Int j simul model
ISSN 1726-4529

USING ADAPTIVE NEURAL NETWORK IN DISCRETE EVENT SIMULATION OPTIMISATION

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 evaluated the behaviour of the adaptive neural network with different evaluation criteria. We compared adaptive neural networks with various pseudo gradient, metaheuristic, evolutionary and swarm optimisation methods (and their combinations).
(Received in Month 202_, accepted in Month 202_. This paper was with the authors _ months for _ revisions.)

Key Words: Adaptive Neural Network, Optimisation Methods, Discrete-Event Simulation Optimisation, Industrial Engineering

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, organization, 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]–[5] 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. [6]
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.

DOI

‘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.’ [7]

We modified and tested an adaptive neural network (ANN) to find suitable settings for the input parameters of discrete simulation models with different specified objective functions. 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 [8]. An adaptive neural network was proposed as a local and global metamodel-based optimisation method in [9].

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 [10]. 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 metamodelling methods and have received only minor attention [11], 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. 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 as follows:

$$\tilde{X} = \prod_{j=1}^{n} \tilde{X}_j = \prod_{j=1}^{n} \left[a_j, b_j\right], a_j \leq b_j, a_j, b_j \in \mathbb{R} \quad (3)$$

where $\tilde{X}$ denotes the search space—the domain of the decision variables; j denotes the index of the j-th decision variable; n denotes the dimension of the search space; a_j denotes the lower bound of the interval of the j-th decision variable; b_j denotes the upper bound of the interval of the j-th 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 relatively simpler. The local minimum of an objective function is an element of the search space where $F: \tilde{X} \mapsto \mathbb{R} \land \tilde{X} \subseteq \mathbb{R}$:

$$\breve{\mathbf{X}}_l: \exists e > 0: F(\breve{\mathbf{X}}_l) \le F(\mathbf{X}) \forall \mathbf{X} \in \tilde{X}, |\mathbf{X} - \breve{\mathbf{X}}_l| < e \quad (4)$$

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 equation (4).
Figure 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 optimizer changes the settings of the decision variables of the model.

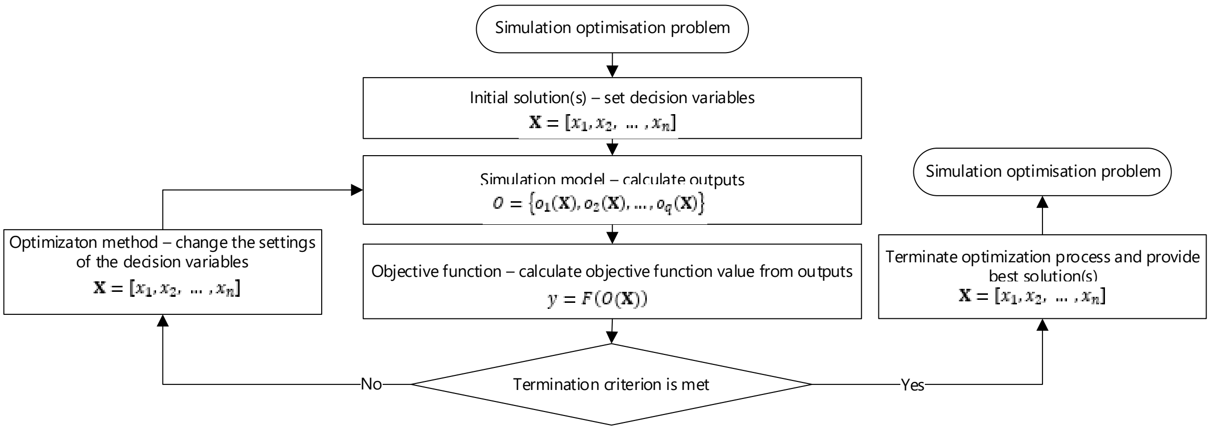

Figure 1: Simulation optimisation problem.

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 performed simulation experiments and reducing the amount of time taken for testing different settings of ANN. Settings of ANN optimisation method parameters affect the progress of the searching 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 decision variables – see equation (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 synchronize 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):

$$x_j = \mathbf{X}[j] \forall j: j = \{0,1,2,\ldots,n-1\} \quad (5)$$

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 in such a way that they are applicable for discrete event simulation optimisation. We have also modified some of these methods to improve their behaviour to find the global optimum and improve their efficiency (the methods use the principles used in evolutionary algorithms – mutation and generation). The selected optimisation methods have been applied to industrial practice problems modelled by discrete-event simulation models.

3. 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 due to trade secrets).
We tested the following models:

- **Assembly line model** - characterises the production line for two different manufacturing processes including the relative error rates at each workstation. The aim is to maximize 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 maximize the utilization of all assembly lines and all tow tractors.
- **Transport model** – the model describes the transport from the warehouse to the 8 production lines using tow tractors. The goal of the simulation study is to find the correct sequence of loading points for the 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 utilization of all assembly lines, which is superior to the average utilisation of all types of AGVs.

4. 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 [7], [12], [13].

We have developed our own applications for managing the parallel simulation optimisers 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 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:

$$\mathbf{X}_i = X_{\text{Pop}}[i] \forall i: i = \{0,1,2,\dots,m-1\} \tag{6}$$

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 a multilayer perceptron (MLP) to approximate the objective function from already gathered candidates, since MLPs are good function approximators [14]. We utilize 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, which is enough to reproduce any continuous function [15]. 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 × *noNeurons* and *noNeurons* neurons respectively.

ALGORITHM 1: ANN Optimisation pseudocode

begin
    $X_{\text{Pop}} \longleftarrow$ CreateInitialPopulation();
    $X_{\text{NewPop}} \longleftarrow X_{\text{Pop}}$;
    // the best solutions candidate in X_{Pop} according to objective function $F(X)$
    $\mathbf{X}_{\text{Best}} \longleftarrow \text{Min}(X_{\text{Pop}}, F(X_{\text{Pop}}))$;
    while not TerminationCriterion( ) do begin
        // train MLP to approximate objective function
        $MLP \longleftarrow \text{Train}(MLP, X_{\text{Pop}}, F(X_{\text{Pop}}))$;
        $X_{\text{LastPop}} \longleftarrow X_{\text{NewPop}}$;
        $X_{\text{NewPop}} \longleftarrow$ GenerateCandidates(MLP, $\mathbf{X}_{\text{Best}}$, RangeFromBestPt);
        RangeFromBestPt ⟵ AdaptRangeFromBestPt(RangeFromBestPt);
        $X_{\text{Pop}} \longleftarrow X_{\text{Pop}} + X_{\text{NewPop}}$;
        $\mathbf{X}_{\text{Best}} \longleftarrow \text{Min}\left(X_{\text{Pop}}, F\left(X_{\text{Pop}}\right)\right)$;
    end;
    result ⟵ $\mathbf{X}_{\text{Best}}$;
end;

Figure 2: Adaptive neural network optimisation method pseudocode.

Algorithm 1 (see Figure 2) 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.
Initial population is used as an initial training population for the neural network. It consists of randomly selected candidates using uniform distribution. The number of initial candidates is given by the *InitialPopulationSize* parameter.
Objective function approximation $\hat{F}(X)$ using MLP. Gathered training candidates X_{Pop} and corresponding objective function values of these candidates $F(X_{\text{Pop}})$ 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 [16] is used as an 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, $F(X_{\text{Pop}})$ values can be also normalised for the training. The results after each iteration are used to extend the training data. This way, more training candidates are present in more promising areas and iteratively training is used to refine the $\hat{F}(X)$.

ALGORITHM 2: GenerateCandidates pseudocode

begin
  $X_{\text{Cand}} \leftarrow \text{RandomNormalDistribution}\begin{pmatrix}\text{NumberOfCandidates} \times \text{CandidateSelectionMul}, \\ \mu = \mathbf{X}_{\text{Best}}, \sigma = \text{RangeFromBestPt}\end{pmatrix}$;
  $Y_{\text{Cand}} \leftarrow \text{MLP}(X_{\text{Cand}})$; // $\hat{F}(X_{\text{Cand}})$
  // sort X_{Cand} in ascending order according to Y_{Cand}
  $X_{\text{Cand}} \leftarrow \text{Sort}_{\text{a}}(X_{\text{Cand}}, Y_{\text{Cand}})$;
  $X_{\text{Prob}} \leftarrow \text{array}[\text{size}(X_{\text{Cand}})]$;
  $\sigma \leftarrow \text{size}(X_{\text{Cand}}) / \text{ProbSelectionSigmas}$;
  for i $\leftarrow$ 0 to $\text{size}(X_{\text{Cand}}) - 1$ do begin
    $X_{\text{Prob}}[\text{i}] \leftarrow \text{e}^{\text{i}^2/-2\sigma^2}$;
  end;
  $\text{s} \leftarrow \text{sum}(X_{\text{Prob}})$;
  for i $\leftarrow$ 0 to $\text{size}(X_{\text{Cand}}) - 1$ do begin
    $X_{\text{Prob}}[\text{i}] \leftarrow X_{\text{Prob}}[\text{i}] / \text{s}$;
  end;
  // select NumberOfCandidates based on their probabilities
  result $\leftarrow \text{Select}(X_{\text{Cand}}, X_{\text{Prob}}, \text{NumberOfCandidates})$
end;

Figure 3: Generating candidates method pseudocode.

Once we have the approximation model, we generate n candidates (X_{Cand}) and approximate their objective function values $\hat{F}(X_{\text{Cand}})$ according to Figure 3. The candidates are generated with the use of normal distribution with the currently best candidate $\mathbf{X}_{\text{Best}}$ as a mean and *RangeFromBestPt* as a standard deviation. The number of generated candidates is given by *NumberOfCandidates* and *CandidateSelectionMul* parameters according to the following equation: $n = NumberOfCandidates \times CandidateSelectionMul$.
NumberOfCandidates candidates are selected based on their probability in a way that the candidate with the 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

```
begin
    ImproveRatio ⟵ 0;
    for i ⟵ 0 to NumberOfCandidates − 1 do begin
        if F(X_NewPop[i]) < F(X_LastPop[i]) then
            ImproveRatio ⟵ ImproveRatio + 1;
        endif;
    end;
    ImproveRatio ⟵ ImproveRatio / NumberOfCandidates;
    if (ImproveRatio < SuccessRatio ) then
        RangeFromBestPt ⟵ RangeFromBestPt × RangeFromBestPtMultiplier;
    else if (ImproveRatio > SuccessRatio) then
        RangeFromBestPt ⟵ RangeFromBestPt / RangeFromBestPtMultiplier;
    endif;
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 Figure 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 defined *SuccessRatio* by multiplying or dividing by *RangeFromBestPtMultiplier*.

5. EVALUATION OF OPTIMISATION EXPERIMENTS

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 [17]–[21]. 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.

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 (minimum, lower quartile, median, and upper quartile and maximum). 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 - 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, was found in each series (a series consists of replicated simulation optimisation experiments with different settings of the optimisation algorithms according to different characteristics of the optimisation):

$$|F(\mathbf{X}_i) - F(\mathbf{X}^*)| \leq \varepsilon \quad (7)$$

where $F(\mathbf{X}_i)$ denotes the objective function of the found solution candidate of the i-th optimisation experiment (the solution whose objective function is in the tolerated deviation from the value of the optimum of the objective function $\varepsilon = 0.001$); $F(\mathbf{X}^*)$ denotes the objective function value of the global optimum.

The difference between the optimum and sub-optimum - 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 this evaluation function):

$$f_{2\,i} = \left(\frac{F(\mathbf{X}^*) - F(\mathbf{X}^*{}_i)}{\Delta F_{\tilde{X}}}\right) \forall i: i = \{1,2,\dots,s\}, f_{2\,i} \in [0,1] \quad (8)$$

where $F(\mathbf{X}^*)$ denotes the objective function value of the best found candidate solution of the search space in all series; $F(\mathbf{X}^*{}_i)$ denotes the objective function value of the best solution candidate found in i-th series; $\Delta F_{\tilde{X}}$ denotes the difference between the objective function value of the found best and worst candidate solutions of the search space in all series; s denotes the number of performed series.

This criterion is useful when there is no series which contains any optimum or a suboptimum whose objective function value is within the tolerance of the optimum objective function value.

The distances of quartiles - the distance between the quartiles of a concrete series (the aim is to minimise this evaluation function). If the first criterion equals zero, then the third criterion also equals zero (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:

$$f_{3\,i} = \frac{f_{3w1\,i} + f_{3w2_i} + f_{3w3\,i} + f_{3w4\,i} + f_{3w5\,i}}{\Delta F_{\tilde{X}}}, \forall i: i = \{1,2,\dots,s\}, f_{3\,i} \in [0,1] \quad (9)$$

$$f_{3w5_i} = |F(\mathbf{X}^*{}_i) - F(\mathbf{X}^*)|, f_{3w4_i} = w_{4_{f_3}}|F(\mathbf{X}^*{}_i) - Q_{1_i}|, f_{3w3\,i} = w_{3_{f_3}}|Q_{1_i} - Q_{2_i}|, f_{3w2\,i} = w_{2_{f_3}}|Q_{2_i} - Q_{3_i}|, f_{3w1\,i} = w_{1_{f_3}}|Q_{3_i} - F(\mathbf{X}_{Worst_i})| \quad (10)$$

where i denotes the index of the series; $F(\mathbf{X}^*{}_i)$ denotes the objective function value of the best solution candidate of the i-th series; $w_{4_{f_3}}$ denotes the weight of the objective function values between the best solution candidate and the lower quartile Q_{1_i} of the i-th series (the values of the weights are defined based on the results of the simulation experiments and their sum equals 1); $w_{3_{f_3}}$ denotes the weight of objective function values between the lower quartile Q_{1_i} and median Q_{2_i} of the i-th series; $w_{2_{f_3}}$ denotes the weight of objective function values between median Q_{2_i} and the upper quartile Q_{3_i} of the i-th series; $w_{1_{f_3}}$ denotes the weight of objective function values between the upper quartile $Q_{3_i}$and the objective function value of the worst found possible solution $F(\mathbf{X}_{Worst_i})$ of the i-th series; s denotes the number of the performed series (different settings of the optimisation algorithm parameters); $\Delta F_{\tilde{X}}$ denotes the difference between the objective function values of the found best and worst candidate solutions of the search space in all series.

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 (the aim is to minimise this evaluation function):

$$f_{4\,i} = \frac{f_{4w1\,i} + f_{4w2_i} + f_{4w3\,i} + f_{4w4\,i} + f_{4w5\,i}}{\tilde{X}_H}, \forall i: i = \{1,2,\dots,s\}, f_{4_i} \in [0,1] \quad (11)$$

$$f_{4w5_i} = |min_{SE_i} - 1|, f_{4w4_i} = w_{4_{f_4}}|min_{SE_i} - Q_{1_i}|, f_{4w3\,i} = w_{3_{f_4}}|Q_{1_i} - Q_{2_i}|, f_{4w2\,i} = w_{2_{f_4}}|Q_{2_i} - Q_{3_i}|, f_{4w1\,i} = w_{1_{f_4}}|Q_{3_i} - max_{SE}| \quad (12)$$

where parameters i, s have the same nature as in the previous equation – see (9); min_{SE} denotes the 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} denotes the maximum number of simulation experiments that the optimisation method performed in the optimisation experiment to find the best solution candidate of the i-th series; $w_{4_{f_4}}$ denotes the weight (penalty) of values between min_{SE} and the lower quartile Q_{1_i} of the i-th series; $w_{3_{f_4}}$ denotes the weight of values between the lower quartile Q_{1_i} and median Q_{2_i} of the i-th series; $w_{2_{f_4}}$ denotes the weight of values between median Q_{2_i} and the upper quartile Q_{3_i} of the i-th series; $w_{1_{f_4}}$ denotes the weight of values between the upper quartile $Q_{3_i}$and the max_{SE} of the i-th series; $\tilde{X}_H$ denotes the maximum number of simulation experiments that the optimisation method can perform in each optimisation experiment in all series – using the entropy termination criterion.

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:

$$f_{5\,i} = \frac{f_{5w1\,i} + f_{5w2_i} + f_{5w3\,i} + f_{5w4\,i} + f_{5w5\,i}}{\Delta F_{\tilde{X}}}, \forall i: i = \{1,2,\dots,s\}, f_{5\,i} \in [0,1] \quad (13)$$

$$f_{5w5_i} = |F(\mathbf{X}^*{}_i) - F(\mathbf{X}^*)|, f_{5w4_i} = w_{4_{f_5}}|F(\mathbf{X}^*{}_i) - Q_{1_i}|, f_{5w3\,i} = w_{3_{f_5}}|Q_{1_i} - Q_{2_i}|, f_{5w2\,i} = w_{2_{f_5}}|Q_{2_i} - Q_{3_i}|, f_{5w1\,i} = w_{1_{f_5}}|Q_{3_i} - F(\mathbf{X}_{Worst_i})| \quad (14)$$

where the parameters have the same nature as in the previous equation – see equation (9) and (*10*); the weights are defined for individual differences in box plot characteristics – the same principle as in equation (*10*).

More detailed information on the evaluation methodology can be found in [22].

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:

$$w_{f_j} \in [0,1] \forall j: j = \{1,2,\dots,5\}, \sum_{j=1}^{5} w_{f_j} = 1 \quad (15)$$

where w_{f_j} denotes the weight of the j-th criterion - the Saaty method (see [23]) was used to set the individual criteria weights -see (16).

$$w_{f_1} = 0.22, w_{f_2} = 0.56, w_{f_3} = 0.12, w_{f_4} = 0.06, w_{f_5} = 0.04 \quad (16)$$

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 (17).

$$f_i = w_{f_1} \cdot (1 - f_{1\,i}) + \sum_{j=2}^{5} f_{j\,i} \cdot w_{f_j} \forall i: i = \{1,2,\dots,s\} \quad (17)$$

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 (9).

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

Table I: Number of tested series

RS	DS	LS	HC	TS	SA	DE	ES	SOMA	PSO	GA	ANN
1	2025	15	90	900	3840	96	16416	2304	470400000	26127360	1944

After evaluating all the series (series = replicated optimisation experiments with the same setting of the optimisation method parameters) 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 Figure 5.

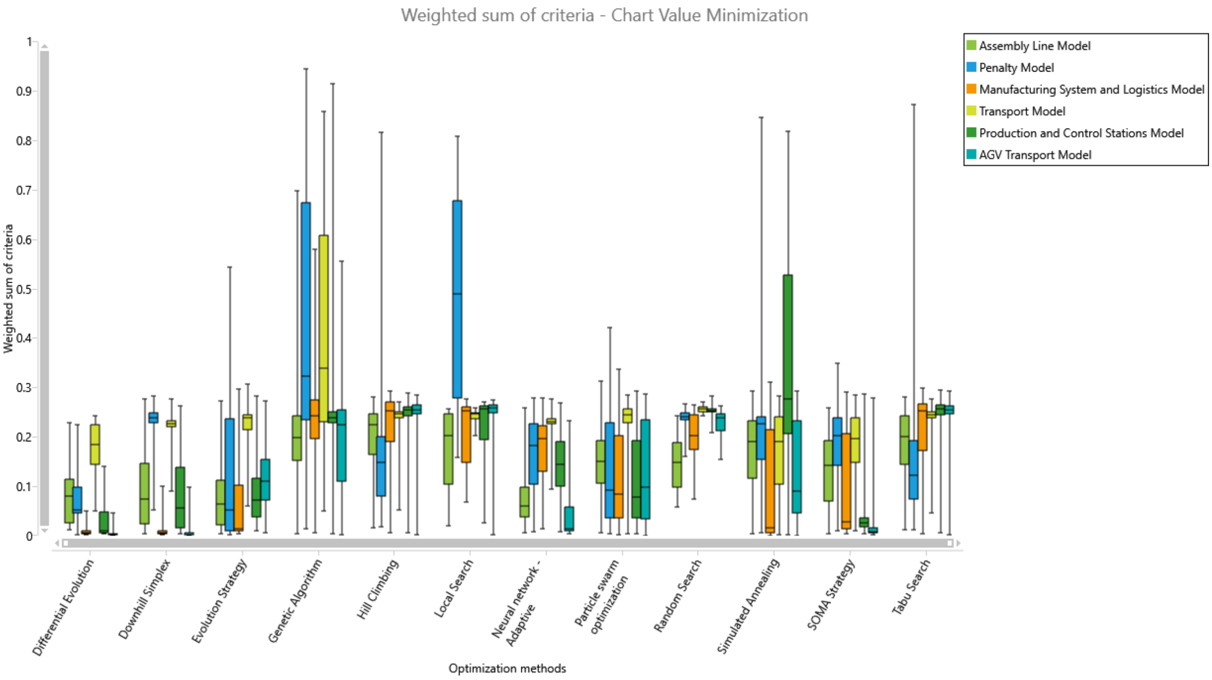

Figure 5: Weighted sum of evaluation criteria.

An ANN is a suitable method which can be applied 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 II.

Table II: 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

6. 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. Based on the tests, it is shown that an ANN can be used as a promising optimisation method for discrete simulation optimisation. If we compare the effectivity of all the tested optimisation methods, the ANN method is in the top 5 best tested methods. There is great potential for modifying this method to improve its efficiency in finding the global optimum.

ACKNOWLEDGEMENTS

This paper was created with the subsidy of the project SGS-2021-028 'Developmental and training tools for the interaction of man and the cyber–physical 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] H. Kagermann, "Change through digitization—value creation in the age of industry 4.0," in *Management of Permanent Change*, 2015. doi: 10.1007/978-3-658-05014-6_2.

[2] D. Balderas, A. Ortiz, E. Méndez, P. Ponce, and A. Molina, "Empowering Digital Twin for Industry 4.0 using metaheuristic optimisation algorithms: case study PCB drilling optimisation," *International Journal of Advanced Manufacturing Technology*, vol. 113, no. 5–6, 2021, doi: 10.1007/s00170-021-06649-8.

[3] G. Büchi, M. Cugno, and R. Castagnoli, "Smart factory performance and Industry 4.0," *Technol Forecast Soc Change*, vol. 150, p. 119790, Jan. 2020, doi: 10.1016/J.TECHFORE.2019.119790.

[4] I. Lee and K. Lee, "The Internet of Things (IoT): Applications, investments, and challenges for enterprises," *Bus Horiz*, vol. 58, no. 4, 2015, doi: 10.1016/j.bushor.2015.03.008.

[5] L. Atzori, A. Iera, and G. Morabito, "Understanding the Internet of Things: definition, potentials, and societal role of a fast evolving paradigm," *Ad Hoc Networks*, vol. 56, 2017, doi: 10.1016/j.adhoc.2016.12.004.

[6] L. Wang, "A hybrid genetic algorithm-neural network strategy for simulation optimisation," *Appl Math Comput*, vol. 170, no. 2, 2005, doi: 10.1016/j.amc.2005.01.024.

[7] A. E. Ezugwu *et al.*, "Metaheuristics: a comprehensive overview and classification along with bibliometric analysis," *Artif Intell Rev*, vol. 54, no. 6, 2021, doi: 10.1007/s10462-020-09952-0.

[8] L. Wang, "A hybrid genetic algorithm-neural network strategy for simulation optimisation," *Appl Math Comput*, vol. 170, no. 2, 2005, doi: 10.1016/j.amc.2005.01.024.

[9] R. R. Barton and M. Meckesheimer, “Chapter 18 Metamodel-Based Simulation Optimisation,” *Handbooks in Operations Research and Management Science*, vol. 13, no. C. 2006. doi: 10.1016/S0927-0507(06)13018-2.

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

[11] R. R. Barton, “Simulation optimisation using metamodels,” in *Proceedings - Winter Simulation Conference*, 2009. doi: 10.1109/WSC.2009.5429328.

[12] W. Trigueiro de Sousa Junior, J. A. Barra Montevechi, R. de Carvalho Miranda, and A. Teberga Campos, “Discrete simulation-based optimisation methods for industrial engineering problems: A systematic literature review,” *Comput Ind Eng*, vol. 128, 2019, doi: 10.1016/j.cie.2018.12.073.

[13] T. Dokeroglu, E. Sevinc, T. Kucukyilmaz, and A. Cosar, “A survey on new generation metaheuristic algorithms,” *Comput Ind Eng*, vol. 137, p. 106040, Nov. 2019, doi: 10.1016/J.CIE.2019.106040.

[14] G. Cybenko and G. Cybenkot, “Approximation by superpositions of a sigmoidal function. Math Cont Sig Syst (MCSS) 2:303-314 Mathematics of Control, Signals, and Systems Approximation by Superpositions of a Sigmoidal Function*,” *Math. Control Signals Systems*, vol. 2, 1989.

[15] A. N. Kolmogorov, “On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition,” *American Mathematical Society Translations Series 2*, vol. 17. pp. 369–373, 1961.

[16] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimisation,” in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[17] B. Al-Khateeb, K. Ahmed, M. Mahmood, and D. N. Le, “Rock Hyraxes Swarm Optimisation: A New Nature-Inspired Metaheuristic Optimisation Algorithm,” *Computers, Materials and Continua*, vol. 68, no. 1, 2021, doi: 10.32604/cmc.2021.013648.

[18] K. Arora *et al.*, “Optimisation 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, 2020, doi: 10.3390/MATH8060980.

[19] K. Hussain, M. N. M. Salleh, S. Cheng, and R. Naseem, “Common benchmark functions for metaheuristic evaluation: A review,” *International Journal on Informatics Visualization*, vol. 1, no. 4–2. 2017. doi: 10.30630/joiv.1.4-2.65.

[20] P. Wang, Y. Zhou, Q. Luo, C. Han, Y. Niu, and M. Lei, “Complex-valued encoding metaheuristic optimisation algorithm: A comprehensive survey,” *Neurocomputing*, vol. 407, 2020, doi: 10.1016/j.neucom.2019.06.112.

[21] A. M. Ahmed, T. A. Rashid, and S. A. M. Saeed, “Cat Swarm Optimisation Algorithm: A Survey and Performance Evaluation,” *Computational Intelligence and Neuroscience*, vol. 2020. 2020. doi: 10.1155/2020/4854895.

[22] P. Raska and Z. Ulrych, “Methodology for evaluating optimisation experiments,” in *32nd European Modeling and Simulation Symposium, EMSS 2020*, 2020. doi: 10.46354/i3m.2020.emss.008.

[23] T. L. Saaty and L. G. Vargas, *DECISION MAKING WITH THE Economic , Political , Social and Technological Applications with Benefits , Opportunities , Costs and Risks*, vol. 95. 2006.