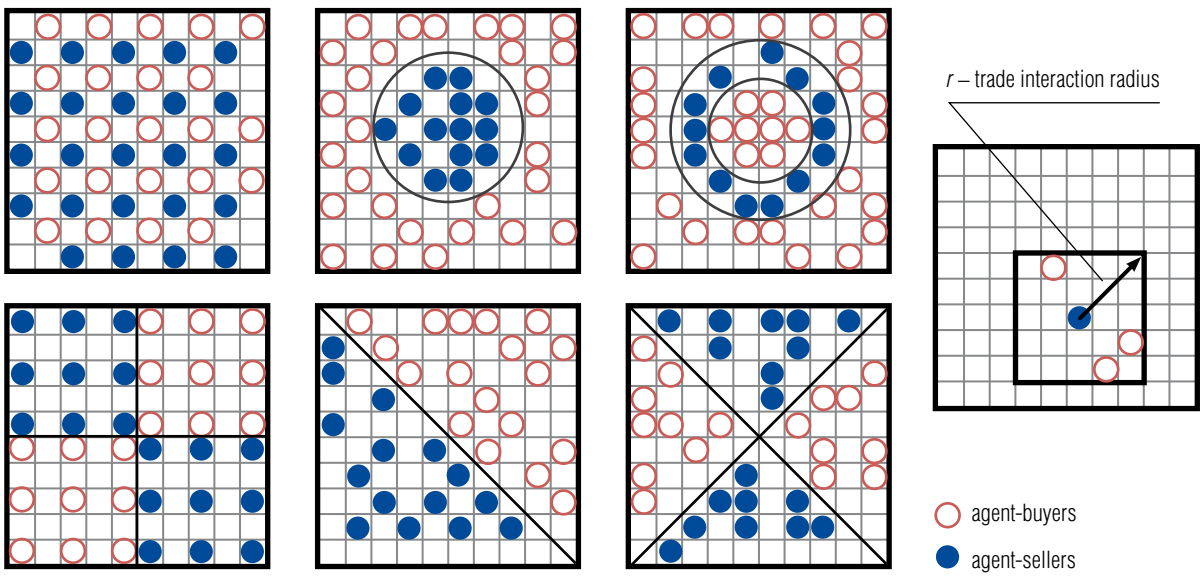


*Fig. 1.* Possible configurations of the initial distribution of agents in the MSES space.

$P = \{p_1, p_i, ..., p_{|P|}\}$ is the set of product indices, $|P|$ is the total number of products, $p_i(t_k) \in P$, $i \in I$, $t_k \in T$ is the product index belonging to the $i^{th}$-agent, $d_i(t_k) \in P$, $i \in I$, $t_k \in T$ is the product index that is needed for the $i^{th}$-agent;

$\{b_i(t_k), m_i(t_k)\} \in \{0, 1\}$, $i \in I$ is the state of readiness of the agent to conclude barter and monetary transactions, respectively, at moment $t_{k-1}$ ($t_{k-1} \in T$): 0 – transactions are prohibited, 1 – transactions are allowed.

Then, the distance between the product of $\tilde{i}^{th}$ agent-seller ($\tilde{i} \in I$) and the product of the $\hat{i}^{th}$ agent-buyer ($\hat{i} \in I$), measured along the arc's length of a numerical circle with evenly distributed numbers 1, 2, ..., $|P|$ at moment $t_{k-1}$ ($t_{k-1} \in T$):

$$\delta_{\tilde{i}\hat{i}}(t_k) = \\ = \frac{1}{|P-1|}\min\left\{\left|p_{\tilde{i}}(t_k) - d_{\hat{i}}(t_k)\right|, \left|P\right| - \left|p_{\tilde{i}}(t_k) - d_{\hat{i}}(t_k)\right|\right\}. \quad (1)$$

At the same time, the assessment of the level of compliance of the product of the agent-seller with the interests of the agent-buyer can be given as:

$$\gamma_{\tilde{i}\hat{i}}(t_k) = \begin{cases} 1, \text{ если } \delta_{\tilde{i}\hat{i}}(t_k) \le \varpi, \\ 0, \text{ если } \delta_{\tilde{i}\hat{i}}(t_k) > \varpi, \end{cases} \quad (2)$$

$\varpi \ge 0$ is the coefficient of threshold compliance of the product of the agent-seller with the interests of the agent-buyer (the coefficient of 'contractability').

At the same time, the readiness states of the $i^{th}$-agent ($i \in I$) to complete barter and monetary transactions can be formed for each moment $t_k$ ($t_k \in T$) using, in particular, the log-normal (*the first method*) or beta distributions (*the second method*) with given characteristics:

$$b_i(t_k) = \begin{cases} \left\lfloor \ln N(\mu_b, \sigma_b^2) \right\rfloor, \text{ if I is fulfilled,} \\ \left\lfloor \ln N(\mu_b, \sigma_b^2) \right\rfloor \\ 0, \text{ if II is fulfilled,} \\ \left\lfloor \text{Beta}(\alpha_b, \beta_b) \right\rfloor, \text{ if III is fulfilled,} \end{cases} \quad (3)$$

$$m_i(t_k) = \begin{cases} \left\lfloor \ln N(\mu_m, \sigma_m^2) \right\rfloor, \text{ if IV is fulfilled,} \\ \left\lfloor \ln N(\mu_m, \sigma_m^2) \right\rfloor \\ 0, \text{ if V is fulfilled,} \\ \left\lfloor \text{Beta}(\alpha_m, \beta_m) \right\rfloor, \text{ if VI is fulfilled,} \end{cases} \quad (4)$$

where

I. if the log-normal distribution is used to form the readiness states of agents for barter transactions and $\ln N(\mu_b, \sigma_b^2) > 0$,

II. if the log-normal distribution is used to form the readiness states of agents for barter transactions and $\ln N(\mu_b, \sigma_b^2) = 0$,

III. if the beta distribution is used to form the readiness states of agents for barter transactions,

IV. if the log-normal distribution is used to form the readiness states of agents for monetary transactions and $\ln N(\mu_m, \sigma_m^2) > 0$,

V. if the log-normal distribution is used to form the readiness states of agents for monetary transactions and $\ln N(\mu_m, \sigma_m^2) = 0$,

VI. if the beta distribution is used to form the readiness states of agents for monetary transactions.

Here,

$\ln N(\mu_b, \sigma_b^2)$, $\ln N(\mu_m, \sigma_m^2)$ are random variables having log-normal distributions with parameters $\mu_b, \sigma_b^2$ and $\mu_m, \sigma_m^2$, where $\mu_b, \mu_m \in [-1, 1]$, $\sigma_b^2, \sigma_m^2 \in (0, 1]$;

$\text{Beta}(\alpha_b, \beta_b)$, $\text{Beta}(\alpha_m, \beta_m)$ are random variables having beta distributions with parameters $\alpha_b, \beta_b$ and $\alpha_m, \beta_m$, respectively.

The value of the utility function of the $i^{th}$ agent ($i \in \{\tilde{i} : \hat{i} \in I, \gamma_{\tilde{i}\hat{i}}(t_k) = 1\}$) at moment $t_k$ ($t_k \in T$) calculated as:

$$u_i(t_k) = \gamma_{\tilde{i}\hat{i}}(t_k)\left((\delta_{\tilde{i}\hat{i}}(t_k) + 1)^{-\nu} - \lambda r\right), \quad (5)$$

where

$r \in [1, \bar{r}]$ is the trade interaction radius, i.e. the range of cells of the discrete location space of agents considered to be neighbors, $\bar{r}$ is the maximum allowable distance between interacting agents;

$\{\nu, \lambda\}$ are coefficients that determine the impact of the costs of the distance between the target and the purchased product, as well as between the buyer and the seller, respectively.

The main control parameters of such the MSES are as follows: the configuration of the initial location of agents in space, the radius of trade interaction, the

'contractability' coefficient, the parameters of log-normal and beta distributions used to form the states of readiness of agents to conclude transactions, the probability of moving agents in space, etc.

Each agent-buyer maximizes its own utility function over the set of control parameters under constraints that have a clear physical and economic meaning. At the same time, the average (over the population of agents) utility of future consumption can be considered as the integral objective function of the MSES:

$$U = \frac{1}{|I|}\sum_{i=1}^{|I|}\sum_{t=1}^{|T|} u_i(t_k). \quad (6)$$

The software implementation of model (1)–(6) is made in the FLAME GPU environment using C++ and the graphics processing unit (GPU) architecture, which allows us, in particular, to parallelize the agent behavior logic through special functions of the FLAMEGPU_AGENT_FUNCTION type.

*Table 1* presents the main functions developed for the considered model with their input and output parameters.

## 2. Synthesis procedure for artificial neural network

The MLP (Multilayer perceptron) model was chosen as the main configuration of the ANN for the task of approximating the objective function of the studied MSES (i.e., the agent-based model of trade interactions).

The most important hyperparameters of the designed ANN which significantly affect the approximation quality are as follows:

$\mu > 0$ is the initial learning rate;

$L = \{l_1, l_2, ..., i_{|L|}\}$ is the number of hidden layers in MLP;

$n_l > 0$, $l \in L$ is the number of neurons in each of the available hidden layers, $n \in N$, where $N$ is the set of all neurons;

$F_l \in \{TANH, ELU, HSig\}$, $l \in L$ is the activation function used for all neurons of the $l^{th}$-hidden layer (hyperbolic tangent, exponential linear, 'hard' sigmoid);

$w_{n_l}$, $n_l \in N$, $l \in L$ are weight coefficients of the $n_l^{ths}$-neurons of the $l^{th}$-hidden layer.

At the same time, the main criterion for the quality estimation of the ANN in the system being considered is the training error (the loss function), which should be minimized by the set of hyperparameters and weights:

$$\min_{\{\tilde{r}, |L|, n_l, F_l\}} E, n_i \in N, l \in L, \quad (7)$$

where

$$E = \frac{1}{|M|}\sum_{m=1}^{|M|}\left(\tilde{U}_{qm}(\mu, |L|, n_l, F_l, w_{n_l} X_m) - \hat{U}_m(X_m)\right)^2, \quad (8)$$

Here,

$M = \{m_1, m_2, ..., |M|\}$ is the set of training sample data, where $|M|$ is the size of the training sample;

$Q = \{q_1, q_2, ..., |Q|\}$ is the set of iterations of the ANN learning algorithm, $|Q|$ is all iterations of the machine learning algorithm;

$\tilde{U}_{qm}$, $q \in Q$, $m \in M$ are the approximated values of the objective function (future consumption utility function) at the ANN output, calculated for the $m^{th}$-data sample at the $q^{th}$-training epoch;

$X_m = \{x_{1m}, x_{2m}, ..., x_{|Xm|}\}$, $m \in M$ is the set of values of independent variables of the $m^{th}$ training sample (input layer of the ANN);

$\hat{U}_m(X_m)$, $m \in M$ are the known (factual) values of the objective function calculated using the previously developed agent-based model for the given $X_m$-set of input parameters values with the use of the Monte Carlo type method [20, 21].

*Figure 2* shows a block diagram of the developed two-step ANN synthesis procedure for its further use as a surrogate model in optimization experiments.

In *Fig. 2* the following denotes are used:

*Table 1.*

**The main functions and procedures of the stochastic model of the goods exchange**

| Function name | Description | Input parameters | Output parameters |
|---|---|---|---|
| FLAMEGPU_INIT_FUNCTION (init_function) | Model parameters initialization. Creation of a population of agents and their placement in a discrete space. | No | No |
| FLAMEGPU_EXIT_CONDITION (exit_condition) | Calculation of the objective function and checking the stop criterion. | No | No |
| FLAMEGPU_AGENT_FUNCTION (all_agents, flamegpu::MessageNone, flamegpu::MessageArray2D) | Sending data about each agent. | No | Coordinates, agent type, state, etc. |
| FLAMEGPU_AGENT_FUNCTION (all_products, flamegpu::MessageNone, flamegpu::MessageArray2D) | Sending product data by each agent. | No | Available product index, target product index. |
| FLAMEGPU_AGENT_FUNCTION (seeking_and_getting_product, flamegpu::MessageArray2D, flamegpu::MessageArray2D) | Search and acquisition of the target product through the goods exchange or for money. Search for the nearest seller with the desired product. Implementation of a monetary or barter transaction. Recalculation of the value of individual utility function. Sending data about the purchased product and money in the case of a monetary transaction. | Available product index, target product index. | The index of the purchased product and the money for the product (if the purchase is for money). |
| FLAMEGPU_AGENT_FUNCTION (getting_product_or_money, flamegpu::MessageArray2D, flamegpu::MessageNone) | Receiving a product (in barter) or money from a buyer (in a monetary transaction). Recalculation of the value of individual utility function. Completion of a trade transition. | The index of the purchased product and the money for the product (if the purchase is for money). | No |
| FLAMEGPU_AGENT_FUNCTION (update_agent_state, flamegpu::MessageNone, flamegpu::MessageNone) | Update the state of each agent and produce a new product if there was a trade deal in the previous step. | No | No |
| FLAMEGPU_AGENT_FUNCTION (update_cell, flamegpu::MessageArray2D, flamegpu::MessageNone) | Update the state of each cell of the discrete space. Checking the availability of a cell for occupation by agents. | Coordinates, agent type, state, etc. | No |
| FLAMEGPU_AGENT_FUNCTION (looking_for_resource, flamegpu::MessageArray2D, flamegpu::MessageArray2D) | Search for an agent that can be placed in a given cell with a given probability. | Coordinates, agent type, state, etc. | Coordinates of target cell. Information about the agent being moved. |
| FLAMEGPU_AGENT_FUNCTION (moving_trasaction, flamegpu::MessageArray2D, flamegpu::MessageNone) | Random movement of agents in a discrete space to a target cell. | Coordinates of target cell. Information about the agent being moved. | No |

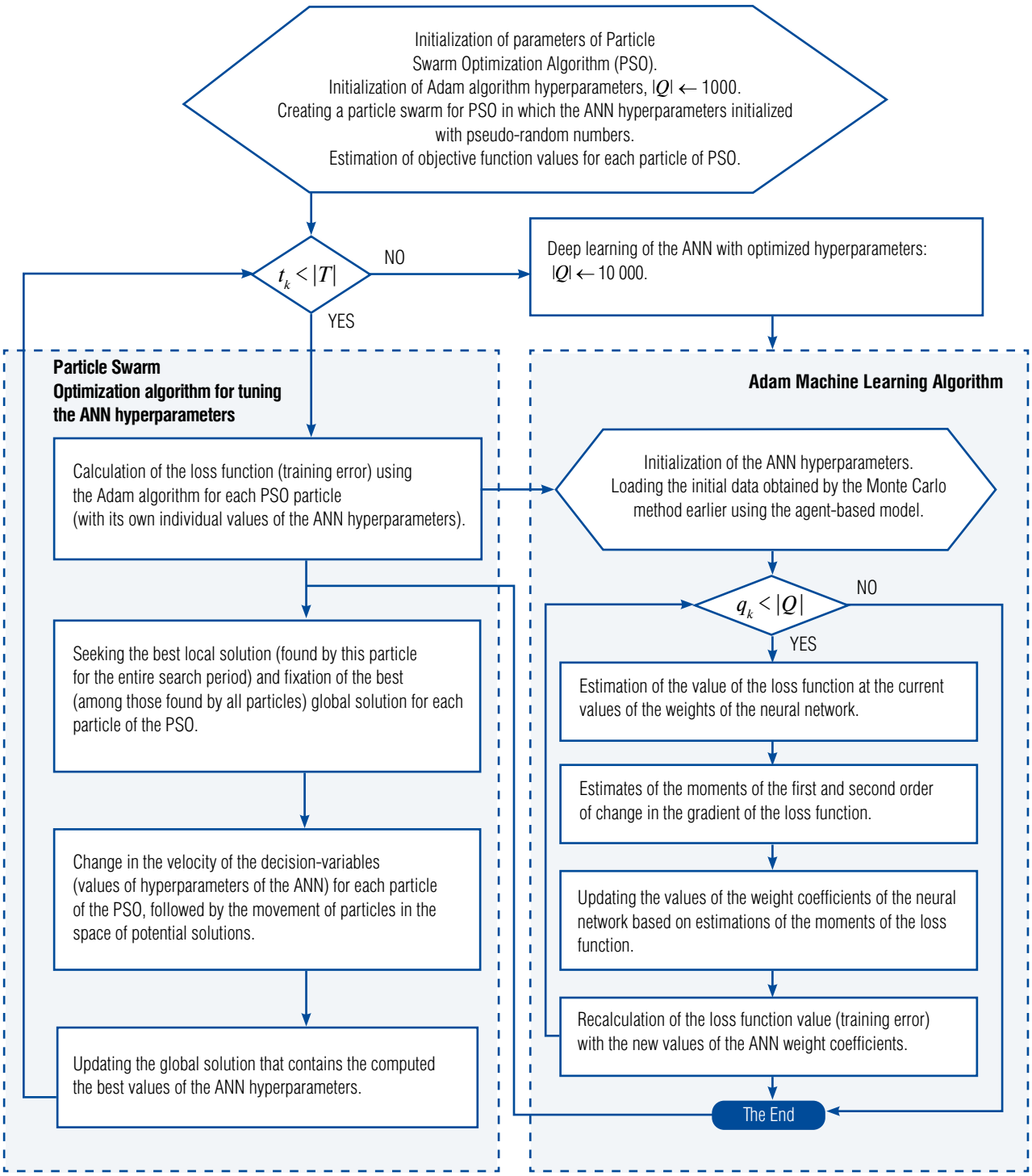


*Fig. 2.* Procedure for synthesising the artificial neural network using the particle swarm method and the Adam algorithm.

$T = \{t_0, t_1, ..., |T|\}$ is the set of swarm algorithm iterations (PSO), $|T|$ is the total number of iterations of the particle swarm algorithm; $t_0 \in T$, $t_{|T|} \in T$ is the initial and final iterations of the particle swarm algorithm;

$Q = \{q_0, q_1, ..., |Q|\}$ is the set of epochs of the training algorithm (the Adam), $|Q|$ is the total number of epochs of the training algorithm; $q_0 \in Q$, $q_{|Q|} \in Q$ are the initial and final epochs of the training algorithm.

On the first step, with a relatively small total number of epochs of the training algorithm ($|Q| = 1000$), the ANN hyperparameters are optimized with the use of the particle swarm optimization algorithm (PSO) aggregated trough the objective function with the Adam machine learning algorithm.

At the second step, after the best values of ANN hyperparameters are determined, deep learning is carried out using the formed ANN with a significantly larger number of epochs of the training algorithm – $|Q| = 10000$.

**The Particle Swarm Optimization Algorithm (PSO) [16]** provides the recalculation of ANN's hyperparameters towards improving the value of the objective function, i.e. minimizing the training error of the ANN calculated using a machine learning algorithm (such as the Adam [22]). The advantage of the particle swarm algorithm is a significantly higher time-efficiency, for example, in comparison with classical genetic algorithms, which also can be used to tune ANN hyperparameters.

Within the framework of the procedure we developed, the PSO algorithm is aggregated by the objective function (learning error of the ANN) with the Adam algorithm (*Fig. 2*). When using the particle swarm algorithm (PSO), the velocity vector of change in the desired variables values (i.e., the ANN's hyperparameters) is calculated, which determines the position of $i^{ths}$-particles ($i \in I$) in the potential solutions space at moment $t_k$ ($t_k \in T$, $k = 1, 2, ..., K$):

$$\mathbf{v}_i(t_k) = \theta \mathbf{v}_i(t_{k-1}) + c_1 h(0, 1)(\mathbf{x}_i^*(t_{k-1}) - \mathbf{x}_i(t_{k-1})) + \\ + c_2 e(0, 1)(\mathbf{x}^g(t_{k-1}) - \mathbf{x}_i(t_{k-1})), \quad (9)$$

$$\mathbf{x}_i(t_k) = \begin{cases} \mathbf{x}_i(t_{k-1}) + \mathbf{v}_i(t_{k-1}), \\ \text{if } \mathbf{x}_i(t_{k-1}) + \mathbf{v}_i(t_{k-1}) \in [\underline{\mathbf{x}}, \overline{\mathbf{x}}], \\ \mathbf{x}_i(t_{k-1}), \\ \text{if } \mathbf{x}_i(t_{k-1}) + \mathbf{v}_i(t_{k-1}) \notin [\underline{\mathbf{x}}, \overline{\mathbf{x}}], \end{cases} \quad (10)$$

where

$I = \{i_1, i_2, ..., i_{|I|}\}$ is the set of particle indices PSO, where $|I|$ is the total number of particles;

$\mathbf{x}_i^*(t_{k-1})$, $\mathbf{x}_i^*(t_{k-1})$ are the best potential values of ANN's hyperparameters obtained by the $i^{th}$-particle of the PSO during the search period and all particles at moment $t_k$ ($t_{k-1} \in T$);

$h(0, 1)$, $e(0, 1)$ are random variables uniformly distributed on the range [0, 1];

$\theta$, $c_1$, $c_2$ are constants the values of which can be set in the following ranges: $\theta \in [0.4, 1.4]$, $c_1 \in [1.5, 2]$, $c_1 \in [2, 2.5]$.

**The Adam machine learning algorithm [22]** provides a calculation of individual training rates for different ANN's parameters (*Fig. 2*) using estimations of the first- and second- gradient moments to adapt the training rate for each neural network weight.

In the Adam algorithm the following rule uses for updating neural network weights:

$$w_q = w_{q-1} - \frac{\eta}{\sqrt{\frac{\tilde{\beta}_2 \upsilon_{q-1} + (1-\tilde{\beta}_1)\left(\nabla_w E_q\right)^2}{1-\tilde{\beta}_2}} + \varepsilon} \times \\ \times \frac{\tilde{\beta}_1 m_{q-1} + (1-\tilde{\beta}_1)\nabla_w E_q}{1-\tilde{\beta}_1}, \quad (11)$$

where

$\nabla_w E_q$ is the gradient of the loss function at epoch $q$ ($q \in Q$);

$w$ are weight coefficients of the neural network;

$m_{q-1}$, $\upsilon_{q-1}$ are estimations of the first- and second-order moments of the change in the gradient of the loss function at epoch $q$ ($q \in Q$);

$\tilde{\beta}_1$, $\tilde{\beta}_2$ are hyperparameters of the Adam algorithm (that usually set up at the level of 0.9 and 0.99, respectively);

$\varepsilon$ is a fairly small number.

The proposed synthesis procedure of the ANN using the particle swarm method and the Adam machine learning algorithm is implemented using C++ and OpenNN. The reason for choosing the OpenNN as a framework for machine learning is to provide a sufficiently high performance comparable to such popular software libraries as the TensorFlow and PyTorch with a more convenient and simple development environment [19].

As a result of executing the given procedure, automatic generation of software modules of the synthesized ANN in C++ and Python is provided, which can be used as surrogate models in parallel real-coded genetic algorithms (RCGAs) [2, 3, 23], and this makes it possible to radically increase the time-efficiency of the optimal solution search procedure for multi-agent socio-economic systems.

The architecture of the ANN formed with the use of data obtained through the proposed agent-based model of trade interactions consists of an input layer, one hidden layer and an output layer. The first layer provides distribution of input signals to other neurons. At the same time, the input signals in the ANN are the scaled (normalized) values of the agent model control parameters, in particular, as follows:

- configuration (numerical equivalent) of the initial distribution of agents in the MSES;
- parameters of the log-normal and beta distributions used to form the readiness states of agents for transactions at different moments;
- trade interaction radius;
- probability of moving agents in space, etc.

The intermediate layer consists of ten neurons that use the hyperbolic tangent as the activation function.

The output layer consists of one neuron, which provides the calculation of the approximate value of the objective function – the average utility of future consumption.

Further, the synthesized ANN was integrated through objective function with the previously proposed parallel multi-agent genetic algorithm MA-RCGA [23] implemented with the FLAME GPU 2 framework [18]. This made it possible to significantly increase the time-efficiency of the evolutionary search procedure when solving the important problem of the MSES to maximize the function of the average utility of future consumption (6). The possibilities of using ANN-based surrogate models to improve the performance of evolutionary algorithms are described in [14] in more detail.

## 3. Results of numerical and optimization experiments

*Figure 3* presents the sensitivity analysis for the process time and accuracy of the solution obtained (i.e., the closeness of the solution to the known optimum). Depending on the frequency of using the proposed agent-based MSES model for calculating the objective and fitness functions we present the corresponding ANN-based surrogate model instead. Numerical experiments were carried out using the local supercomputer CEMI RAS – DSWS PRO (2 x Intel Xeon Silver 4114, 1 x NVIDIA QUADRO RTX 600) using 10 parallel interacting agents-processes that implement the evolutionary search procedure. Such a procedure is based on well-known heuristic selection operators, crossovers and mutations. At the same time, both the original (reference) agent-based model of trade interactions and the corresponding ANN-based surrogate model are used to recalculate the values of the objective and fitness functions.

As follows from *Fig. 3*, even under conditions of the prevailing use of the ANN-based surrogate model, a sufficiently high level of accuracy of the solution obtained is ensured (more than 95%). At the same time, a significantly greater time-efficiency of the solution search procedure is achieved in comparison with the approach in which only the original (reference) agent-based model is used to recalculate the objective and fitness functions values in MA-RCGA (*Fig. 3*). Thus, the accuracy of the optimization algorithm with periodic use of the ANN-based surrogate model directly depends on the approximation characteristics of the corresponding ANN.

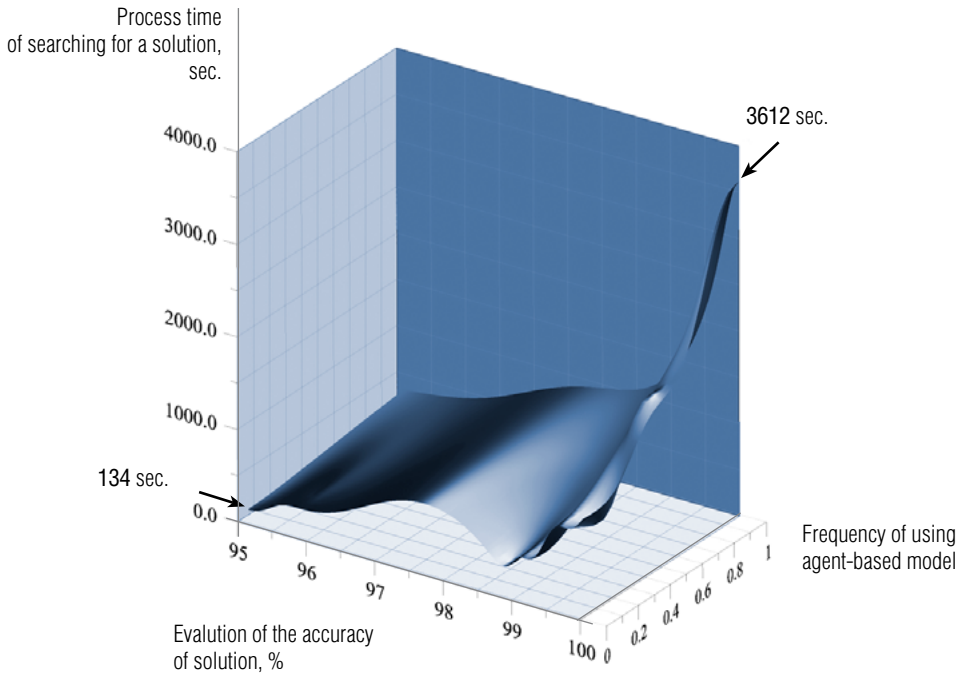


*Fig. 3.* Analysis of the sensitivity for the process time and accuracy of the solution obtained depending on the frequency of using the reference agent-based model.

In *Fig. 4* we present the results of optimization experiments on maximizing the average (for an ensemble of agents) utility of future consumption performed using the previously developed genetic algorithm MA-RCGA [23] and the generated ANN-based surrogate model. At the same time, various configurations of the initial arrangement of agents in space and various ways of forming the states of readiness of agents for transactions are used.

Two important conclusions follow from *Fig. 4*. Firstly, the choice of the initial configuration for agents' arrangement in space affects the value of the objective function of the MSES studied – the average utility of future consumption, and secondly, it affects the choice of a rational way to form states of readiness for transactions. In particular, for some configurations (*Fig. 4*) the use of the beta distribution is more preferable than the log-normal distribution.

## Conclusion

This article presents a novel approach to modeling and optimizing strategies for making individual decisions in large-scale multi-agent socio-economic systems (MSES) using the proposed agent-based model of trade interactions as the example. A new procedure for the synthesis and training of an artificial neural network (ANN) has been developed based on the combined use of particle swarm optimization methods (to determine the optimal values of hyperparameters) and the Adam machine learning algorithm (to calculate the weight coefficients of the ANN). The ANN we designed related to the class of a multilayer perceptron (MLP) and is used as a surrogate model embedded in a previously developed multi-agent genetic algorithm (MA-RCGA) to approximate the values of the objective and fitness functions – the average (over an ensem-

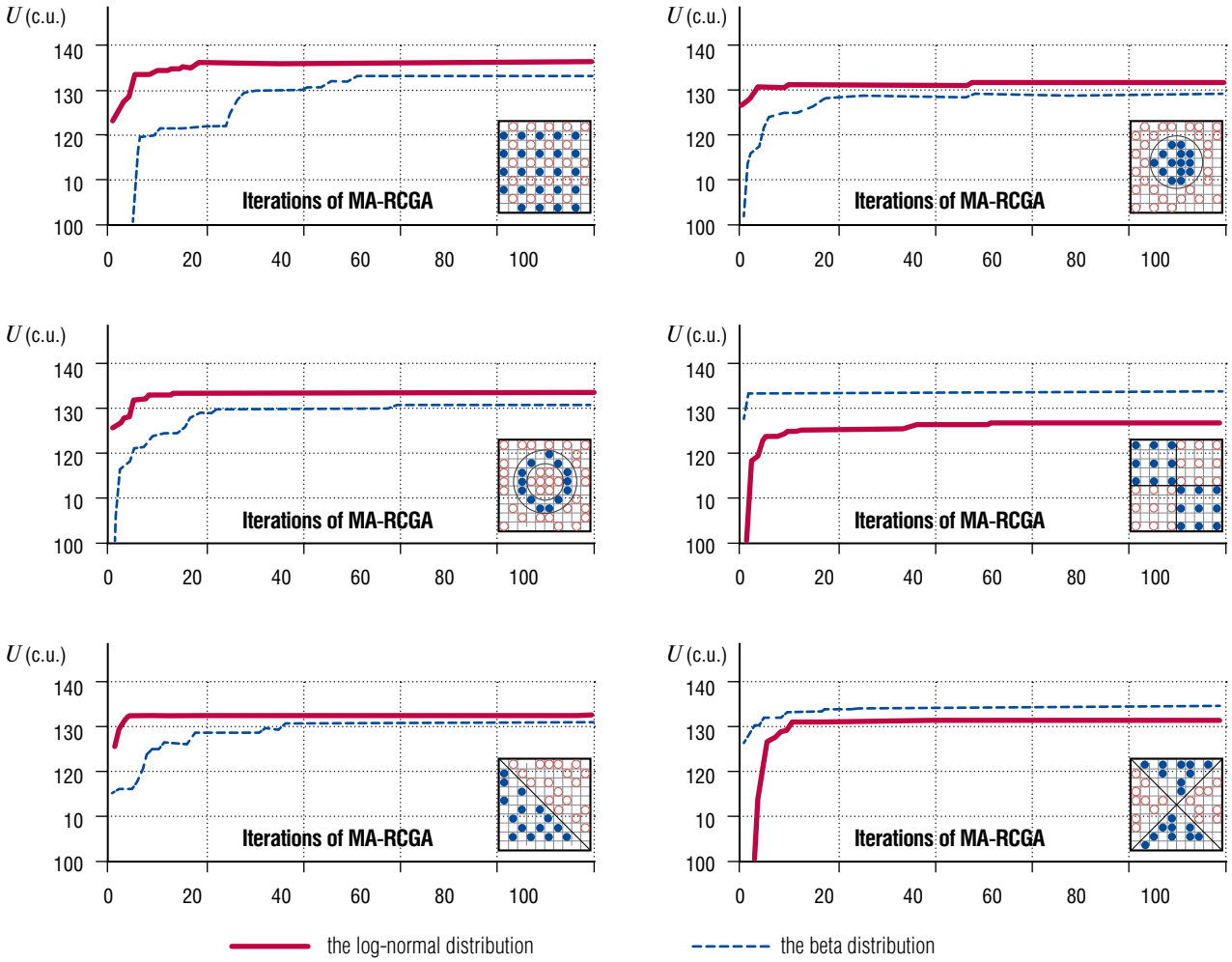


*Fig. 4.* Results of optimization experiments for maximizing the average utility of future consumption ($U$) when forming the states of readiness for transactions.

ble of agents) utility of future consumption. As a result of the numerical studies, it is shown that even under the conditions of the prevailing use of the ANN-based surrogate model, a sufficiently high level of accuracy for the solution obtained is provided. At the same time, the choice of the initial configuration for agents' arrangement in space affects the value of the objective function and the optimal way to form the states of agent readiness for transactions.

Further research will be aimed at creating simulation models of large-scale multi-agent socio-economic systems using machine learning methods and genetic optimization algorithms for optimal control of the ensemble of interacting economic agents' behavior. ■

**Acknowledgments**

The research was supported by RSF (project No. 23-21-00012).

**References**


1. Akopov A.S. (2014) Parallel genetic algorithm with fading selection. *International Journal of Computer Applications in Technology*, vol. 49, nos. 3–4, pp. 325–331. https://doi.org/10.1504/IJCAT.2014.062368

2. Akopov A.S., Beklaryan A.L., Thakur M., Verma B.D. (2019) Developing parallel real-coded genetic algorithms for decision-making systems of socio-ecological and economic planning. *Business Informatics*, vol. 13, no. 1, pp. 33–44. https://doi.org/10.17323/1998-0663.2019.1.33.44

3. Akopov A.S., Beklaryan L.A., Thakur M. (2022) Improvement of maneuverability within a multiagent fuzzy transportation system with the use of parallel biobjective real-coded genetic algorithm. *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 12648–12664. https://doi.org/10.1109/TITS.2021.3115827

4. Akopov A.S., Beklaryan L.A. (2022) Simulation of rates of traffic accidents involving unmanned ground vehicles within a transportation system for the 'smart city'. *Business Informatics*, vol. 16, no. 4, pp. 19–35. https://doi.org/10.17323/2587-814X.2022.4.19.35

5. Khachatryan N.K., Akopov A.S. (2017) Model for organizing cargo transportation with an initial station of departure and a final station of cargo distribution. *Business Informatics*, no. 1 (39), pp. 25–35. https://doi.org/10.17323/1998-0663.2017.1.25.35

6. Saaty T. (1993) *Decision making. The Analytic Hierarchy Process.* Translation from English by R.G. Vachnadze. Moscow: Radio and communication (in Russian).

7. Kravchenko T.K., Isaev D.V. (2017) *Decision support systems*. Moscow: Urait (in Russian).

8. Kravchenko T.K., Bruskin S.N., Isaev D.V., Kuznetsova E.V. (2020) Prioritization of IT product backlog items using decision support systems. *Informacionnye Tehnologii*, vol. 26, no. 11, pp. 631–640 (in Russian). https://doi.org/10.17587/it.26.631-640

9. Kravchenko T., Shevgunov T. (2023) Equivalent exchange method for decision-making in case of alternatives with incomparable attributes. *Inventions*, vol. 8, no. 1, article 12. https://doi.org/10.3390/inventions8010012

10. Pospelov I.G. (2018) A model of random sales. *Mathematical Notes*, vol. 103, pp. 453–465. https://doi.org/10.1134/S0001434618030112

11. Makarov V., Bakhtizin A., Epstein J. (2022) Agent-based modelling for a complex world. Part 1. *Economics and the Mathematical Methods*, vol. 58, no. 1, pp. 5–26 (in Russian). https://doi.org/10.31857/S042473880018970-6

12. Makarov V., Bakhtizin A., Epstein J. (2022) Agent-based modelling for a complex world. Part 2. *Economics and the Mathematical Methods*, vol. 58, no. 2, pp. 7–21 (in Russian). https://doi.org/10.31857/S042473880020009-8

13. Tan R.K., Qian C., Wang M., Ye W. (2022) An efficient data generation method for ANN-based surrogate models. *Structural and Multidisciplinary Optimization*, vol. 65, article 90. https://doi.org/10.1007/s00158-022-03180-6

14. Díaz-Manríquez A., Toscano G., Barron-Zambrano J.H., Tello-Leal E. (2016) A review of surrogate assisted multiobjective evolutionary algorithms. *Computational Intelligence and Neuroscience*, vol. 2016, article 9420460. https://doi.org/10.1155/2016/9420460

15. Herrera F., Lozano M., Verdegay J.L. (1998) Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial Intelligence Review*, vol. 12, no. 4, pp. 265–319. https://doi.org/10.1023/A:1006504901164

16. Bonyadi M.R., Michalewicz Z. (2017) Particle swarm optimization for single objective continuous space problems: a review. *Evolutionary Computation*, vol. 25, no. 1, pp. 1–54.

17. Beklaryan L.A., Flerova A.Y., Zhukova A.A. (2018) *Optimal control methods: textbook*. MIPT (in Russian).

18. Richmond P., Chisholm R., Heywood P., Leach M., Kabiri C.M. (2021) *FLAME GPU Zenodo*. https://doi.org/10.5281/zenodo.5428984

19. Lopez R. (2014) *Open NN: An open source neural networks C++ library [software]*. Available at: http://www.cimne.com/flood (accessed 10 May 2023).

20. Whitelam S., Selin V., Benlolo I., Casert C., Tamblyn I. (2022) Training neural networks using Metropolis Monte Carlo and an adaptive variant. *Machine Learning: Science and Technology,* vol. 3, article 045026. https://doi.org/10.1088/2632-2153/aca6cd

21. de Freitas J.F., Niranjan M., Gee A.H., Doucet A. (2000) Sequential Monte Carlo methods to train neural network models. *Neural Computation*, vol. 12, no. 4, pp. 955–993. https://doi.org/10.1162/089976600300015664

22. Kingma D.P., Ba J. (2014) Adam: A method for stochastic optimization. *arXiv preprint*, arXiv:1412.6980. https://doi.org/10.48550/arXiv.1412.6980

23. Akopov A.S., Beklaryan L.A., Thakur M., Verma B.D. (2019) Parallel multi-agent real-coded genetic algorithm for large-scale black-box single-objective optimisation. *Knowledge-Based Systems*, vol. 174, pp. 103–122. https://doi.org/10.1016/j.knosys.2019.03.003

**About the author**

**Andranik S. Akopov**

Dr. Sci. (Tech.), Professor, Professor of the Russian Academy of Sciences;

Chief Researcher, Laboratory of Dynamic Models of Economy and Optimisation, Central Economics and Mathematics Institute, Russian Academy of Sciences, 47, Nachimovky Prospect, Moscow 117418, Russia;

E-mail: akopovas@umail.ru

ORCID: 0000-0003-0627-3037