


Article

Investigation of Optimization Algorithms for Neural Network Solutions of Optimal Control Problems with Mixed Constraints

Irina Bolodurina and Lyubov Zabrodina * 

Department of Applied Mathematics, Orenburg State University, 460018 Orenburg, Russia; prmat@mail.osu.ru
* Correspondence: zabrodina97@gmail.com; Tel.: +7-(987)-89-21-537

Abstract: In this paper, we consider the problem of selecting the most efficient optimization algorithm for neural network approximation—solving optimal control problems with mixed constraints. The original optimal control problem is reduced to a finite-dimensional optimization problem by applying the necessary optimality conditions, the Lagrange multiplier method and the least squares method. Neural network approximation models are presented for the desired control functions, trajectory and conjugate factors. The selection of the optimal weight coefficients of the neural network approximation was carried out using the gravitational search algorithm and the basic particle swarm algorithm and the genetic algorithm. Computational experiments showed that evolutionary optimization algorithms required the smallest number of iterations for a given accuracy in comparison with the classical gradient optimization method; however, the multi-agent optimization methods were performed later for each operation. As a result, the genetic algorithm showed a faster convergence rate relative to the total execution time.

Keywords: optimal control problem; evolutionary optimization; neural network approach



Citation: Bolodurina, I.; Zabrodina, L. Investigation of Optimization Algorithms for Neural Network Solutions of Optimal Control Problems with Mixed Constraints. *Machines* **2021**, *9*, 102. <https://doi.org/10.3390/machines9050102>

Academic Editors: Sergey M. Andreev and Vadim R. Gasiyarov

Received: 10 April 2021
Accepted: 5 May 2021
Published: 17 May 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Currently, there are many computer modeling problems that require reduction to the class of optimal control problems (OCP) to automate the process of finding a solution, as well as to reduce the complexity of calculations. The most common tool for solving such problems is numerical methods, including the apparatus of artificial neural networks (ANN). The advantage of ANN is the ability to construct a solution in the form of a functional dependence with high accuracy [1,2]. At the same time, the use of other numerical optimization methods requires the subsequent interpolation of the discrete solution, which, in turn, imposes an additional error. In addition, the efficiency of using the neural network approach for solving the OCP is expressed in the possibility of obtaining a solution not only satisfying the necessary optimality conditions, but also smoothness conditions.

In the study [3], A. Nazemi and R. Karami presented the main stages of solving OCP with mixed constraints. Based on the optimal Karush–Kuhn–Tucker conditions, the authors constructed a function for calculating the error and formulated a nonlinear optimization problem, in which neural network approximations are defined for the state function, control, and Lagrange multipliers. For the obtained scheme of dynamic optimization of the weight coefficients of the neural network solution, the analysis of stability and convergence is carried out.

The most relevant application of the neural network approach is in cases where the solution of optimal control problems does not have an analytical solution [4] or the type of analytical solution cannot be determined directly [5,6]. In addition, the approach under study does not require the search for the optimal ratio of the step size along the axes under consideration, in contrast to finite-dimensional methods [7].

In the study [8], the authors T. Kmet and M. Kmetova considered the use of ANN for solving an OCP with a free end time and constraints on control and phase trajectory.

Note that the construction of the corresponding nonlinear programming problem was carried out on the basis of neural network control using adaptive criticism. The proposed approach demonstrated the effectiveness in the field of optimal management of photosynthetic production and confirmed the relevance of the introduction of an adaptive critical system approach.

The authors of the study [9], F. Kheyrintaj and A. Nazemi, presented a numerical method for solving the OCP with fractional delay, which implements nonlinear polynomial expansions in a neural network with an adaptive structure. The developed neural network approach has demonstrated its effectiveness in many specific examples.

The presented neural network solutions, as well as other modifications of the ANN for solving the OCP [10–12], can be combined and the main advantage can be highlighted—the presentation of the problem solution as continuous over the entire domain of definition, while the rest of the numerical methods allow you to find values directly at discrete points and require interpolation to calculate the values between them [13]. In addition, with an increase in the number of partitions of the original domain of definition [14] and the dimension of the problem [15], as a rule, the computational complexity does not increase significantly.

In this paper, we consider the computational features of the neural network approach to solving optimal control problems using various methods of evolutionary optimization. This paper describes the structure of a neural network solution that satisfies the Kreinovich theorem and corresponds to a hole-layer perceptron, and also presents a general scheme for optimizing its parameters. This area of research is relevant, since the use of evolutionary algorithms in many applied areas [16–19] has made it possible to increase the efficiency of finding optimal values even in the case of analyzing functions that have many local optima or do not have a clear global optimum.

Note that the evolutionary algorithms not only allow us to obtain well-interpreted results, but also quite simply combine with other methods [20–22]. At the same time, these algorithms do not provide a strict finding of the optimal solution in a finite time and require adjustment of the parameters of the models used [23–25]. Thus, evolutionary algorithms are heuristic, that is, the accuracy and rigor of the solution do not prevail over the feasibility. In this regard, the use of this type of optimization methods in solving applied problems requires a more detailed analysis.

In this regard, this study is aimed at studying the computational features of the neural network approach to solving optimal control problems with mixed constraints using evolutionary optimization algorithms: the genetic algorithm, the gravitational search algorithm, and the basic particle swarm algorithm. Thus, this paper is devoted to the study of the neural network approach to solving optimal control problems and optimizing the structure of the functional representation by evolutionary algorithms in order to analyze their efficiency and convergence rate. This paper contains examples of optimal control problems that have an analytical solution and allow us to evaluate the convergence of the algorithms under study.

The rest of the paper is organized as follows: Section 2 describes a formal mathematical formulation of the optimal control problem with mixed constraints. Section 3 presents a neural network model for approximating the optimal control problem, as well as a description of the studied evolutionary algorithms for optimizing the neural network solution. In Section 4, the practical implementation of the presented algorithms is presented and the results are evaluated.

2. Necessary Optimality Conditions for OCP with Mixed Constraints

Consider the general formulation of the optimal control problem corresponding to the Bolza problem with mixed constraints. Let us introduce function $x : [t_0, t_1] \rightarrow \mathbb{R}^n, n \geq 1$, which describes the development of a certain system. Function $x(t)$ is called a state variable (or trajectory) satisfying the continuity condition.

Note that for the system at the initial moment of time the state is known:

$$x(t_0) = x_0 \quad (1)$$

We will assume that the further dynamics of the system depends on some particular choice (or strategy) at each moment of time and this strategy is given by function $u : [t_0, t_1] \rightarrow U \subset R^k, k \geq 1$. A fixed set U in R^k is called a control set, and function $u(t)$ is called a control function.

Based on the fact that $u(t)$ determines the development of the system, the dynamics of the state variable will take the form:

$$\dot{x}_i(t) = g_i(t, x(t), u(t)), \quad i = 1, \dots, n \quad (2)$$

where $g : [t_0, t_1] \times R^n \times R^k \rightarrow R^n$ and function g belongs to the space of continuously differentiable functions $g \in C^1([t_0, t_1] \times R^{n+k})$.

We say that a piecewise continuous function $u : [t_0, t_1] \rightarrow U$ is an admissible control for (1)–(2) if there is a unique solution to this ordinary differential equation, defined on $[t_0, t_1]$, and solution $x(t)$ is called the trajectory corresponding to control $u(t)$.

In addition, for each system, control constraints can be described in one way or another. Within the framework of this study, it is assumed that there are mixed constraints for the system, simultaneously linking control and a state variable:

$$c_i(t, x(t), u(t)) \leq 0, \quad i = 1, \dots, p \quad (3)$$

Let function $f : [t_0, t_1] \times R^{n+k} \rightarrow R$ describe variable costs (or operating costs, variable payments, etc.). Let the values of t_0, t_1 be fixed.

Consider the set C of admissible controls and define the quality criterion $J(\omega)$ as follows:

$$J(\omega) = \int_{t_0}^{t_1} F(t, x(t), u(t)) dt + \Phi(t_1, x(t_1)) \rightarrow \min \quad (4)$$

which describes the variable costs over the entire time interval and the trajectory constraints at the end of the time interval. Suppose that functions F and Φ also belong to the space of continuously differentiable functions, i.e., $F \in C^1([t_0, t_1] \times R^{n+k})$ and $\Phi \in C^1(R^n)$.

The optimal control problem (1)–(4) is a Bolza problem with mixed constraints, a free right end of the trajectory, and a fixed end time.

One of the most effective approaches to solving problems in optimal control theory is the Lagrange multiplier method. This method consists in reducing the original OCP to the solution of the problem of minimizing the corresponding Lagrange function:

$$L(t, x(t), u(t), \lambda(t), \lambda_0, \mu(t)) = -\lambda_0 F(t, x(t), u(t)) + \lambda(t) g(t, x(t), u(t)) + \mu(t) c(t, x(t), u(t)). \quad (5)$$

where $\lambda_0 \in R, \lambda(t) \in R^n, \mu(t) \in R^p$.

Then, based on the necessary optimality conditions, it can be argued that the optimal control $u^*(t)$, the corresponding trajectory $x^*(t)$, and the Lagrange multipliers must satisfy [3]:

- The dynamics of the state variable;

$$\dot{x}_i^* = g_i, \quad i = 1, \dots, n; \quad (6)$$

- Conditions of stationarity with respect to x and u , respectively:

$$\dot{\lambda}_i^* = -\frac{\partial F}{\partial x_i} - \sum_{i=1}^n \frac{\partial g_i}{\partial x_i} \lambda_i - \sum_{l=1}^p \frac{\partial c_l}{\partial x_i} \mu_l, \quad i = 1, \dots, n; \quad (7)$$

$$\frac{\partial F}{\partial u_s} + \sum_{i=1}^n \frac{\partial g_i}{\partial u_s} \lambda_i + \sum_{l=1}^p \frac{\partial c_l}{\partial u_s} \mu_l = 0, \quad s = 1, \dots, k; \quad (8)$$

- Additional non-rigidity conditions;

$$\mu_l \cdot c_l = 0, \quad l = 1, \dots, p; \quad (9)$$

- Nonnegativity condition;

$$\mu_l \geq 0, \quad l = 1, \dots, p; \quad (10)$$

- Mixed constraints;

$$c_l \leq 0, \quad l = 1, \dots, p; \quad (11)$$

- Initial conditions;

$$x_i^*(t_0) = x_{0i}, \quad i = 1, \dots, n; \quad (12)$$

- Transversality conditions;

$$\lambda_i^*(t_1) = \lambda_0^* \cdot \Phi_{x_i}(t_1, x^*(t_1)), \quad i = 1, \dots, n. \quad (13)$$

To simplify the solution of this system, we write down constraints (6)–(8) in an equivalent form. To do this, we introduce the concept of an NCP function that satisfies the nonlinear complementarity problem (NCP):

$$\varphi(a, b) = 0 \Leftrightarrow a \geq 0, \quad b \geq 0, \quad ab = 0. \quad (14)$$

To do this, we introduce the concept of the perturbed NCP-function of the Fischer–Burmeister:

$$\varphi_{FB}^\varepsilon(a, b) = (a + b) - \sqrt{a^2 + b^2 + \varepsilon}, \quad \varepsilon \rightarrow 0_+. \quad (15)$$

Using the perturbed NCP-function $\varphi_{FB}^\varepsilon(a, b)$, we transform conditions (6)–(8) into equality-type constraints in the form:

$$\varphi_{FB}^\varepsilon(\mu_l, -c_l) = 0, \quad \varepsilon \rightarrow 0_+, \quad i = 1, p. \quad (16)$$

Thus, the initial OCP is reduced to the problem of nonlinear optimization (6)–(8), (12)–(13), (16), which must be solved by means of neural networks. Consider the process of building a neural network model that approximates the solution to a nonlinear optimization problem.

3. Building a Neural Network Structure for Solving OCP

We represent the functional approximation as a sum of two parts based on two facts. The first term contains no tunable parameters and satisfies the initial or boundary conditions. The second term uses one output direct neural network with adjustable parameters and input signals.

It should be noted that in the second case, the weighting coefficients are adjusted, taking into account the minimization problem and are constructed so as not to contribute to the initial or boundary conditions. Based on these facts and using the neural networks, functional approximations of solutions for the phase trajectory, Lagrange multipliers and control for the system (6)–(8), (12), (13), and (16) can be defined as follows:

$$\left\{ \begin{array}{l} x_T = x_0 + (t - t_0)n_x, \\ \lambda_T = - \left. \frac{\partial \Phi}{\partial x_i} \right|_{t_1} + (t - t_1)n_\lambda, \\ \mu_T = n_\mu, \\ u_T = n_u. \end{array} \right. \quad (17)$$

where functions $n_x, n_\lambda, n_\mu, n_u$ have a structure:

$$\begin{cases} n_x = \sum_{i=1}^I v_x^i \sigma(z_x^i), z_x^i = w_x^i t + b_x^i \\ n_\lambda = \sum_{i=1}^I v_\lambda^i \sigma(z_\lambda^i), z_\lambda^i = w_\lambda^i t + b_\lambda^i \\ n_\mu = \sum_{i=1}^I v_\mu^i \sigma(z_\mu^i), z_\mu^i = w_\mu^i t + b_\mu^i \\ n_u = \sum_{i=1}^I v_u^i \sigma(z_u^i), z_u^i = w_u^i t + b_u^i \end{cases} \quad (18)$$

The described structure (18) corresponds to a neural network with a single hidden layer. The input signals x , passing through the neurons of the network and multiplying by their weight coefficients \vec{w} , eventually form a single weighted feature $\sum_i = \sum_j w_{ij}^{(0)} x_j$. The weighted trait \sum_i is further transformed using the activation function (some nonlinear converter) and forms the ANN output for the neuron on the next layer $\sigma(\sum_i)$ (Figure 1). At the output layer, the values of the neurons are multiplied by the weight coefficients of the next layer $w_i^{(1)}$, and the response (the output of the ANN) is considered to be the resulting weighted sum.

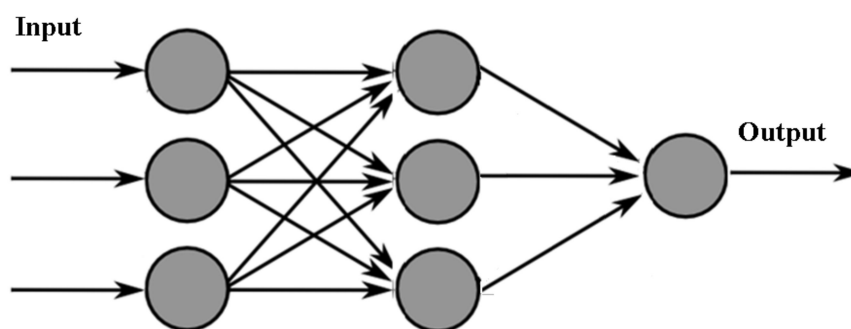


Figure 1. The structure of an artificial neural network for solving the OCP.

Thus, the function $N(t, \vec{p})$ has the form:

$$N(t, \vec{p}) = \sum_{i=1}^n w_i^{(1)} \sigma \left(\sum_{j=1}^m w_{ij}^{(0)} t_j + b_i^* \right),$$

where m is the number of neurons on the input layer, n is the number of neurons on the next layer.

The task is to select the optimal parameters of the ANN in such a way that the approximated function (17) satisfies the system (6)–(8), (16) with some error. The parameters of the ANN are the number of neurons in the input and output layers, their weight coefficients, and the activation function used.

The selection of the optimal weight coefficients of the ANN can be carried out using the algorithm of back propagation of the error, for which it is necessary to create an appropriate optimization problem.

We represent the system of Equations (6)–(8), (16) as an optimization problem: we divide the time interval $[t_0, t_1]$ into q equal parts by points $t_i = t_0 + i \cdot \Delta t$, then, based on the least squares method, we have the following formulation of the problem:

$$\min_y E(y) = \frac{1}{2} \sum_{h=1}^q \{E_1(t_h, y) + E_2(t_h, y) + E_3(t_h, y) + E_4(t_h, y)\}, \quad (19)$$

where $y = (w_x, w_\lambda, w_\mu, w_u, b_x, b_\lambda, b_\mu, b_u, v_x, v_\lambda, v_\mu, v_u)^T$ and

$$\left\{ \begin{array}{l} E_1(t_h, y) = \sum_{i=1}^n [\dot{x}_{T i}^h - g_i]^2, \\ E_2(t_h, y) = \sum_{i=1}^n [\dot{\lambda}_{T i}^h + \frac{\partial F_T^h}{\partial x_{T i}} + \sum_{i=1}^n \frac{\partial g_{T i}^h}{\partial x_{T i}} \lambda_{T i}^h + \sum_{l=1}^p \frac{\partial c_{T l}^h}{\partial x_{T i}} \mu_{T l}^h]^2, \\ E_3(t_h, y) = \sum_{s=1}^k [\frac{\partial F_T^h}{\partial u_{T s}} + \sum_{i=1}^n \frac{\partial g_{T i}^h}{\partial u_{T s}} \lambda_{T i}^h + \sum_{l=1}^p \frac{\partial c_{T l}^h}{\partial u_{T s}} \mu_{T l}^h]^2, \\ E_4(t_h, y) = \sum_{l=1}^p [\varphi_{FB}^\varepsilon(\mu_{T l}, -c_{T l})]^2, \varepsilon \rightarrow 0_+. \end{array} \right. \quad (20)$$

The finite-dimensional optimization problems (19) and (20) makes it possible to use various solution methods and select weight coefficients with a given accuracy for constructing a neural network approximation.

Thus, the ANN weighting coefficients are updated by optimizing the error function (19) by the error backpropagation algorithm, which can be schematically represented according to Figure 2.

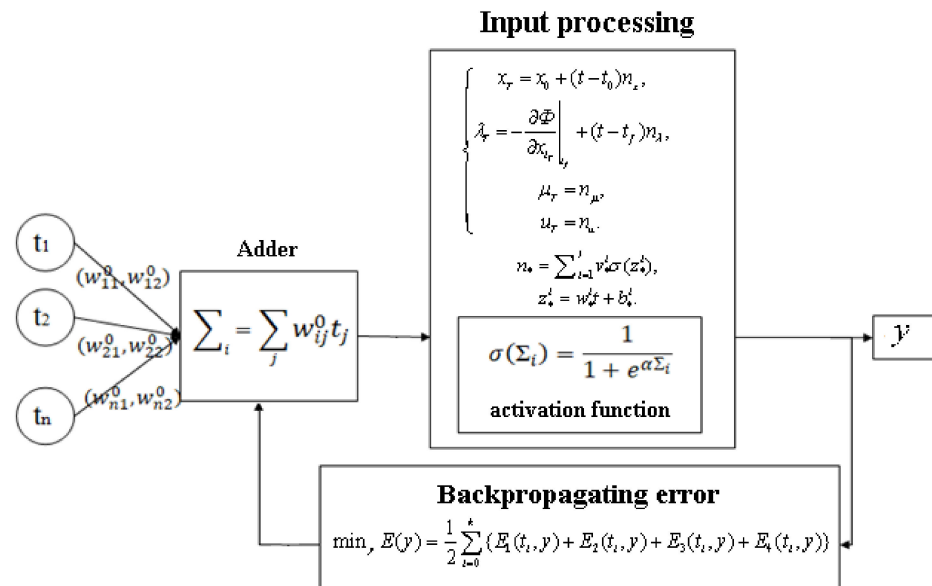


Figure 2. Schematic representation of optimization of a neural network solution.

3.1. General Optimization Scheme for A Neural Network Solution

Figure 2 shows a flowchart of a neural network solution (NNS) optimization process, and its specific steps are presented in Algorithm 1.

Algorithm 1 Optimization of a NNS.

Input: NNS, $\varepsilon_1, \varepsilon_2$

Output: y^*

- 1 put $i = 0$;
 - 2 initialize weight coefficients $y_i = (w_x, w_\lambda, w_\mu, w_u, b_x, b_\lambda, b_\mu, b_u, v_x, v_\lambda, v_\mu, v_u)^T$;
 - 3 calculate the values of the optimized function $E(y_i)$;
 - 4 update weight coefficients $y_{i+1} = (\tilde{w}_x, \tilde{w}_\lambda, \tilde{w}_\mu, \tilde{w}_u, \tilde{b}_x, \tilde{b}_\lambda, \tilde{b}_\mu, \tilde{b}_u, \tilde{v}_x, \tilde{v}_\lambda, \tilde{v}_\mu, \tilde{v}_u)^T$;
 - 5 calculate the values of the optimized function $E(y_{i+1})$;
 - 6 if $\|E(y_{i+1})\| \leq \varepsilon_1$ or $\|E(y_{i+1}) - E(y_i)\| \leq \varepsilon_2$ then
 go to step 7;
 - 7 else $y_i = y_{i+1}, i = i + 1$ and go to step 4;
 - 7 **Return** $y^* = y_{i+1}$.
-

Thus, the efficiency of the neural network solution search depends on the algorithm used to update and optimize the weight coefficients. This study analyzes the accuracy and convergence rate of a neural network approach with various evolutionary optimization algorithms: a genetic algorithm, a population gravity search algorithm, and a basic particle swarm algorithm. The constructed neural network approximations by various optimization methods are compared with the gradient descent algorithm.

3.2. Genetic Optimization Algorithm

Genetic algorithms search for the solution space of a function using simulated evolution, i.e., the survival of the most complex strategy. At the same time, the fittest individuals of any population tend to reproduce and survive until the next generation, thereby improving subsequent generations. However, lower individuals may accidentally survive and also reproduce.

Research has shown that genetic algorithms solve linear and nonlinear problems by exploring all areas of the state space and exponentially exploiting promising areas through mutations, crossovers, and selection operations applied to individuals in a population. A more complete discussion of genetic algorithms, including extensions and related topics, is presented in [26].

The genetic algorithm used in this study is shown in Figure 3.

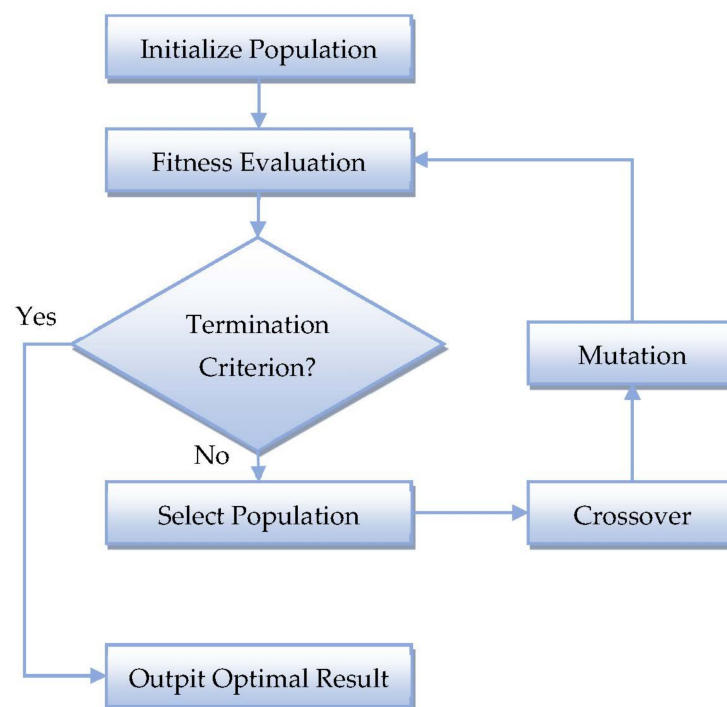


Figure 3. Basic structure of genetic algorithm.

To apply this algorithm, we define a population of size S as a set of different weights of the neural network approximation.

The fitness function for each set of ANN weights is the minimized error function (19) and (20), which ensures the fulfillment of the necessary optimality conditions.

The optimal set of ANN weighting parameters is such that the error of the approximation function is equal to the given computational accuracy.

The mutation operation is performed as a change in some weight by a random set of values.

The crossing of one set of weight coefficients with another is determined using one-point crossing over.

Selection (selection of the best individuals in the population) is carried out by preserving a certain percentage of the best individuals in the new generation.

The optimization process will go on until the specified computation error is reached.

3.3. Basic Particle Swarm Algorithm

The main idea of the particle swarm algorithm [27,28] is to move the population of possible solutions in the direction of the best found position of the particles (solution). This algorithm belongs to the class of multi-agent methods.

Algorithm 2 Basic Particle Swarm Algorithm.

Input: NNS, the size of the neighborhood $\sigma < N$, the values of the maximum influence $\varphi_{1,\max}$ and $\varphi_{2,\max}$, as well as the maximum speed v_{\max}

Output: b

- 1 initialize a random population of individuals $\{x_i\}, i \in [1, N]$, as well as the n-element velocity vector of each individual $\{v_i\}, i \in [1, N]$;
 - 2 calculate the best position for each individual so far : $b_i \leftarrow x_i, i \in [1, N]$;
 - 3 **if** $\|E(b_i)\| \leq \varepsilon$ **then**
to step 13;
 - else** put $i = 0$ and to step 3;
 - 4 put $H_i \leftarrow \{\sigma \text{ the nearest neighbors for } x_i\}$;
 - 5 put $h_i \leftarrow \operatorname{argmin}_x \{f(x) : x \in H_i\}$;
 - 6 generate a random vector φ_1 , where $\varphi_1 \in U[0, \varphi_{1,\max}]$;
 - 7 generate a random vector φ_2 , where $\varphi_2 \in U[0, \varphi_{2,\max}]$;
 - 8 calculate $v_i \leftarrow v_i + \varphi_1 \circ (b_i - x_i) + \varphi_2 \circ (h_i - x_i)$;
 - 9 **if** $|v_i| < v_{\max}$ **then**
adjust the speed $v_i \leftarrow v_i v_{\max} / |v_i|$;
 - 10 calculate $x_i \leftarrow x_i + v_i$;
 - 11 calculate $b_i \leftarrow \operatorname{argmin}_x \{f(x_i) : f(b_i)\}$;
 - 12 **if** $i < N$ **then**
put $i = i + 1$ and go to step 4;
 - else** go to step 3;
 - 13 **Return** $b = \operatorname{argmin}\{f(b_i), i = 1, N\}$.
-

At the initial moment of time, the particles are located chaotically throughout the space and contain a randomly specified velocity vector. For each particle located at a certain point, the value of the fitness function is determined and compared with its best location, as well as the best location relative to the entire population.

At each iteration, the direction and value of the particle velocity is corrected, relying on the approach to the best point among a given number of neighbors, as well as on the approach to the global optimum point. The proposed method is aimed at the fact that after a finite number of iterations, a larger number of particles will be located near the most optimal point. The particle swarm algorithm is shown in the flowchart in Figure 4, and its specific steps are shown in Algorithm 2.

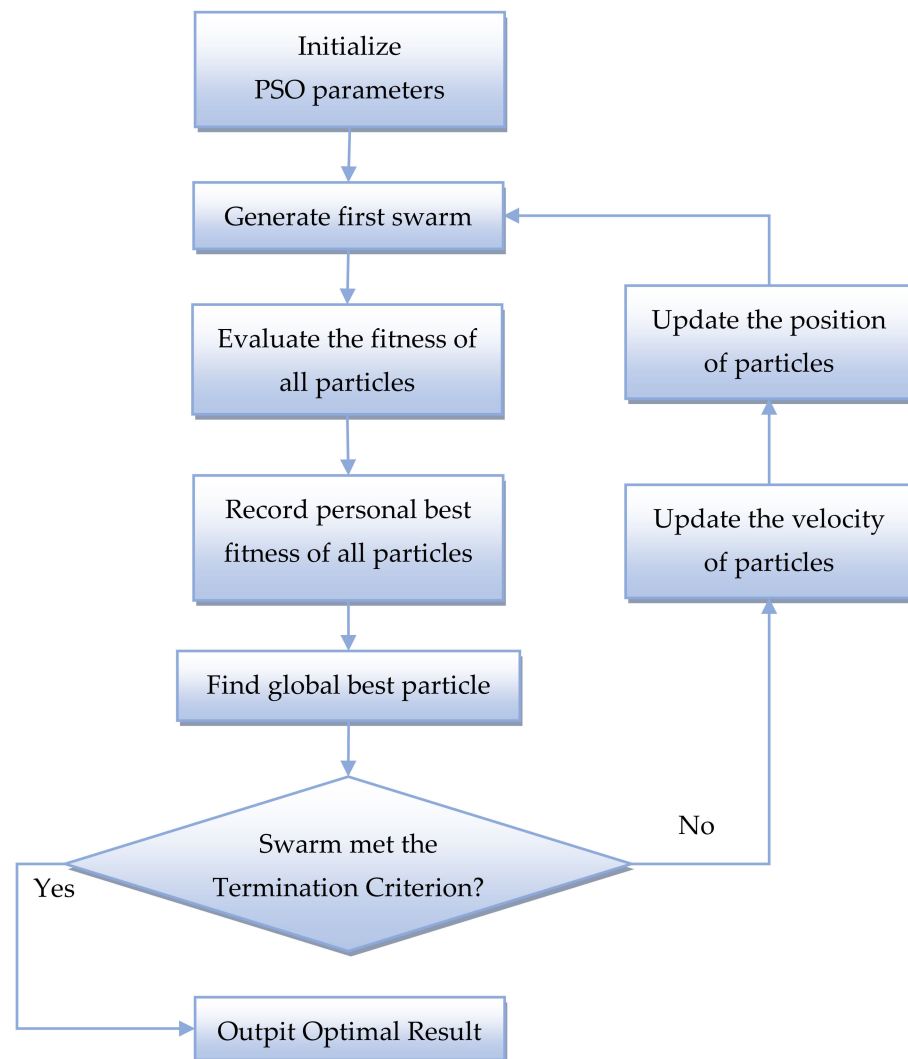


Figure 4. Basic structure of particle swarm algorithm.

3.4. Gravitational Search Algorithm

Similarly to the previous method, the gravitational search algorithm [29,30] moves the population of possible solutions; however, the direction of movement is set based on the relationship of particles according to the principles of the laws of gravity and mass interaction.

The gravitational search algorithm (Figure 5) uses two laws:

1. The law of gravitation: each particle attracts others and the force of attraction between two particles is directly proportional to the product of their masses and inversely proportional to the distance between them. Note that, unlike the Universal Law of Gravitation, the square of the distance is not used, which provides the numerical algorithm with more efficient results.
2. Law of motion: the current speed of any particle is equal to the sum of the part of the speed at the previous moment of time and the change in speed, which is equal to the force with which the system acts on the particle, divided by the inertial mass of the particle.

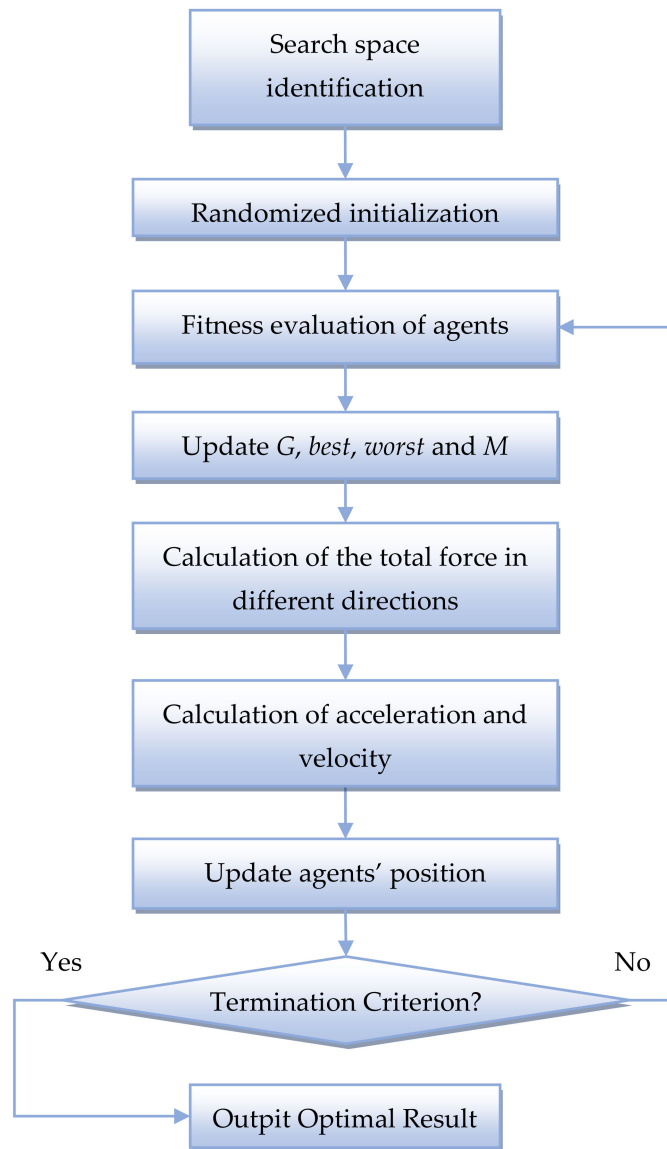


Figure 5. Basic structure of gravitational search algorithm.

The gravity search algorithm is presented in Algorithm 3.

Algorithm 3 Gravitational Search Algorithm.

Input: NNS, N is the maximum number of particles in the system, ε is small constant
Output: p^*

- 1 initialize a random of the system (a population is a set of different pairs of weighting factors)
 $S = \{p_i^1, p_i^2, \dots, p_i^3\}_{i=1}^N$, initialize ξ_i are random variables uniformly distributed from zero to one;
- 2 **for** $i = 1:N$ **do**
 calculate the value of the fitness function $f(p_i)$;
end
- 3 update the gravitational constant, best and worst particles, and masses
for $i = 1 : N$ **do**
 $M_{ai}(t) = M_{pi}(t) = M_{ii}(t) = M_i(t) = \frac{m_i(t)}{\sum_{j=1}^N m_j(t)}$;
 $m_i(t) = \frac{f(p_i) - \max_{j=1,N} f(p_j)}{\min_{j=1,N} f(p_j) - \max_{j=1,N} f(p_j)}$;
end
- 4 calculate the value of the gravitational constant :
 $G(t) = \frac{G_0}{e^{\beta t}}, \beta > 0$;
- 5 calculation of the resulting force in different directions :
for $i = 1 : N$ **do**
 for $j = 1 : N$ **do**
 if $j \neq i$ **then**
 $F_{ij}(t) = G(t) \frac{M_{pi}(t) \times M_{aj}(t)}{\|p_i, p_j\| + \varepsilon} (p_j(t) - p_i(t))$;
 $F_i(t) + = \xi_i F_{ij}(t)$
 end
 end
- 6 calculation of accelerations and speeds :
for $i = 1 : N$ **do**
 $a_i(t) = M_{ii}(t)$;
 $v_i(t+1) = (\xi_1, \dots, \xi_n)^T \times v_i(t) + \frac{F_i(t)}{a_i(t)}$;
end
- 7 updating the positions of particles (pairs of weighting factors) :
for $i = 1 : N$ **do**
 $p_i(t+1) = p_i(t) + v_i(t+1)$.
end
- 8 $p^* = \operatorname{argmin}\{f(p_i), i = 1, N\}$
- 9 **if** $\|E(p^*)\| \leq \varepsilon$ **then**
 go to step 10;
else go to step 2;
- 10 **Return** p^* .

All the optimization algorithms proposed in this study have various advantages and disadvantages presented in Table 1. Due to the fact that the investigated optimal control problem has been transformed into an equivalent problem of nonlinear optimization of a function of many variables, it is impossible to unambiguously identify the most efficient of the algorithms based on theoretical assumptions.

Table 1. Comparative analysis of optimization algorithms.

| Optimization Methods | Advantages | Disadvantages |
|-----------------------------|---|--|
| Gradient descent | + easy implementation; + applicable to any loss function. | – getting stuck in local optimum; – slow convergence. |
| Genetic algorithm | + can be used to optimize functions that do not directly depend on arguments; + does not require additional information; + high productivity. | – too large value of the mutation parameter reduces the probability of finding the exact global optimum. |
| Gravity search | + in practice, the method is more accurate than genetic algorithms; + high convergence rate. | – lengthy calculations; – low accuracy of optimization of multi-modal functions; – functions of exit from local optima are not provided. |
| Particle swarm optimization | + simplicity and the ability to quickly modify the algorithm; + high convergence rate. | – lengthy calculations; – low accuracy; – functions of exit from local optima are not provided. |

4. Computational Experiments

Earlier in our research, we showed that the application of the neural network approach to solving optimal control problems demonstrates good results for the OCP with the Lagrange function linear with respect to the control. An analysis of the application of various optimization algorithms for neural networks for solving linear OCPs showed that the evolutionary optimization algorithm uses the least number of iterations to achieve a given accuracy.

Within the framework of this study, we will consider the class of quadratic optimal control problems and analyze the behavior of the considered optimization methods in this case. In addition, within the framework of this study, an *example of the OCP for which algorithms are stuck at a local optimum* of the considered algorithms falling into the local extremum of the optimized function is given, for linear OCPs.

Example 1. *The Quadratic OCP. Consider a particular OCP with a quadratic functional with respect to a control of the form:*

$$\begin{cases} J(\omega) = \int_0^1 u^2(t)dt - x(1) \rightarrow \min, \\ \dot{x}(t) = x(t) + u(t), \\ -2 \leq u(t) \leq -0.4, \\ x(0) = 0. \end{cases} \quad (21)$$

Let us compose the Lagrange function corresponding to problem (21):

$$L(t, x, u, \lambda, \lambda_0, \mu) = \lambda_0 u^2(t) + \lambda(t)(x(t) + u(t)) + \mu_1(t)(u(t) + 0.4) + \mu_2(t)(u(t) - 2). \quad (22)$$

The corresponding optimization problem (21) has the form:

$$\min_y E(y) = \frac{1}{2} \sum_{h=1}^q \{E_1(t_h, y) + E_2(t_h, y) + E_3(t_h, y) + E_4(t_h, y)\}, \quad (23)$$

where $y = (w_x, w_\lambda, w_\mu, w_u, b_x, b_\lambda, b_\mu, b_u, v_x, v_\lambda, v_\mu, v_u)^T$ and

$$\left\{ \begin{array}{l} E_1(y) = \sum_{i=1}^q [\dot{x}^i - x^i - u^i]^2, \\ E_2(y) = \sum_{i=1}^q [\dot{\lambda}^i + \lambda^i]^2, \\ E_3(y) = \sum_{i=1}^q [\lambda^i - 2u^i + \mu_1^i - \mu_2^i]^2, \\ E_4(y) = \sum_{i=1}^q [\varphi_{FB}^\varepsilon(\mu_1^i, 2 - u^i)]^2 + \sum_{i=1}^q [\mu_2^i, -0.4 - u^i]^2, \varepsilon \rightarrow 0_+. \end{array} \right. \tag{24}$$

We define functional approximations of the neural network taking into account the boundary conditions in the form:

$$\left\{ \begin{array}{l} \bar{x} = t \cdot n_x, \\ \bar{u} = n_u, \\ \bar{\lambda} = -1 + (t - 1)n_\lambda, \\ \bar{\mu}_1 = n_{\mu_1}, \\ \bar{\mu}_2 = n_{\mu_2}. \end{array} \right. \tag{25}$$

Let us compare the results of the neural network approach of finding a solution to the OCP using various methods for optimizing weight coefficients. The initial data for the implementation of the algorithms are presented in Table 2.

Table 2. Variable algorithm parameter values.

| Algorithm | Parameters | Value |
|-----------------------------|---|-----------------------------|
| Genetic algorithm | N_t—number of split points | (10, 50, 100, 200, 400) |
| | size—number of individuals in the population | (400, 500, 600, 800, 1000) |
| | pcross—percentage of individuals not subject to crossbreeding | (0.5, 0.55, 0.6, 0.65, 0.7) |
| | pmut—percentage of individuals subject to mutation | (0.1, 0.15, 0.2, 0.25, 0.3) |
| Particle swarm optimization | N_t—number of split points | (10, 50, 100, 200, 400) |
| | Size—maximum number of particles in the system | (50, 100, 150) |
| | c0, c1—accelerations | (0.5, 1.5) |
| | w—inertia weight | 0.75 |
| Gravity search | N_t—number of split points | (10, 50, 100, 200, 400) |
| | Size—maximum number of particles in the system | (50, 100, 150) |
| | G0—gravitational constant | 100.0 |
| | Alpha—coefficient of regulation of the monotony of the gravitational constant | 20.0 |

Figure 6 shows neural network solutions to the optimal control problem (21). Figure 7 shows graphs of errors (deviations of functional approximations from the analytical solution) over the entire time interval along the phase trajectory and control, respectively. The calculated errors of the target functional of the neural network approximation, as well as the values of the additive error of the optimal pair are presented in Table 3.

Table 3. Computing characteristics of methods of optimization of neural network approximation.

| Algorithm | Count | E(y) | ΔJ(u) | $\bar{\Delta x}$ | $\bar{\Delta u}$ |
|-----------------------------|--------|-----------|-----------|------------------|------------------|
| Gradient descent | 45,981 | 0.0101829 | 0.0004729 | 0.0109726 | 0.2697622 |
| Genetic algorithm | 10,316 | 0.0347283 | 0.0039726 | 0.0286430 | 0.1952428 |
| Gravity search | 12,532 | 0.0811294 | 0.0089354 | 0.1294026 | 0.1962571 |
| Particle swarm optimization | 13,941 | 0.1952745 | 0.0298056 | 0.2197037 | 0.1681095 |

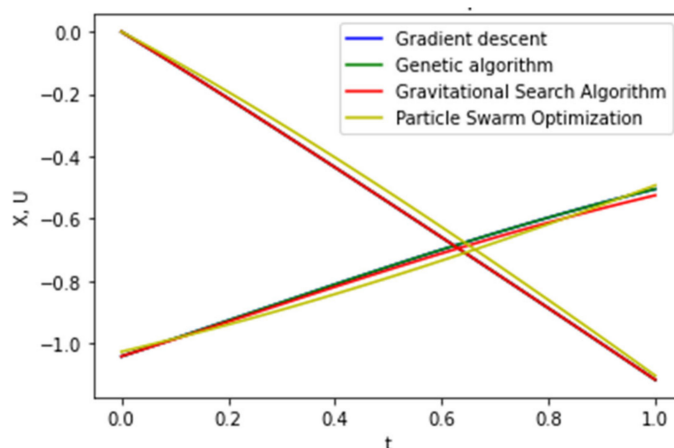


Figure 6. Neural network solution for OCP.

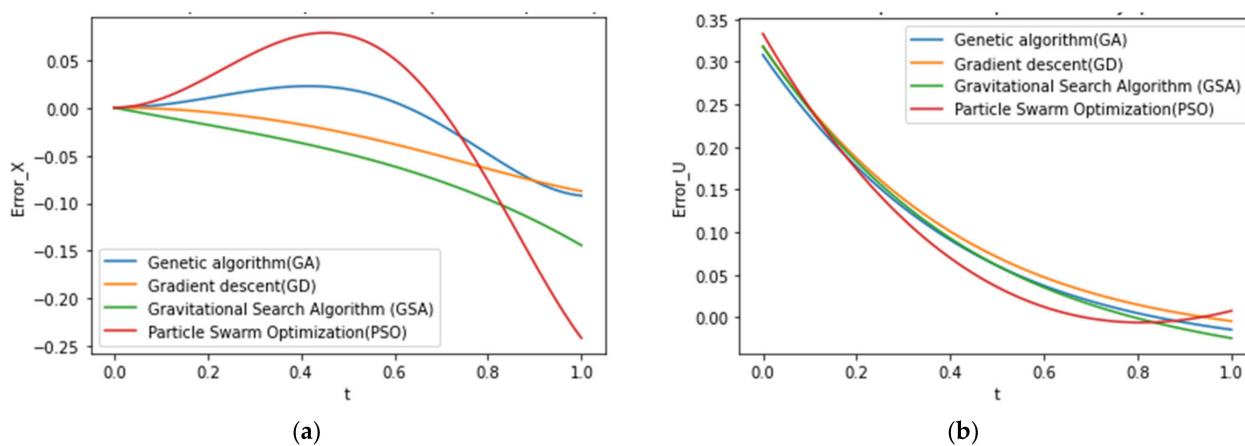


Figure 7. (a) Phase variable error; (b) Control error.

The results of the study showed that to achieve a given accuracy, the gradient descent algorithm requires the largest number of iterations $Count = 45,981$, but this method allows achieving better accuracy for the phase trajectory and functional.

The genetic algorithm ($Count = 10,316$) has the lowest convergence time among the considered methods for solving OCP (21). In this case, the accuracy of the solution $E(y)$ differs from the gradient algorithm in the second order.

The results of the evolutionary gravity search algorithm showed comparable results in terms of accuracy, but the number of iterations $Count = 12,532$ required to achieve it is greater than the genetic algorithm. The basic particle swarm algorithm did not get the specified accuracy ($E(y) = 0.1952745$) and hit the local optimum for the number of iterations $Count = 17,941$.

Note that the view of the deviation graphs of the neural network approximation for various optimization methods confirms that the gradient descent algorithm and the genetic algorithm showed better calculation accuracy. The least accurate functional approximation of the phase trajectory is constructed using the particle swarm algorithm and has an average deviation of the phase trajectory $\Delta\bar{x} \approx 0.2197037$. These function values are the best result of the approximation model approximation (25) obtained experimentally for the PSO method. The results obtained can be explained by the disadvantage of the PSO algorithm in the absence of procedures for exiting local optima, as well as the complexity of selecting the algorithm parameters.

Thus, for a given OCP, multi-agent methods require less iteration for convergence; however, the particle swarm algorithm did not achieve the specified computational accuracy and fell into a local optimum.

In addition, the multi-agent gravity search and particle swarm methods show the longest execution time for iteration, but the overall convergence rate of the algorithms is less than the gradient descent algorithm. The genetic algorithm showed the highest rate of convergence in terms of the total execution time of the algorithm (see Table 4).

Table 4. Computing machine time costs of neural network approximation optimization methods.

| Algorithm | TimeIt | Time |
|-----------------------------|-------------|-------------|
| Gradient descent | 1.582092474 | 72,746.1939 |
| Genetic algorithm | 3.195061662 | 39,350.3794 |
| Gravity search | 5.603621509 | 53,413.7201 |
| Particle swarm optimization | 5.162834727 | 46,160.9052 |

Thus, for quadratic optimal control problems, as well as for linear ones, the genetic optimization algorithm showed the fastest rate of convergence relative to the total execution time of the algorithm. However, for linear optimal control problems, the number of iterations *Count* required to achieve the specified accuracy was approximately 2 times less.

Example 2. *Stuck algorithms at the local optimum.*

In order to check how an increase in the dimension of the optimal control problem affects the convergence and accuracy of the resulting solution, we investigate the following problem. Consider a two-dimensional problem with respect to the trajectory of optimal control with mixed constraints of the form:

$$\left\{ \begin{array}{l} J(\omega) = \int_0^3 2x_1(t)dt \rightarrow \min \\ \dot{x}_1(t) = x_2(t), \\ \dot{x}_2(t) = u(t), \\ |u(t)| \leq 2, \\ x_1(t) \geq -7, \\ x_1(0) = 2, x(0) = 0. \end{array} \right. \quad (26)$$

The corresponding optimization problem (26) has the form:

$$\min_y E(y) = \frac{1}{2} \sum_{h=1}^q \{ E_1(t_h, y) + E_2(t_h, y) + E_3(t_h, y) + E_4(t_h, y) \}, \quad (27)$$

where $y = (w_x, w_\lambda, w_\mu, w_u, b_x, b_\lambda, b_\mu, b_u, v_x, v_\lambda, v_\mu, v_u)^T$ and

$$\left\{ \begin{array}{l} E_1(y) = \sum_{i=1}^q [x_1^i - x_2^i]^2 + \sum_{i=1}^q [x_2^i - u^i]^2, \\ E_2(y) = \sum_{i=1}^q [\lambda_1^i + 2 - \mu_3^i]^2 + \sum_{i=1}^q [\lambda_2^i + \lambda_1^i]^2, \\ E_3(y) = \sum_{i=1}^q [\lambda_2^i + \mu_1^i - \mu_2^i]^2, \\ E_4(y) = \sum_{i=1}^q [\varphi_{FB}^\varepsilon(\mu_1^i, 2 - u^i)]^2 + \sum_{i=1}^q [\mu_2^i, -2 - u^i]^2 + \sum_{i=1}^q [\mu_3^i, 7 + x_1^i]^2, \varepsilon \rightarrow 0_+. \end{array} \right. \quad (28)$$

Figure 8 shows neural network solutions obtained using various optimization methods. The calculated values of the additive error of the optimal pair of neural network approximation are presented in Table 5.

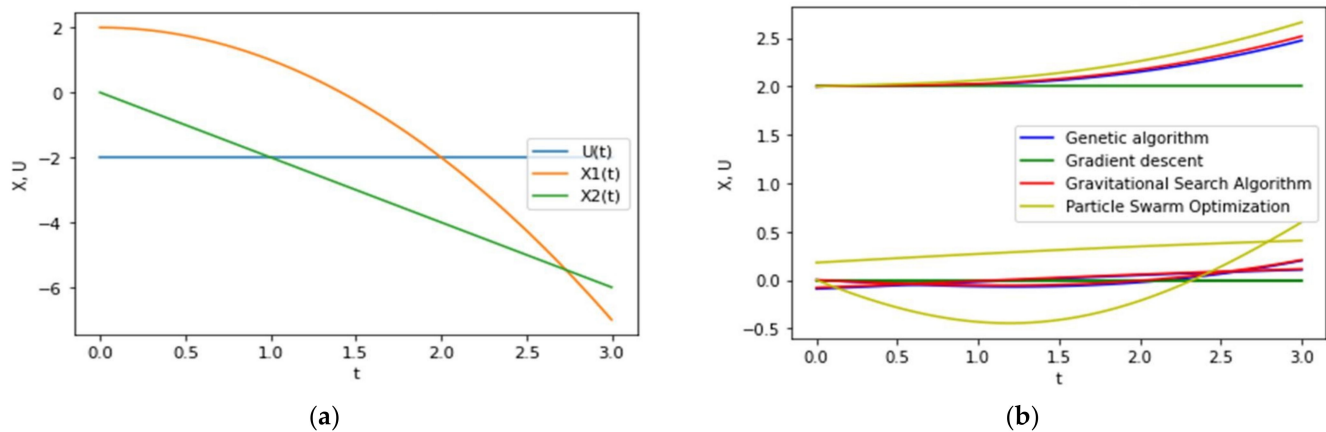


Figure 8. (a) Analytical solution; (b) Neural network solution.

Table 5. Computing characteristics of methods of optimization of neural network approximation.

| Algorithm | $\bar{\Delta x}_1$ | $\bar{\Delta x}_2$ | $\bar{\Delta u}$ |
|-----------------------------|--------------------|--------------------|------------------|
| Gradient descent | 202.07342431 | 304.51482502 | 302.32155329 |
| Genetic algorithm | 203.26250408 | 317.43769457 | 302.27943816 |
| Gravity search | 215.19872669 | 319.05872786 | 303.42128037 |
| Particle swarm optimization | 232.72998470 | 325.55120591 | 286.98811501 |

An increase in the dimensionality in this OCP along the phase trajectory and conjugate variables led to the fact that the considered evolutionary algorithms, as well as the gradient descent algorithm, fell into the local optimum of the minimized function.

5. Conclusions and Future Work

An analysis of the application of various optimization algorithms for neural network solution of optimal control problems showed that evolutionary function optimization algorithms use the least number of iterations to achieve a given accuracy, but finding the exact global minimum is difficult and requires significant computing resources;

Multi-agent methods of gravity search and swarm of particles show the longest execution time for iteration. For quadratic optimal control problems, as well as for linear ones, the genetic optimization algorithm showed the fastest rate of convergence relative to the total execution time of the algorithm. However, for linear optimal control problems, the number of iterations required to achieve the specified accuracy was approximately two times less.

Among evolutionary algorithms, the basic algorithm for optimizing a swarm of particles turned out to be less resistant to hitting a local optimum. The results obtained can be explained by the disadvantage of the PSO algorithm in the absence of procedures for exiting local optima, as well as by the complexity of the selection of the algorithm parameters. In addition, the algorithm of gravitational search and swarm of particles, when implemented, requires lengthy computations of iteration due to the need to recalculate many parameters.

Following from this research, it is planned to study a wider range of multi-agent optimization methods and genetic algorithms, their various modifications, and also to apply the results obtained in the framework of solving applied problems in various scientific fields.

Author Contributions: I.B. supervised the scientific methodology of neural network modeling and the setting of experiments; L.Z. wrote the software and implemented scientific experiments on the approximation of neural network approximation. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Russian Federal Property Fund in the framework of scientific project No. 20-07-01065, as well as a grant from the President of the Russian Federation for state support of leading scientific schools of the Russian Federation (NSh-2502.2020.9).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Rizaner, F.B.; Rizaner, A. Approximate Solutions of Initial Value Problems for Ordinary Differential Equations Using Radial Basis Function Networks. *Neural Process. Lett.* **2017**, *48*, 1063–1071. [[CrossRef](#)]
2. Vasiliev, A.N.; Tarkhov, D.A. Neural networks as a new universal approach to numerical solution of mathematical physics problems. *Neurocomput. Dev. Appl. Radiotech.* **2004**, *7*, 111–118.
3. Nazemi, A.; Karami, R. A Neural Network Approach for Solving Optimal Control Problems with Inequality Constraints and Some Applications. *Neural Process. Lett.* **2017**, *45*, 995–1023. [[CrossRef](#)]
4. Effati, S.; Pakdaman, M. Optimal control problem via neural networks. *Neural Comput. Appl.* **2013**, *23*, 2093–2100. [[CrossRef](#)]
5. Ruthotto, L.; Osher, S.J.; Li, W.; Nurbekyan, L.; Fung, S.W. A machine learning framework for solving high-dimensional mean field game and mean field control problems. *Proc. Natl. Acad. Sci. USA* **2020**, *117*, 9183–9193. [[CrossRef](#)] [[PubMed](#)]
6. Ghasemi, S.; Nazemi, A.R. A Neural Network Method Based on Mittag-Leffler Function for Solving a Class of Fractional Optimal Control Problems. *AUT J. Model. Simul.* **2018**, *50*, 211–218. [[CrossRef](#)]
7. Yadav, N.; Yadav, A.; Kumar, M. *An Introduction to Neural Network Methods for Differential Equations*; Springer: Berlin, Germany, 2015; Volume 1, p. 114.
8. Kmet, T.; Kmetova, M. *Neural Networks Solving Free Final Time Optimal Control Problem*; Springer: Berlin, Germany, 2012; Volume 7505, pp. 174–186.
9. Kheyrinataj, F.; Nazemi, A. Fractional power series neural network for solving delay fractional optimal control problems. *Connect. Sci.* **2019**, *53*, 53–80. [[CrossRef](#)]
10. Irigoyen, E.; Galvan, J.B.; Perez-Illarbe, M.J. Neural networks for constrained optimal control of nonlinear systems. In Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN 2000), Como, Italy, 27 July 2000; Volume 4, pp. 299–304. [[CrossRef](#)]
11. Liangyu, M.; Yinping, G. Superheated steam temperature predictive optimal control based on external time-delay BP neural network and a simpler PSO algorithm. In Proceedings of the 31st Chinese Control Conference, Hefei, China, 25–27 July 2012; pp. 4050–4055.
12. Li, H.; Zhang, D.-J.; Lin, J.; Li, W. Application based on neural network of the boiler optimal control in power station. In Proceedings of the International Conference on Machine Learning and Cybernetics, Guangzhou, China, 18–21 August 2005; Volume 2, pp. 1192–1195. [[CrossRef](#)]
13. Li, D.; Tian, X.; Ji, Z. A new numerical approach based on piecewise interpolation for solving a class of fractional optimal control problems. In Proceedings of the 35th Chinese Control Conference (CCC), Chengdu, China, 27–29 July 2016; pp. 10442–10446. [[CrossRef](#)]
14. Gaiduk, A.R.; Vershinin, Y.A.; West, M.J. Neural networks and optimization problems. In Proceedings of the International Conference on Control Applications, Glasgow, UK, 18–20 September 2002; 2002; Volume 1, pp. 37–41. [[CrossRef](#)]
15. Supeng, Z.; Wenxing, F.; Jun, Y.; Jianjun, L. Applying RBF neural network to missile control system parameter optimization. In Proceedings of the 2nd International Asia Conference on Informatics in Control, Automation and Robotics (CAR 2010), Wuhan, China, 6–7 March 2010; pp. 337–340. [[CrossRef](#)]
16. Yildiz, A.R. Comparison of evolutionary-based optimization algorithms for structural design optimization. *Eng. Appl. Artif. Intell.* **2013**, *26*, 327–333. [[CrossRef](#)]
17. Tautou, L.; Pierreval, H. Using evolutionary algorithms and simulation for the optimization of manufacturing systems. In Proceedings of the INRIA/IEEE Symposium on Emerging Technologies and Factory Automation. ETFA'95, Paris, France, 10–13 October 1995; Volume 3, pp. 509–516. [[CrossRef](#)]
18. Narukawa, K.; Rodemann, T. Examining the Performance of Evolutionary Many-Objective Optimization Algorithms on a Real-World Application. In Proceedings of the Sixth International Conference on Genetic and Evolutionary Computing, Kitakyushu, Japan, 25–28 August 2012; pp. 316–319. [[CrossRef](#)]
19. Vikhar, P.A. Evolutionary algorithms: A critical review and its future prospects. In Proceedings of the International Conference on Global Trends in Signal. Processing, Information Computing and Communication, Jalgaon, India, 22–24 December 2016; pp. 261–265. [[CrossRef](#)]
20. Mahammed, N.; Benslimane, S.M.; Ouldkradda, A.; Fahsi, M. Evolutionary Business Process Optimization using a Multiple-Criteria Decision Analysis method. In Proceedings of the International Conference on Computer, Information and Telecommunication Systems (CITS), Colmar, France, 11–13 July 2018; pp. 1–5. [[CrossRef](#)]

21. Gu, F.; Li, K.; Liu, Y. A Hybrid Evolutionary Algorithm for Solving Function Optimization Problems. In Proceedings of the 12th International Conference on Computational Intelligence and Security (CIS), Wuxi, China, 16–19 December 2016; pp. 526–529. [[CrossRef](#)]
22. Wen, L.; Ximing, L. An effective hybrid evolutionary algorithm for constrained engineering optimization. In Proceedings of the Advanced Information Technology, Electronic and Automation Control. Conference, Chongqing, China, 19–20 December 2015; pp. 930–933. [[CrossRef](#)]
23. Xu, Q.; Xu, Z.; Ma, T. A Short Survey and Challenges for Multiobjective Evolutionary Algorithms Based on Decomposition. In Proceedings of the International Conference on Computer, Information and Telecommunication Systems (CITS), Beijing, China, 28–31 August 2019; pp. 1–5. [[CrossRef](#)]
24. Yadav, N.; Yadav, V.; Verma, P. Role of Evolutionary algorithms in Software Reliability Optimization. In Proceedings of the International Conference System Modeling & Advancement in Research Trends (SMART), Moradabad, India, 25–27 November 2016; pp. 45–48. [[CrossRef](#)]
25. He, Z.; Yen, G.G.; Lv, J. Evolutionary Multiobjective Optimization with Robustness Enhancement. *Proc. IEEE Trans. Evol. Comput.* **2020**, *24*, 494–507. [[CrossRef](#)]
26. Sharma, J.; Singhal, R.S. Comparative research on genetic algorithm, particle swarm optimization and hybrid GA-PSO. In Proceedings of the 2nd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 11–13 March 2015; pp. 110–114.
27. Li, T.; Lai, X.; Wu, M. An Improved Two-Swarm Based Particle Swarm Optimization Algorithm. In Proceedings of the 6th World Congress on Intelligent Control. and Automation, Dalian, China, 21–23 June 2006; pp. 3129–3133. [[CrossRef](#)]
28. Yitong, L.; Mengyin, F.; Hongbin, G. A Modified Particle Swarm Optimization Algorithm. In Proceedings of the Chinese Control Conference, Zhangjiajie, China, 26–31 July 2007; pp. 479–483. [[CrossRef](#)]
29. Sheikhpour, S.; Sabouri, M.; Zahiri, S. A hybrid Gravitational search algorithm—Genetic algorithm for neural network training. In Proceedings of the 21st Iranian Conference on Electrical Engineering (ICEE), Mashhad, Iran, 14–16 May 2013; pp. 1–5. [[CrossRef](#)]
30. Li, W. An Improved Gravitational Search Algorithm for Optimization Problems. In Proceedings of the Chinese Control and Decision Conference (CCDC), Nanchang, China, 3–5 June 2019; pp. 2605–2608. [[CrossRef](#)]