Int j simul model 23 (2024) 2, 227-238
Original scientific paper

USING ADAPTIVE NEURAL NETWORKS FOR OPTIMISING DISCRETE EVENT SIMULATION

Raska, P.[*]; Ulrych, Z.[**]; Baloun, J.[***]; Malaga, M.[*] & Lenc, L.[***]

[*] Department of Industrial Engineering – FME, University of West Bohemia, Univerzitní 22, Pilsen 306 14, Czech Republic

[**] Department of Computing and Didactic Technology, Faculty of Education, University of West Bohemia, Veleslavínova 42, 301 00 Pilsen, Czech Republic

[***] Department of Computer Science and Engineering, Faculty of Applied Sciences, University of West Bohemia, Technická 8, 301 00 Pilsen, Czech Republic

E-Mail: praska@fst.zcu.cz, ulrychz@kvd.zcu.cz, balounj@kiv.zcu.cz, malaga@fst.zcu.cz, llenc@kiv.zcu.cz

Abstract
The paper presents the use of adaptive neural networks for carrying out simulation optimisation using digital models (discrete event simulation models) created in accordance with the Industry 4.0 concept. The digital models reflect different problems in industrial engineering. The simulation optimisers use an adaptive neural network to find the best settings of the digital models according to defined objective functions for each model. We compared the effectiveness (using different evaluation criteria) of the adaptive neural network (ANN) optimisation method used on 6 different discrete event simulation models. We compared adaptive neural networks with 11 optimisation methods – pseudo gradient, metaheuristic, evolutionary and swarm optimisation methods (and their combinations). The ANN method demonstrated the ability to efficiently find the global optimum of the objective function in different cases of the objective function – the ANN method is in the top 5 best tested methods from the 12 optimisation methods.
(Received in November 2023, accepted in April 2024. This paper was with the authors 3 weeks for 2 revisions.)

Key Words: Adaptive Neural Network, Comparison of Optimisation Methods, Discrete-Event Simulation Models, Simulation Optimisation

1. INTRODUCTION

Today's turbulent times are increasingly demonstrating that humanity itself is in many cases dependent on the use of information and production technologies. The recent global socio-economic crisis caused by the coronavirus is a clear example of this. Labour shortages have adversely affected production in many companies. The impacts are further exacerbated by other problems such as the scarcity of imported natural resources used in manufacturing.

One way to mitigate these effects of the crisis is to implement the core of Industry 4.0. This approach changes strategies, organisation, business models, value and supply chains, processes, products, skills, and stakeholder relationships [1].

The shared idea of the Industry 4.0 concept is that individual physical elements and processes are completely captured in the digital world and are able to communicate and interact with each other [2, 3]. In such a cyber-physical system, we could not only control these elements by means of commands (a certain degree of autonomous control is also a prerequisite for these elements), but also test different options and find out what benefits a given option has for the enterprise – i.e., to optimise it. It is advisable to use simulation and link it to the optimisation – i.e. use simulation optimisation.

Simulation optimisation aims at determining the best values of the input parameters, while the analytical objective function and constraints are not explicitly known in terms of design variables and their values can only be estimated by complicated analysis or time-consuming simulation [4].

It is impossible to test all the solutions due to it being an NP-hard modelled problem. Moreover, running a simulation model to test individual settings of the decision variables is time-consuming and thus costly.

'In addition, the revolution of the artificial intelligence era has led to the recent development of intelligent optimisation techniques that are able to comfortably provide near-optimal solutions to hard and complex real-world optimisation problems, which would not have been practicable using the traditional or exact optimisation methods.' [5].

We modified and tested an adaptive neural network (ANN). We used the ANN to find suitable settings for the input parameters of discrete event simulation models reflecting different problems in industrial engineering.

The use of neural networks (NNs) is typical for facial recognition, stock market prediction, social media etc., but the use of NNs is not so common for discrete event simulation optimisation.

A hybrid approach for simulation optimisation of a pressure vessel design problem is presented in [4]. An adaptive neural network was proposed as a local and global metamodel-based optimisation method in [6]. It combines genetic algorithms and neural networks to predict the fitness function. Architectures combining machine learning and discrete event simulation for determining the route of a robot are presented in [7]. They use reinforcement learning and require the dynamics of the studied system to be known in sufficient detail, which is not applicable to all the problems. Generally, NNs are considered as global metamodeling methods and have received only minor attention [8], which is unfortunate considering the results in other areas.

2. GLOBAL OPTIMISATION

There is a wide class of optimisation techniques for solving industrial engineering optimisation problems in different levels [9, 10]. Discrete event simulation models where state variables change only at a discrete set of points in time usually have many different possible solutions – solution candidates – which cannot all be evaluated as they are NP-hard problems. The main problem is how to find the best solution (or near-optimal solution) in the big search space as follows:

$$\breve{\mathbf{X}} = \min_{\mathbf{X}\in\tilde{X}} F(\mathbf{X}) = \left\{\breve{\mathbf{X}} \in \tilde{X} : F\left(\breve{\mathbf{X}}\right) \leq F(\mathbf{X}) \forall \mathbf{X} \in \tilde{X}\right\} \quad (1)$$

where $\breve{\mathbf{X}}$ denotes the global minimum of the objective function (or set of functions, where the compound function represents the goal of the simulation study); $F(\mathbf{X})$ denotes the objective function value of the possible solution – solution candidate (the range usually includes real numbers $F(\mathbf{X}) \subseteq \mathbb{R}$, and the objective function maximisation can be converted to function minimisation or vice versa); $\tilde{X}$ denotes the search space.

Each solution candidate in the search space is represented as the vector of the values for each decision variable as follows:

$$\mathbf{X} = [x_1, x_2, \dots, x_n] \quad (2)$$

where x_1 denotes the value of the first decision variable – simulation model input parameter.

The search space (especially for discrete simulation models) is usually a boundary constrained problem defined by an upper and lower bound for each decision variable. Finding the global minimum of an objective function is much more difficult because analytical methods are often inapplicable and using numerical methods leads to complication of the problem or is often inefficient. Global optimisation focuses on finding the global minimum or maximum in a defined search space, as opposed to finding local minima or maxima in the case of local optimisation, which tends to be simpler. The local minimum of an objective function is an element of the search space where $F: \tilde{X} \mapsto \mathbb{R} \wedge \tilde{X} \subseteq \mathbb{R}$:

$$\breve{\mathbf{X}}_l: \exists e > 0: F(\breve{\mathbf{X}}_l) \leq F(\mathbf{X}) \forall \mathbf{X} \in \tilde{X}, |\mathbf{X} - \breve{\mathbf{X}}_l| < e \tag{3}$$

where $\breve{\mathbf{X}}_l$ denotes the local minimum (for all $\mathbf{X}$ neighbouring $\breve{\mathbf{X}}_l$); e denotes the boundary around the local minimum; $F(\breve{\mathbf{X}}_l)$ denotes the objective function value of the local minimum; the meanings of $\mathbf{X}$, $\tilde{X}$ are described in the previous Eq. (3).

Fig. 1 shows the general principle of solving the simulation optimisation problem represented by the digital model built in commonly available simulation software (Arena and Tecnomatix PlantSimulation). The digital model has its own specified main objective function (mainly consisting of several partial objective functions) and its value is calculated based on the results of the simulation run with the model. The optimisation method – Adaptive Neural Network – **ANN** – implemented in the simulation optimiser changes the settings of the decision variables of the model.

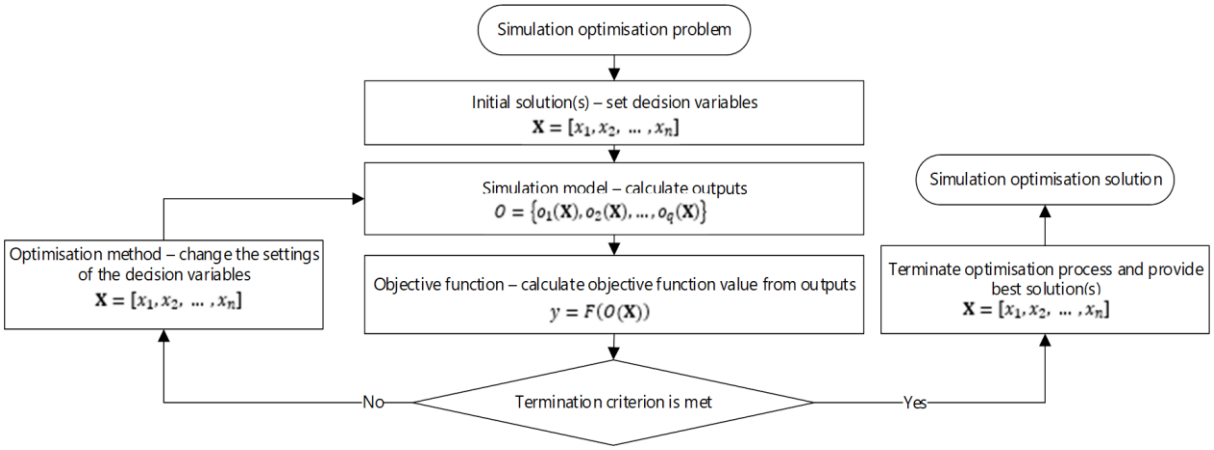

Figure 1: Simulation optimisation problem.

We used the Server-Client architecture to manage the testing of different settings of the ANN. This architecture allows use of the external database of the simulation experiments and reduction of the amount of time taken for testing different settings of the ANN. Settings of the ANN optimisation method parameters affect the progress of searching for the global optimum of the simulation model's objective function. Optimisation methods often work with individual elements of the search space (in the form of vectors with values of all the decision variables – see Eq. (2)). It is often necessary to access the individual values of these decision variables (e.g., transforming the values of the decision variables of the solution candidates to create new solution candidates; for the termination criterion, etc.). The following notation uses square brackets to make the notation more readable (instead of listing individual subscripts separated by commas). Another reason for using this notation is to synchronise the standards used in algorithmizing and programming where a possible solution, i.e., an element (of the space of all solutions), is represented by a list where the values of the decision variables of the element are indexed according to the order of the individual decision variables from index 0 to the index number of axes – 1 (the decision variables represent axes in the n-dimensional search space):

$$x_j = \mathbf{X}[j] \forall j: j = \{0, 1, 2, \ldots, n-1\} \tag{4}$$

where x_j denotes the value of the j^{th} decision variable of the solution candidate $\mathbf{X}$; j denotes the index of the decision variable; n denotes the dimension of the search space containing all possible solutions.

We have modified the optimisation methods to improve their efficiency in finding the global optimum (implementing the principles of mutation and generation) and to use them for discrete event simulation optimisation purposes. The selected optimisation methods have been applied to industrial practice problems modelled by discrete-event simulation models.

3. EVALUATION CRITERIA

Many research papers are focused on testing and comparing the effectiveness of optimisation methods at finding the optimum for commonly used testing functions (Testing Benchmark), e.g., see [11-13]. Mean, standard deviation, deviation from optimum, etc. are commonly used to evaluate the performance of optimisation methods. These evaluation criteria are sufficient for commonly used testing functions where the global optimum is known and where no additional simulation software is used (the computational time taken to evaluate the objective function is negligible). These papers also use recommended settings that may not be appropriate for a different simulated optimisation problem. It is often necessary to perform simulation runs on additional simulation software when discrete simulation optimisation is done. Such simulation runs may take a long time. Therefore, it is advisable to define additional evaluation criteria for these optimisation runs.

The box plot characteristics are calculated for each series. A series consists of replicated optimisation experiments performed with a specific setting of the optimisation method parameters. These evaluation criteria are normalised in a closed interval from 0 to 1 and can be divided into basic areas concerning the following criteria:

The success of finding the optimum $f_{1\,i}$ – the criterion represents the percentage of times the global optimum or candidate solution, whose objective function value is less than a defined value from the objective function value of the global optimum of the objective function:

$$|F(\mathbf{X}_i) - F(\mathbf{X}^*)| \leq \varepsilon \tag{5}$$

The difference between the optimum and sub-optimum $f_{2\,i}$ – this criterion evaluates the difference between the objective function value of the found best solution in the series and the optimum of the objective function value (the aim is to minimise all following criterion):

$$f_{2\,i} = \left(\frac{F(\mathbf{X}^*) - F(\mathbf{X}^*{}_i)}{\Delta F_{\tilde{X}}}\right) \forall i: i = \{1, 2, \dots, s\}, f_{2\,i} \in [0,1] \tag{6}$$

The distances of quartiles – the distance between the quartiles of a concrete series. If criterion $f_{1\,i} = 0$ then criterion $f_{3\,i} = 0$ (in each optimisation experiment performed in the series, an optimum was found). If the objective function is minimised, the third criterion can be formulated as follows:

$$f_{3\,i} = \frac{f_{3w1\,i} + f_{3w2_i} + f_{3w3\,i} + f_{3w4\,i} + f_{3w5\,i}}{\Delta F_{\tilde{X}}}, \forall i: i = \{1, 2, \dots, s\}, f_{3\,i} \in [0,1] \tag{7}$$

$$\begin{gathered} f_{3w5_i} = |F(\mathbf{X}^*{}_i) - F(\mathbf{X}^*)|, f_{3w4_i} = w_{4_{f_3}} |F(\mathbf{X}^*{}_i) - Q_{1_i}|, f_{3w3\,i} = w_{3_{f_3}} |Q_{1_i} - Q_{2_i}|, \\ f_{3w2\,i} = w_{2_{f_3}} |Q_{2_i} - Q_{3_i}|, f_{3w1\,i} = w_{1_{f_3}} |Q_{3_i} - F(\mathbf{X}_{Worst_i})| \end{gathered} \tag{8}$$

The number of simulation experiments until the suboptimum was found – evaluates the number of performed simulation experiments until the best solution candidate was found in each series:

$$f_{4\,i} = \frac{f_{4w1\,i} + f_{4w2_i} + f_{4w3\,i} + f_{4w4\,i} + f_{4w5\,i}}{\tilde{X}_H}, \forall i: i = \{1, 2, \dots, s\}, f_{4_i} \in [0,1] \tag{9}$$

$$\begin{gathered} f_{4w5_i} = |min_{SE_i} - 1|, f_{4w4_i} = w_{4_{f_4}} |min_{SE_i} - Q_{1_i}|, f_{4w3\,i} = w_{3_{f_4}} |Q_{1_i} - Q_{2_i}|, f_{4w2\,i} = w_{2_{f_4}} |Q_{2_i} - Q_{3_i}|, \\ f_{4w1\,i} = w_{1_{f_4}} |Q_{3_i} - max_{SE_i}| \end{gathered} \tag{10}$$

The convergence to the optimum – represents the evolution of the values of the objective function of the solutions generated towards the desired objective function value to be achieved in the optimisation experiment. If the objective function is minimised the third criterion can be formulated as follows:

$$f_{5\,i} = \frac{f_{5w1\,i} + f_{5w2_i} + f_{5w3\,i} + f_{5w4\,i} + f_{5w5\,i}}{\Delta F_{\tilde{X}}}, \forall i: i = \{1, 2, \dots, s\}, f_{5\,i} \in [0,1] \tag{11}$$

$$f_{5w5_i} = |F(\mathbf{X}^*_i) - F(\mathbf{X}^*)|, f_{5w4_i} = w_{4_{f_5}}|F(\mathbf{X}^*_i) - Q_{1_i}|, f_{5w3\ i} = w_{3_{f_5}}|Q_{1_i} - Q_{2_i}|,$$
$$f_{5w2\ i} = w_{2_{f_5}}|Q_{2_i} - Q_{3_i}|, f_{5w1\ i} = w_{1_{f_5}}|Q_{3_i} - F(\mathbf{X}_{Worst_i})| \quad (12)$$

Table I: Evaluation criteria parameters [14].

Parameter	Parameter meaning
s	Number of the performed series
$F(\mathbf{X}_i)$	Objective function of the found solution candidate of the i^{th} optimisation experiment
$F(\mathbf{X}^*)$	Objective function value of the global optimum
$F(\mathbf{X}^*_i)$	Objective function value of the best solution candidate found in i^{th} series
$\Delta F_{\tilde{X}}$	Difference between the objective function value of the found best and worst candidate solutions of the search space in all series
$\varepsilon = 0.001$	Tolerated deviation
$F(\mathbf{X}_{Worst_i})$	Worst found possible solution found in the i^{th} series
$w_{k_{f_j}}$	Weight of relevant difference between boxplot characteristics of the j^{th} evaluation criterion (k = 4 ... difference between the best solution candidate and the lower quartile Q_1; k = 3 ... difference between the lower quartile Q_1 and median Q_2; k = 2 ... difference between the median Q_2 and the upper quartile Q_3; k = 1 ... difference between the upper quartile Q_2 and the worst found possible solution)
min_{SE_i}	Minimum number of simulation experiments that the optimisation method performed in the optimisation experiment to find the best solution candidate of the i^{th} series
max_{SE_i}	Maximum number of simulation experiments that the optimisation method performed in the i^{th} series
$\tilde{X}_H$	Maximum number of simulation experiments that the optimisation method can perform (using the entropy termination criterion)

Weighted sum – is used for prioritising some criteria over others by setting the weights of individual criteria where the sum of the weights equals one:

$$w_{f_j} \in [0,1] \forall j: j = \{1, 2, \dots, 5\}, \sum_{j=1}^{5} w_{f_j} = 1 \quad (13)$$

where w_{f_j} denotes the weight of the j^{th} criterion – the Saaty method was used to set the individual criteria weights – see Eq. (14).

$$w_{f_1} = 0.22, w_{f_2} = 0.56, w_{f_3} = 0.12, w_{f_4} = 0.06, w_{f_5} = 0.04 \quad (14)$$

The resulting evaluation function is the sum of the individual evaluation criteria multiplied by the corresponding weights for the i^{th} series (minimising the value representing the negative aspects of the algorithm's behaviour) – see Eq. (15).

$$f_i = w_{f_1} \cdot (1 - f_{1\ i}) + \sum_{j=2}^{5} f_{j\ i} \cdot w_{f_j} \forall i: i = \{1,2, \dots, s\} \quad (15)$$

where f_i denotes the weighted sum of specified criteria of the i^{th} series (criterion minimisation); $f_{j\ i}$ denotes the standardised scalar value of the j^{th} criterion of i^{th} series; w_{f_j} denotes the weight of the j^{th} criterion of the i^{th} series; parameters i, s have the same nature as in the previous equation – see Table I.

More detailed information on the evaluation methodology can be found in [14].

4. SIMULATION MODELS

Simulation models used in Industry 4.0 represent real elements of a modelled system (entities, processes, etc.) – they are also known as digital twins. We selected different digital models –

to compare the effectiveness of ANNs compared to other commonly used optimisation methods in discrete event simulation optimisation. The discrete event simulation models were built in the simulation software Arena (by Rockwell Software) and Tecnomatix PlantSimulation (by Siemens). These models represent different optimisation problems in the field of industrial engineering in the industrial companies for which they were created (the models had to be slightly modified to protect trade secrets).

We tested the following models:

- **Assembly line model** – represents the production line for two different manufacturing processes including the relative error rates at each workstation. The aim is to maximise the number of defect-free products produced.
- **Penalty model** – represents a production workshop where two types of products are produced. The objective is to produce a defined quantity of each product type in a defined time.
- **Manufacturing system and logistics model** – the model captures the overall internal logistics in the production hall and warehouse. The goal is to maximise the utilisation of all assembly lines and all tow tractors.
- **Transport model** – the model represents the transport from the warehouse to the different production lines using tow tractors. The goal is to find the best sequence of loading points for the production lines and to determine the individual loading points for each tow tractor.
- **Production and control stations model** – the model represents a production workshop consisting of different types of workplaces. The goal of the simulation optimisation is to determine the number of machines and inspectors at each workstation so that forklifts, machines, and inspectors are utilised as much as possible while the production is maximised.
- **AGV transport model** – the model represents the supply of individual assembly lines by automatically guided vehicles (AGVs) - tow tractors with trailers. The goal is to maximise the average utilisation of all assembly lines, which is superior to the average utilisation of all types of AGVs.

A more detailed description of the discrete simulation models can be found in [15].

5. PROPOSED ADAPTIVE NEURAL NETWORK

The tested methods have been selected based on directly integrated simulation optimisers for commonly used simulation software, other stand-alone applications that are used for a discrete-event simulation optimisation and the representation of optimisation methods in a systematic search for industrial engineering problems – see [5, 16, 17].

We have developed our own applications for managing the parallel simulation optimisers (using the proposed adaptive neural network) to analyse and evaluate the behaviour of each optimisation method compared to various criteria of simulation optimisation efficiency. This application allows us to: set parameters of optimisation methods and analyse their behaviour under different settings; statistically evaluate the behaviour of optimisation methods based on different aspects; select different search strategies for the methods; use parallel simulation optimisation; use a database with the results of optimisation experiments already performed (this approach is particularly effective in parallel simulation optimisation); specify the termination criteria of the optimisation experiment; identify which method was used and how successful it was (methods are switched during the optimisation process, e.g. due to competing characteristics or the use of artificial intelligence); implement behavioural modifications of the methods due to software closure etc.

Testing all possible solutions to find a suitable solution candidate or the best solution candidate of the search space (global optimum of the objective function of a modelled problem) is a very inefficient way and mostly impossible due to it being an NP-hard problem. Many of

the optimisation algorithms (especially naturally inspired algorithms) generate the whole population containing a big number of these solution candidates which are iteratively refined as follows:

$$\mathbf{X}_i = X_{\mathrm{Pop}}[i] \forall i: i = \{0, 1, 2, \dots, m-1\} \tag{16}$$

where $\mathbf{X}_i$ denotes the i^{th} generated solution candidate; X_{Pop} denotes the list – population of generated solution candidates; m denotes the length of the list X_{Pop} – population size.

Ideally, more candidates are generated in more promising areas of the search space. These areas can be identified based on the objective function $F(X)$. However, the objective function is unknown. Therefore, we use gathered candidates and a multilayer perceptron (MLP) to approximate the objective function, since MLPs are good function approximators [18]. We further utilise the approximation of the objective function $\hat{F}(X)$ to generate new candidates.

The architecture of the network consists of input, output and three hidden layers. It is inspired by [19], where a three-hidden-layer neural network with super approximation power is introduced. But even single hidden layer neural networks are proven to be universal approximators [18], therefore we consider three hidden layers as adequate. The input layer accepts the candidate (vector X_i). The output layer is a single neuron with a linear activation function to predict $\hat{F}(X_i)$. The hidden layers are fully connected layers with a leaky ReLU activation function. The number of neurons is given by the *noNeurons* parameter. In the first, second and third layer there are *noNeurons*, $2 \times$ *noNeurons* and *noNeurons* neurons respectively.

ALGORITHM 1: ANN optimisation pseudocode

```
1   begin
2       X_Pop ← CreateInitialPopulation();
3       X_NewPop ← X_Pop;
4       // the best solutions candidate in X_Pop according to objective function F(X)
        X_Best ← Min(X_Pop,F(X_Pop));
5       while not TerminationCriterion( ) do begin
6           // train MLP to approximate objective function
            MLP ← Train(MLP, X_Pop, F(X_Pop) );
7           X_LastPop ← X_NewPop;
8           X_NewPop ← GenerateCandidates(MLP, X_Best, RangeFromBestPt);
9           RangeFromBestPt ← AdaptRangeFromBestPt(RangeFromBestPt);
10          X_Pop ← X_Pop + X_NewPop;
11          X_Best ← Min(X_Pop, F(X_Pop));
12      end;
13      result ← X_Best;
14  end;
```

Figure 2: Adaptive neural network optimisation method – pseudocode.

The proposed adaptive neural network is implemented in the simulation optimisers searching for the global optimum and is described in Algorithm 1 (see Fig. 2). This iterative algorithm consists of several steps such as creation of initial population, training of the neural network to approximate the objective function, generation of candidates, adaptive setting of parameters for candidate generation and extending training data after each iteration.

The initial training population consists of randomly selected candidates using uniform distribution. The number of initial candidates is given by the *InitialPopulationSize* parameter. Gathered training candidates X_{Pop} and corresponding objective function values of these candidates $F(X_{\mathrm{Pop}})$ are used as training data.

The MLP is trained as a regression model to predict $\hat{F}(X)$. Therefore, we use the mean squared error (*MSE*) loss function. The Adam method [20] is used as a training optimiser and we utilise the early stopping technique. To standardise the inputs for various tasks and minimise high input values and gradients, the position coordinates (X_{Pop}) are normalised to an interval between 0 and 1. Optionally, $F(X_{\mathrm{Pop}})$ values can also be normalised for the training.

ALGORITHM 2: GenerateCandidates pseudocode

```
1   begin
2       X_Cand ⟵ RandomNormalDistribution(NumberOfCandidates × CandidateSelectionMul,
                                          μ = X_Best, σ = RangeFromBestPt);
3       Y_Cand ⟵ MLP(X_Cand); // F̂(X_Cand)
4       // sort X_Cand in ascending order according to Y_Cand
        X_Cand ⟵ Sort_a(X_Cand, Y_Cand);
5       X_Prob ⟵ array[size(X_Cand)];
6       σ ⟵ size(X_Cand)/ ProbSelectionSigmas;
7       for i ⟵ 0 to size(X_Cand) − 1 do begin
8           X_Prob[i] ⟵ e^(i²/−2σ²);
9       end;
10      s ⟵ sum(X_Prob);
11      for i ⟵ 0 to size(X_Cand) − 1 do begin
12          X_Prob[i] ⟵ X_Prob[i] / s;
13      end;
14      // select NumberOfCandidates based on their probabilities
        result ⟵ Select(X_Cand, X_Prob, NumberOfCandidates)
15  end;
```

Figure 3: Generating candidates – pseudocode.

Once we have the approximation model, we generate n candidates (X_{Cand}) and approximate their objective function values $\hat{F}(X_{\mathrm{Cand}})$ according to Fig. 3. The candidates are generated using normal distribution, where the mean is the currently best candidate $\mathbf{X}_{\mathrm{Best}}$, and the standard deviation is *RangeFromBestPt*. The number of generated candidates is:

$n = NumberOfCandidates \times CandidateSelectionMul.$

NumberOfCandidates candidates are selected based on their probability in a way that a candidate with a better approximated value has a higher probability to be selected. The probability of candidate selection is based on n, the *ProbSelectionSigmas* parameter and a gaussian function where $\sigma = n/ProbSelectionSigmas$.

ALGORITHM 3: AdaptRangeFromBestPt Pseudocode

```
1   begin
2       ImproveRatio ⟵ 0;
3       for i ⟵ 0 to NumberOfCandidates − 1 do begin
4           if F(X_NewPop[i]) < F(X_LastPop[i]) then
5               ImproveRatio ⟵ ImproveRatio + 1;
6           endif;
7       end;
8       ImproveRatio ⟵ ImproveRatio / NumberOfCandidates;
9       if (ImproveRatio < SuccessRatio ) then
10          RangeFromBestPt ⟵ RangeFromBestPt × RangeFromBestPtMultiplier;
11      else if (ImproveRatio > SuccessRatio) then
12          RangeFromBestPt ⟵ RangeFromBestPt / RangeFromBestPtMultiplier;
13      endif;
14  end;
```

Figure 4: AdaptRangeFromBestPt – pseudocode.

We set up the standard deviation of normal distribution *RangeFromBestPt* adaptively to allow both local and global optimisation to generate candidates (see Fig. 4). We define *ImproveRatio* as a ratio of candidates with better objective functions than candidates from the previous iteration. Further, we force *ImproveRatio* to match the defined *SuccessRatio* by multiplying or dividing by *RangeFromBestPtMultiplier*.

The resulting candidates after each iteration extend the training data. This way, more training candidates are present in more promising areas and iterative training is used to refine the $\hat{F}(X)$.

6. EVALUATION OF OPTIMISATION EXPERIMENTS

We propose using different evaluation criteria for analysing the efficiency of finding the optimum in the search space. These evaluation criteria can also analyse the behaviour of the optimisation method depending on different settings of the ANN method parameters. The behaviour of the tested ANN optimisation methods is partially random (the method contains elements of randomness e.g. generating candidate solutions), so we had to perform many optimisation experiments to identify the pure nature of the ANN method.

Evaluation criteria are calculated from the box plot characteristics. The same rules of testing methods were always followed – the same values of simulation model parameters in the optimisation experiments, the same rules of termination criteria using the information entropy, the same intervals of simulation model parameter values, the settings of the optimisation methods parameters etc. We tested different series to reduce the random behaviour of the ANN optimisation method depending on different settings of the ANN method parameters. The number of tested series depends on the number of the tested optimisation method parameters. The following table contains the number of tested series of optimisation methods – see Table II.

Table II: Number of tested series.

Methods	Differential Evolution	Downhill Simplex	SOMA Strategy	Evolution Strategy	Adaptive Neural Network	Particle Swarm Optimisation	Simulated Annealing	Random Search	Tabu Search	Hill Climbing	Local Search	Genetic Algorithm
Number of series	96	2,025	2,304	1,6416	1,944	470,400,000	3,840	1	900	90	15	26,127,360

We used intervals that include the recommended values of the optimisation methods parameters (if the parameter significantly influenced the behaviour of the optimisation method): *noNeurons* $\in [32, 64]$, *step* = 32; *InitialPopulationSize* $\in [100, 500]$, *step* = 100; *RangeFromBestPt* = 0.33; *RangeFromBestPtMultiplier* = 0.82; *SuccessRatio* $\in [0.5, 0.8]$, *step* = 0.1; *NumberOfCandidates* $\in [10, 50]$, *step* = 20; *CandidateSelectionMultiple* $\in [10, 50]$, *step* = 20; *ProbSelectionSigmas* $\in [2, 6]$, *step* = 2; *EarlyStopPatience* $\in [1, 5]$, *step* = 2.

After evaluating all the series performed on the simulation models using a weighted sum (including all criteria with different weights), the characteristics of the following box plot are calculated and visualised in the box plot chart – see Fig. 5.

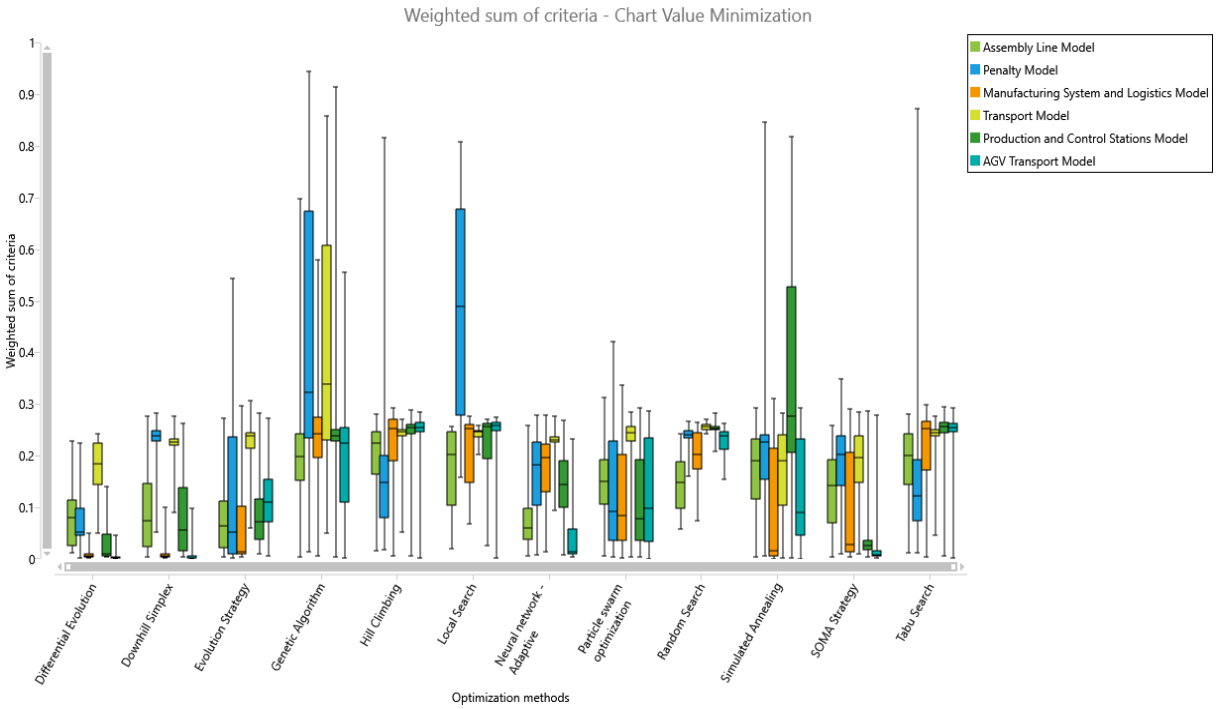

Figure 5: Weighted sum of evaluation criteria.

An ANN is a suitable method for applying to the tested discrete simulation models. The ANN achieved satisfying results for the tested series e.g., the AGV Transport discrete event simulation model contains 15 model input parameters (all the simulation models' input parameters could be varied within an interval of acceptable values specified for each model). When calculating the mean from the calculated means of all the series applied (calculated box plot characteristics) to all the discrete simulation models, the ANN is the fifth best method from the twelve tested optimisation methods – see Table III.

Table III: Arithmetic mean of weighted sum of evaluation criteria.

Methods	Differential Evolution	Downhill Simplex	SOMA Strategy	Evolution Strategy	Adaptive Neural Network	Particle Swarm Optimisation	Simulated Annealing	Random Search	Tabu Search	Hill Climbing	Local Search	Genetic Algorithm
Mean	0.066	0.109	0.115	0.117	0.135	0.146	0.191	0.208	0.216	0.224	0.268	0.296

7. CONCLUSION

This paper is focused on the testing and evaluation of an Adaptive Neural Network in comparison with various pseudo gradient, metaheuristic, evolutionary and swarm optimisation methods (and their combinations) – Random Search, Hill Climbing, Tabu Search, Local Search, Downhill Simplex, Simulated Annealing, Differential Evolution, Evolution Strategy and Particle Swarm Optimisation.

Various methods were used in the ANN testing, e.g. a case study – where an ANN was deployed for finding the global optimum on different discrete simulation models representing different problems in manufacturing processes; experimental simulations – simulated different situations in industrial plants without the need to intervene in physical reality; quantitative research – testing different parameters of optimisation methods that affected the efficiency of

finding the global optimum of the objective function by the optimisation method on different discrete simulation models.

Based on the tests and proposed different evaluation criteria, it is shown that an ANN can be used as a promising optimisation method for simulation optimisation of different discrete event simulation models. If we compare the effectivity of all the tested optimisation methods, the ANN method is in the top 5 best tested methods from the 12 optimisation methods.

There is great potential for modifying this method to improve its efficiency in finding the global optimum of the objective function specified for a discrete event simulation model. The proposed approach is general and requires no prior knowledge of the simulation model. However, in many cases, we do possess some understanding of the partial structure or a computational graph. Therefore, we plan to explore incorporating this knowledge into the NN architecture and/or approximating only unknown components.

Another possible direction is to use the model as a heuristic in conjunction with other methods (e.g. as a fitness function within a genetic algorithm). Finally, NNs can also approximate the derivative of the function, offering potential advantages for gradient-based techniques.

ACKNOWLEDGEMENTS

This paper was created with the subsidy of the project SGS-2024-032 ‘Intelligent production system’ carried out with the support of the Internal Grant Agency of the University of West Bohemia.

This work has been partly supported by the Grant No. SGS2022-016 ‘Advanced methods of data processing and analysis’.

REFERENCES

[1] Kagermann, H. (2015). Change through digitization – value creation in the age of Industry 4.0, Albach, H.; Meffert, H.; Pinkwart, A.; Reichwald, R. (Eds.), *Management of Permanent Change*, Springer Gabler, Wiesbaden, 23-45, doi:10.1007/978-3-658-05014-6_2

[2] Balderas, D.; Ortiz, A.; Méndez, E.; Ponce, P.; Molina, A. (2021). Empowering Digital Twin for Industry 4.0 using metaheuristic optimization algorithms: case study PCB drilling optimization, *The International Journal of Advanced Manufacturing Technology*, Vol. 113, Nos. 5-6, 1295-1306, doi:10.1007/s00170-021-06649-8

[3] Grznar, P.; Gregor, M.; Gaso, M.; Gabajova, G.; Schickerle, M.; Burganova, N. (2021). Dynamic simulation tool for planning and optimisation of supply process, *International Journal of Simulation Modelling*, Vol. 20, No. 3, 441-452, doi:10.2507/IJSIMM20-3-552

[4] Wang, L. (2005). A hybrid genetic algorithm–neural network strategy for simulation optimization, *Applied Mathematics and Computation*, Vol. 170, No. 2, 1329-1343, doi:10.1016/j.amc.2005.01.024

[5] Ezugwu, A. E.; Shukla, A. K.; Nath, R.; Akinyelu, A. A.; Agushaka, J. O.; Chiroma, H.; Muhuri, P. K. (2021). Metaheuristics: a comprehensive overview and classification along with bibliometric analysis, *Artificial Intelligence Review*, Vol. 54, No. 6, 4237-4316, doi:10.1007/s10462-020-09952-0

[6] Barton, R. R.; Meckesheimer, M. (2006). Chapter 18 Metamodel-based simulation optimization, Henderson, S. G.; Nelson, B. L. (Eds.), *Handbooks in Operations Research and Management Science: Simulation*, Elsevier / North-Holland, Amsterdam, Vol. 13, 535-574, doi:10.1016/S0927-0507(06)13018-2

[7] Greasley, A. (2020). Architectures for combining discrete-event simulation and machine learning, *Proceedings of the 10th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2020)*, 47-58, doi:10.5220/0009767600470058

[8] Sun, H. (2023). Optimizing manufacturing scheduling with genetic algorithm and LSTM neural networks, *International Journal of Simulation Modelling*, Vol. 22, No. 3, 508-519, doi:10.2507/IJSIMM22-3-CO13

[9] Grznar, P.; Gregor, M.; Gola, A.; Nielsen, I.; Mozol, S.; Seliga, V. (2022). Quick workplace analysis using simulation, *International Journal of Simulation Modelling*, Vol. 21, No. 3, 465-476, doi:10.2507/IJSIMM21-3-612

[10] Li, Z. P. (2022). Management decisions in multi-variety small-batch product manufacturing process, *International Journal of Simulation Modelling*, Vol. 21, No. 3, 537-547, doi:10.2507/IJSIMM21-3-CO15

[11] Al-Khateeb, B.; Ahmed, K.; Mahmood, M.; Le, D.-N. (2021). Rock hyraxes swarm optimization: a new nature-inspired metaheuristic optimization algorithm, *Computers, Materials & Continua*, Vol. 68, No. 1, 643-654, doi:10.32604/cmc.2021.013648

[12] Arora, K.; Kumar, A.; Kamboj, V. K.; Prashar, D.; Jha, S.; Shrestha, B.; Joshi, G. P. (2020). Optimization methodologies and testing on standard benchmark functions of load frequency control for interconnected multi area power system in smart grids, *Mathematics*, Vol. 8, No. 6, Paper 980, 23 pages, doi:10.3390/math8060980

[13] Wang, P.; Zhou, Y.; Luo, Q.; Han, C.; Niu, Y.; Lei, M. (2020). Complex-valued encoding metaheuristic optimization algorithm: a comprehensive survey, *Neurocomputing*, Vol. 407, 313-342, doi:10.1016/j.neucom.2019.06.112

[14] Raska, P.; Litvinenko, V. (2020). Methodology for evaluating optimization experiments, *Proceedings of the 32nd European Modeling & Simulation Symposium (EMSS 2020)*, 50-61, doi:10.46354/i3m.2020.emss.008

[15] Raska, P.; Ulrych, Z. (2019). Self-organizing migrating algorithm applied to discrete event simulation optimization, *Proceedings of the International Conference on Industrial Engineering and Operations Management*, 269-280

[16] Trigueiro de Sousa Junior, W.; Barra Montevechi, J. A.; de Carvalho Miranda, R.; Teberga Campos, A. (2019). Discrete simulation-based optimization methods for industrial engineering problems: a systematic literature review, *Computers & Industrial Engineering*, Vol. 128, 526-540, doi:10.1016/j.cie.2018.12.073

[17] Dokeroglu, T.; Sevinc, E.; Kucukyilmaz, T.; Cosar, A. (2019). A survey on new generation metaheuristic algorithms, *Computers & Industrial Engineering*, Vol. 137, Paper 106040, 29 pages, doi:10.1016/j.cie.2019.106040

[18] Hornik, K.; Stinchcombe, M.; White, H. (1989). Multilayer feedforward networks are universal approximators, *Neural Networks*, Vol. 2, No. 5, 359-366, doi:10.1016/0893-6080(89)90020-8

[19] Shen, Z.; Yang, H.; Zhang, S. (2021). Neural network approximation: three hidden layers are enough, *Neural Networks*, Vol. 141, 160-173, doi:10.1016/j.neunet.2021.04.011

[20] Kingma, D. P.; Ba, J. L. (2015). Adam: a method for stochastic optimization, *3rd International Conference on Learning Representations (ICLR 2015)*, 15 pages, doi:10.48550/arXiv.1412.6980