Iğdır Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 12(3):1277 -1291, 2022
Journal of the Institute of Science and Technology, 12(3): 1277 - 1291, 2022
ISSN: 2146-0574, eISSN: 2536-4618
Computer Engineering **DOI: 10.21597/jist.1099154**
Research Article
Received: 05.04.2022 **Accepted: 16.05.2022**

To Cite: İmik Şimsek O, Alagöz B B, 2022. Model Based Demand Order Estimation by Using Optimal Architecture Artificial Neural Network with Metaheuristic Optimizations. Journal of the Institute of Science and Technology, 12(3): 1277 - 1291.

Model Based Demand Order Estimation by Using Optimal Architecture Artificial Neural Network with Metaheuristic Optimizations

Özlem IMİK SİMŞEK[1*], Barış Baykant ALAGÖZ[2]

ABSTRACT: With the increase of e-commerce volumes in recent years, it is useful to estimate daily demand order numbers in order to improve the demand forecasts, production-distribution planning and sales services. In this manner, data-driven modeling and machine learning tools have been preferred to enhance demand order predictions, timely delivery, incomes and customer satisfaction in electronic trading because real-time data collection is possible in e-commerce platforms. Artificial Neural Networks (ANNs) are widely used for data-driven modeling and prediction problems. Since affecting the approximation performance of neural network function, the modeling performance of ANNs strongly depends on the architecture of neural networks, and architectural optimization of ANN models has become a main topic in the neuroevolution field. This study presents an architecture optimization method that implements Particle Swarm Optimization (PSO) and Differential Evolution (DE) algorithms to optimize ANN model architecture for the estimation of total demand order numbers from the sparse demand order data. In this approach, PSO and DE algorithm only optimizes neural model architecture according to an effective network search policy and the training of ANN models is carried out by using backpropagation algorithm. This neural architecture model optimization approach considers generalization of data, reducing neuron and training epoch numbers and it can yield an optimal architecture data-driven neural model for estimation of the daily total orders. In the experimental study, optimal architecture ANN models are obtained according to the daily order forecasting dataset.

Keywords: Neural network model, hyper-parameter optimization, orders demand forecasting, particle swarm optimization, differential evolution

[1]Özlem IMIK ŞİMŞEK (**Orcid ID:** 0000-0002-4192-0255), İnönü University, Faculty of Engineering, Department of Computer Engineering, Malatya, Turkey
[2]Barış Baykant ALAGÖZ (**Orcid ID:** 0000-0001-5238-6433), İnönü University, Faculty of Engineering, Department of Computer Engineering, Malatya, Turkey
***Corresponding Author:** Özlem İMİK ŞİMŞEK, e-mail: oimiksimsek@gmail.com

INTRODUCTION

Today, with the increase of e-commerce and global trading, the uncertainty in demand and distribution side increases, and this uncertainty makes planning of production and logistic stages difficult and negatively affects trading. Also, there is a growing tendency in the diversity of the demands. Such diversity and increase in uncertainty in the demand orders increase workload of the companies due to the poor planning and latencies in order responses. Particularly, uncertainty in order requests complicates management of orders and decision making processes in logistics. Consistent estimation of order volumes can contribute to production, transportation and distribution planning of companies and support efficiency in decision-making stages. These improvements can serve for producer and consumer satisfaction.

Artificial Neural Networks (ANNs) have been widely used in prediction problems (Aminian and Shahhosseini, 2008). Estimation performance of ANNs depends on hyper-parameters such as network architecture, training algorithms, type of activation function (Carvalho et al., 2011; Akay et al., 2021). Network architectures were commonly selected by trial and error method and it can negatively affect the training and test performance (Kapanova et al., 2018). Optimal ANN architecture also depends on the content of the training datasets. Deep learning and deep neural network are very advantageous when learning high-level features and it is very effective in feature extraction in large and difficult dataset (Pacal et al., 2020), and applications of deep learning algorithms in detection and classification systems such as in medical systems (Pacal and Karaboga, 2021; Pacal ae al., 2022), remote sensing systems (Jeppesen et al., 2019), production and quality monitoring systems (Zhao et al., 2021) have been demonstrated. Deep neural networks can be effectively utilized in many areas. Size of training data can significantly affect accuracy of classification performance of Convolutional Neural Networks (Cho et al., 2015). Depth of neural network and hyper-parameters should be optimal according to dataset content (Kaya et al.,2019; Akay et al., 2021), an optimal network depth and neural complexity is useful to improve the learning performance of the neural networks according to datasets Use of a metaheuristic method for the architecture optimization of ANNs enables automatic-determination of optimal neural architectures according to the dataset (Carvalho et al., 2011; Kapanova et al., 2018; Ettaouil and Ghanou, 2009; Akay et al., 2021) An up-to-date review study for optimizing deep learning models by using metaheuristics has been presented by Akay et al.

Metaheuristic methods were also used in intelligent problem solving (Caserta and Voß, 2009). In recent years, it has been seen that data mining problems can be considered as optimization problems and metaheuristic methods can be used to solve them (Dhaenens and Jourdan, 2022). Combining ANN models with metaheuristic optimization allows intelligent problem solving based on data-driven learning models as shown in Figure 1. In this scheme, design of optimal ANN models requires neuroevolution (Floreano et al., 2008) according to content of data flow and metaheuristic optimization can effectively optimize configuration of ANN and automatically improve data-driven modeling skills of ANN models.

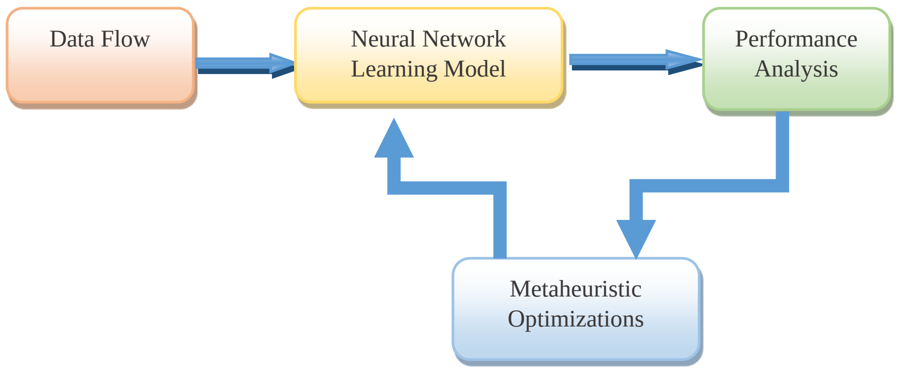

Figure 1. A block diagram for metaheuristic data analysis based on data-driven learning models for intelligent problem solving

The first studies began with the first simple artificial neuron model proposed in 1943 by physiologist Warren McCulloch and mathematician Walter Pitts to mimic the functioning of biological neurons (McCulloch and Pitts, 2008). In 1961, Rosen Blatt came up with the idea of a backpropagation algorithm for training multilayer networks (Widrow and Lehr, 1990). In this way, the use of multi-layer artificial neural networks in every field has become widespread in many areas in recent years. Artificial neural networks have evolved into a computational intelligence tool used in data science classification, identification, modeling and prediction problems (Çevik et al., 2014), applied to problems ranging from speech recognition to protein secondary structure prediction, classification of cancers and gene prediction (Krogh, 2008). If an example of application areas in this subject is given, in control (Hasanien, 2011), signal processing (Vijaya et al., 1998), prediction problems (Aminian and Shahhosseini, 2008), image processing (Egmont-Petersen et al., 2002)… etc.

Metaheuristic algorithms can develop the solution by using trial and error techniques in order to solve complex and highly nonlinear optimization problems and they are widely used computation tools in applications such as adjusting system parameters to their optimal values according to simulation results or experimental data (Liu et al., 2021; Birs et al., 2020). Among population-based metaheuristic methods, Particle Swarm Optimization (PSO) has been used in a wide variety of applications. In current study, performance improvements of the PSO algorithm are demonstrated for architectural optimization of ANNs for data-driven modeling. Since PSO can provide satisfactory results in multimodal problems, there are a wide variety of areas where PSO were used (Poli et al., 2007). At the same time, another metaheuristic algorithm, which is tested in this study, is the Differential Evolution (DE) algorithm. The DE algorithm is an effective evolutionary search algorithm for optimization problems (Slowik and Bialko, 2008). The DE algorithm is preferred due to its potential in searching the global minimum of a multimodal function with few parameters to adjust in the algorithm compared to Genetic Algorithm (GA) (Landa Becerra and Coello, 2006). PSO and DE algorithms have been used to improve performance of ANNs. In the architecture optimization problem, two PSO algorithms have been used hierarchically for both weight coefficient and architecture optimization (Carvalho and Ludermir, 2008). In another study, the PSO algorithm was used for the weight and structure optimization of three layer ANNs (Yu et al., 2008). On the other hand, the performance of the DE algorithm was demonstrated for the training of radial based neural networks (Oh et al., 2012), feed-forward neural networks (Ilonen et al., 2003). Recently, hyperparameters of Long Short-term Memory (LSTM) networks were optimized to automate LSTM configuration such as hidden neuron numbers and batch size (Nakisa et al., 2018). Nakisa et al concluded that the DE and PSO algorithms could improve the average accuracy of the optimized LSTM network in emotion classification problems. On the other hand, it is known a fact that when the number of optimization parameters highly increases, search performances of metaheuristic methods severely decrease. Therefore, the training of large ANN networks cannot be effectively performed by using metaheuristic methods so that there are a large number of weight coefficients to be optimized. The metaheuristic methods can be rather effective in architectural optimization of ANN models because the architectural optimization requires optimization of quite less parameters compared to weight coefficient numbers in training tasks of neural networks.

In this study, architectural optimizations of ANN estimation models are performed using PSO and DE algorithms by employing Carvalho et al.'s neural architecture objectives (Carvalho et al., 2011). Carvalho et al.'s objectives aim the improvement of generalization, reducing neuron numbers in the network and number of training epochs for faster training of networks. The training and architectural optimization are performed separately by using different optimization techniques. This hybrid approach allows benefiting from advantages of relevant optimization algorithms in order to improve total daily demand order estimation results. In this hybrid optimization, the training of optimal ANNs is only performed by using Levenberg-Marquardt (LM) backpropagation method. Thus, training tasks are conducted separately by using an effective training algorithm in order to obtain an improved estimation model. Architectural optimization is only performed by using metaheuristic algorithms. This objective function was previously improved for the self-configuring of ANNs by using a multi-particle collision algorithm in (Anochi et al., 2016; Badr et al., 2019). In the current study, Carvalho et al.'s objective is enhanced to perform more effective heuristic search of neural architecture parameters. We applied the minimum value search policy on the Carvalho et al.'s objective by using several repeated training trails of the candidate neural architecture. In the experimental study, we used daily demand forecasting orders data set (UCI, 2007) which involves 60-day order data from a real database of a Brazilian company of large logistics. The mean square error performance of all ANN models is used to evaluate the architectural performance of ANNs in the PSO and DE optimization process. We observed satisfactory performance in daily order demand by using this architecture optimization approach. The results obtained were compared with other methods obtained from previous studies.

MATERIALS AND METHODS

Introduction of Particle Swarm Optimization Algorithm

Particle swarm optimization has inspired the swarm behavior of living things. It is a population-based metaheuristic optimization algorithm that employs the searching mechanism that aims to find the best solution by navigating swarm individuals representing the solution in the search space (Eberhart and Kennedy, 1995; Shi and Eberhart, 1998; Wang et al., 2018). One of the main advantages of PSO is that it uses collective intelligence that is called swarm intelligence (Eberhart and Kennedy, 1995; Wang et al., 2018; Imik and Alagoz, 2017). The fact that the number of parameters that need to be adjusted according to the problem is relatively low compared to other classical metaheuristic methods provides ease of use (Eberhart and Kennedy, 1995). The PSO models the motion of particles representing the candidate solution towards the best solutions in the search space by including social swarm interactions (Wang et al., 2018; Clerc and Kennedy, 2002). The PSO algorithm was effectively used in several engineering problems (Çevik et al., 2014; Imik and Alagoz, 2017; İmik Şimşek, 2018).

At the start of the PSO algorithm, the particles are randomly distributed over the search space (del Valle et al., 2008). In each iteration, the fitness value of each particle representing candidate solutions is calculated. The local and global best particles of the swarm are determined according to the fitness value. In the next iteration, the motion of each particle is updated considering the positions of the obtained local and global best particle (Wang et al., 2018). This process continues until the last iteration is completed. Each particle tends to constantly move towards better solutions, guided by the local and global best particles.

In the multidimensional search space, let the position of particle i be denoted by $x_i[t]$ and its velocity by $v_i[t]$. The PSO algorithm updates the velocities of the particles according to the following equation (Eberhart and Kennedy, 1995; Shi and Eberhart, 1998; Wang et al., 2018).

$$V_i[t+1] = w[t] \times V_i[t] + c_1 r_1 \times (X_L[t] - X_i[t]) + c_2 r_2 (X_G[t] - X_i[t]) \quad (1)$$

In response to this new velocity of the particle, the new position of the particle is updated with

$$X_i[t+1] = X_i[t] + V_i[t+1]. \quad (2)$$

In this structure, each particle's position and velocity in an n-dimensional search space are represented by an n-dimensional vector. In Figure 2, the determination of the direction of motion of a particle according to the local and global best particle is shown as a representation. In equation 1, the local best solution is expressed as $X_L[t]$ and the global best solution as $X_G[t]$. The coefficient c_1 is the local learning coefficient and it adjusts motions of particles towards the local best particle. The coefficient c_2 is the global learning coefficient and it adjusts motions of particles towards the global best particle (Imik and Alagoz, 2017). Parameters r_1 and r_2 are random numbers and allow random movement in particle motions. This freedom degree in particle motion enriches the possibility of searching different regions of search space and this enables finding different optimal points at several runs of the algorithm. The acceleration parameter $w[t]$ causes particles to slow down according to $w[t+1] = \xi w[t], 0 < \xi \leq 1$ at each iteration step (Shi and Eberhart, 1998). The acceleration $w[t]$ decreases as the iterations progress, allowing the particles to settle into the good solutions they find (Imik and Alagoz, 2017).

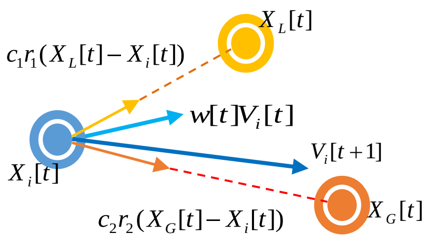

Figure 2. The effects of local and global values on particle motion in the PSO algorithm (İmik Şimşek, 2018)

Introduction of Differential Optimization Algorithm

The DE algorithm is one of popular evolutionary optimization methods. The DE algorithm uses essential genetic processes (mutation, crossover and selection) and implements relatively simple formulas in order to generate a new generation of candidate solutions. It maintains the best candidates through generations by selecting them according to their fitness performance. (Storn and Price 1997; Qin et al. 2009). A candidate solution of the DE algorithm is expressed for a D-dimensional parameter space as

$$X_{i,G} = \{x_{i,G}^1\, x_{i,G}^2\, x_{i,G}^3 \dots x_{i,G}^D\},\ i = 1,2,3,\cdots,N_p\ , \quad (3)$$

where the subscript G denotes the generation number and the parameter N_p represents the population size of the candidate solution set (Qin et al. 2009). To form an initial population of candidate solutions, candidate solutions are distributed uniformly into the entire search space by using uniform random numbers. In the current study, the following formulations of DE algorithm are implemented for the genetic processing (Qin et al. 2009):

(i) *Mutation Process*: The mutation process enriches the genetic properties of the population in the search space. The new candidate solutions are represented by the vector set $\boldsymbol{V_{i,G}} = \{\boldsymbol{v^1_{i,G}}, \boldsymbol{v^2_{i,G}}, \boldsymbol{v^3_{i,G}}, \cdots \boldsymbol{v^D_{i,G}}\}$. In this study, the "DE/rand/1" mutation strategy is used (Qin et al. 2009).

$$V_{i,G} = X_{r^i_1,G} + S.(X_{r^i_2,G} - X_{r^i_3,G}) \quad (4)$$

where the subscripts r^i_1, r^i_2, r^i_3 are randomly selected numbers of individuals from the population in the range of $[1, N_p]$. The parameter S is the scale factor that is used to adjust length of the difference vector $(X_{r^i_2,G} - X_{r^i_3,G})$(Qin et al. 2009).

(ii) *Crossover Process*: A crossover operation is used with a crossover rate of $\boldsymbol{C_r}$ to form new individuals of the population. The new candidate solution of the crossover process are represented by the vector set $\boldsymbol{U_{i,G}} = \{\boldsymbol{u^1_{i,G}}, \boldsymbol{u^2_{i,G}}, \boldsymbol{u^3_{i,G}}, \cdots, \boldsymbol{u^D_{i,G}}\}$. New individuals are formed by randomly selected part from mutated solutions $\boldsymbol{V_{i,G}}$ (Qin et al. 2009):

$$u^j_{i,G} = \begin{cases} v^j_{i,G} & (rand_j[0,1) \leq C_r) or\ (j = j_{rand}) \\ x^j_{i,G} & else \end{cases}, j = 1,2,3,\cdots,D \quad (5)$$

(iii) *Selection Process*: The selection process is very essential process of differential evolution that allows the maintenance of good candidate solutions through generations of evolution process. The objective function value of new individuals and old individuals are denoted by $\boldsymbol{f(U_{i,G})}$ and $\boldsymbol{f(X_{i,G})}$ respectively. The best fitting individuals for a minimization problem are selected according to the the objective function values of them as follows (Qin et al. 2009).

$$X_{i,G+1} = \begin{cases} U_{i,G}, & f(U_{i,G}) \leq f(X_{i,G}) \\ X_{i,G}, & else \end{cases} \quad (6)$$

Design of Optimal Architecture Artificial Neural Network with Metaheuristic Optimizations

This section demonstrates the realization of architectural optimization of neural networks by using metaheuristic optimization. The noise, linearity, and complexity of the data affect the performance of ANN models. In order to obtain an optimum training performance, a suitable configuration of the neural network model should be determined according to the training dataset. Therefore, data-driven ANN modeling requires architectural optimization to reach an optimal modeling performance in terms of generalization of data, neuron number and training speed. When architecture configuration that complies with these features is obtained, this ANN architecture is optimal for the training and test dataset and this enhances modeling performance. This effort is called self-configuration or architectural optimization of ANNs (Anochi et al., 2014; Carvalho et al., 2011). Manual determination of the optimal architecture according to trial and error method is time consuming (Kapanova et al., 2018), the automation of this process by using metaheuristic based architecture optimization is important for the data-driven ANN modeling efforts in order to obtain satisfactory model performance. It is necessary in case of real-time intelligent system operation, where models should be updated continuously according to real-time data flow.

To implement PSO and DE algorithms to optimize ANN architecture, the number of neurons in the layers of the ANN is considered as a candidate vector. Accordingly, the optimized parameter vectors of the PSO and DE algorithms define the number of neurons in each layer. This real number vector is represented as $\mathrm{H_{LMH}}$ and it is a raw description of the number of neurons in each layer because it is composed of noninteger numbers. Therefore, this vector is interpreted to a feasible description by applying a repairmen process. Figure 3 shows the steps of processing the $\mathrm{H_{LMH}}$ vector to generate a feasible exposition for the neural architecture description $\mathrm{H_L}$vector from $\mathrm{H_{LMH}}$ vector. There are two processes, namely integerization and neural repair processes. The candidate solution vector $\mathrm{H_{LMH}}$ is first converted to the integer vector $\mathrm{H_{Lint}}$. This operation is called integer enumeration. After this process, $\mathrm{H_{Lint}}$ vector is obtained. However, layers with zero neurons can be found in the $\mathrm{H_{Lint}}$ vector. This means that the network is cut off. For this reason, it is necessary to remove layers with zero neuron number. By removing the zero-valued layer from the network, the $\mathrm{H_L}$vector becomes ready to describe a feasible neural network. The architecture of ANN models is configured according to the $\mathrm{H_L}$ vector. This architecture with $\mathrm{H_L}$ is trained by using an ANN backpropagation algorithm. The architectural performance of this candidate solution is calculated according to the formulation given by Carvalho et al. (Carvalho et al., 2011). This architectural performance in the objective function is used by metaheuristic algorithms to develop the next-generation candidate solution $\mathrm{H_{LMH}}$.

The main purpose of automatically configuring an ANN model is the ability to obtain a near-optimal ANN architecture without requiring the ANN approach and/or the knowledge of any expert in its implementation (Anochi et al., 2014; Anochi et al., 2016). The results obtained in this way prevent unnecessary loss of time and effort.

In this study, we implemented the objective function that was proposed by Carvalho et al. (Carvalho et al., 2011) and improved by Anochi et al. (Anochi et al., 2014). To enhance search of the optimal neural architecture by using metaheuristics, the minimum value of the cost function E_j for N number of training trials was used for more opportunistic evaluation of performnce of the $\mathrm{H_L}$ architecture. The cost function for N trials was written according to Carvalho et al.'s formulation as

$$E_j = penalty_j \times \left(\frac{\rho_1 \times sse_{train,j} + \rho_2 \times sse_{gen,j}}{\rho_1 + \rho_2}\right), \tag{7}$$

where weights $\rho_1 = 1$ and $\rho_2 = 0.1$ are the same weight values used in (Carvalho et al., 2011). The sse_{train} and sse_{gen} are Sum of Square Error (SSE) for the repeated training and test phases of each architecture with $\mathrm{H_L}$.

$$sse_{train,j} = \frac{1}{2}\sum_{i=1}^{Ntrain}(f_{ANN}(x_i) - y_i)^2, \tag{8}$$

$$sse_{gen,j} = \frac{1}{2}\sum_{i=1}^{Ngen}(f_{ANN}(x_{g,i}) - y_{g,i})^2, \tag{9}$$

where N_{train} is the number of data in the training dataset while N_{gen} is the number of data in the test dataset. In this study, ANN training was performed with the backpropagation algorithm with random initial weights. Thus, each training session for the same architecture $\mathrm{H_L}$ can provide models with different performances because of different convergence paths. In order to perform more consistent performance analyzes for each $\mathrm{H_L}$ configuration, training and test calculations were repeated N times with the H_L configurations and minimum value of E_j is used to assess the performance of each H_L configuration. Accordingly, the objective function f_{obj} is modified as

$$f_{obj} = \min\{E_j | j = 1,2,..N\} \tag{10}$$

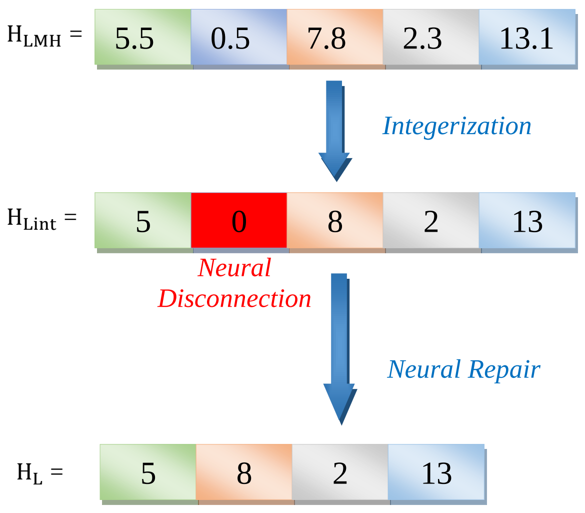

Figure 3. Steps of processing the candidate solution of metaheuristic optimization to obtain a feasible interpretation for the neural architecture description

The $y_{nn} = f_{ANN}(x_i)$ represents the approximation function of the trained ANN model. Besides improving the training and test performances by minimizing sse_{train} and sse_{gen}, other important architectural optimization goals are network complexity (number of neurons) and training speed (number of epochs). The penalty factor in equation 7 is formulated for j^{th} training trial as follows (Carvalho et al., 2011; Anochi et al., 2014)

$$penalty_j = \gamma_1(\varepsilon_{neu,j})^2 \times \gamma_2(epochs_j) + 1, \tag{11}$$

where γ_1 and γ_2 are the tuning parameters used to achieve a balance between the number of neurons (ε_{neu}) and the number of training epoch ($epochs$) in the optimization process. These two parameters were configured $\gamma_1 = 0.1$ and $\gamma_2 = 0.001$. The ε_{neu} parameter is the total number of neurons in the neural network. It is calculated by adding the number of neurons in each hidden layer. (Sum of elements of vector H_L)

Considering sse_{train} and sse_{gen} in this architecture optimization, optimal ANN architecture can improve generalization performance of ANN models. With $penalty$ term, the network complexity can be reduced and the computation time for the training of optimal architecture ANN can be increased. Figure 4 shows basic system components of this optimal architecture data-driven modeling system that manages the demand order forecasting. The next section demonstrates an experimental study for this system.

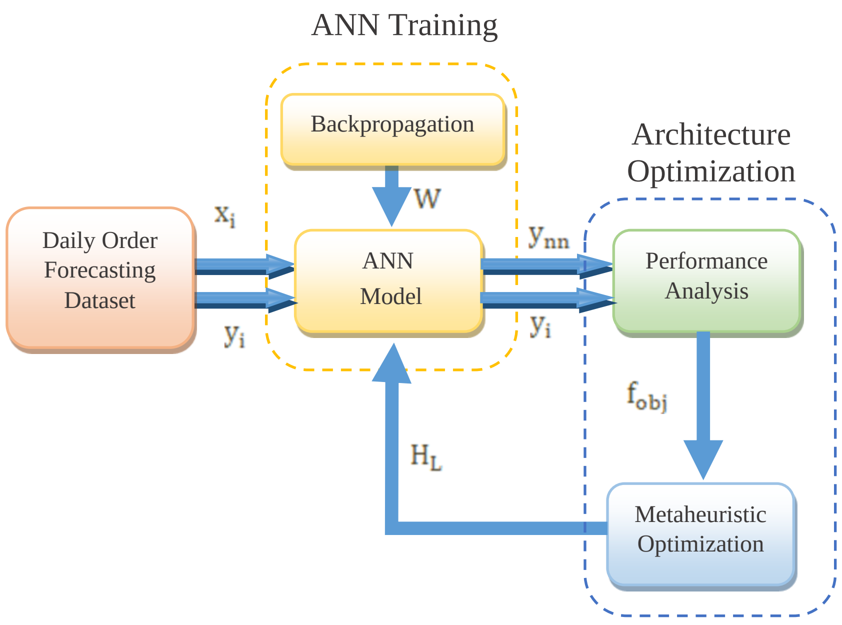

Figure 4. Basic system components for optimal architecture ANN based data-driven modeling of daily total order data by using metaheuristic optimization

RESULTS AND DISCUSSION

Experimental Study

The demand order dataset used in this study involves twelve predictive attributes from 60 business days from a real database of a Brazilian company of large logistics (UCI, 2007), Ten predictive attributes that exclude time data (week of the month and day of the week) are used. There Table 1 shows details of 12 entries of the data. The dataset includes twelve attributes as input and a target that is the total of orders for daily treatment. In this data, we used 40 of them for training and 20 of them for testing purposes. While the dataset reserved for training was used for training and optimization, the dataset reserved for testing was used for performance evaluation. Since the backpropagation algorithm of ANN can yield different training performances, average performance of resulting ANN models is evaluated according to 20 times independent training of each candidate architecture.

For the architectural optimization by PSO algorithm, 30 particles were used in the optimization. The maximum number of iterations is limited to 50. The individual learning coefficients were set as $c_1 = 2.0$ and $c_2 = 2.0$. The acceleration coefficients are taken as $w = 1$ and $\xi = 0.99$. At the same time, the configuration settings of the DE algorithm, which is another metaheuristic method to compare results of swarm-intelligence based optimization and evolutionary optimization in this article, are 0.2 for crossover probability (pcr), lower limit of scaling factor 0.2, and upper limit of scaling factor 0.8. The maximum number of iterations is limited to 50.

Figure 5a shows decrease of f_{obj} function values during the architectural optimization of ANN models by using the PSO. The figure shows that the PSO minimizes the objective function to optimize neural architecture for the improved total order demand forecast. Figure 5b shows the reduction of f_{obj} values during the architectural optimization of the ANN model using DE. Figure 6a shows the actual data and total daily demand order estimates of the optimal architecture ANN models for the test data set. As can be seen, it almost overlaps with the test data. Figure 6b shows similar results for the DE algorithm. As can be seen, there is overlap with the data. In Figure 7a and Figure 7b, changes of the absolute error ($e = y_d - y_{pso}, e = y_d - y_{de}$) are shown, which are calculated by subtracting the estimated value obtained from the real data. It is seen that the ANN model optimized with PSO and DE is very consistent in total daily order estimation.

Table 1. Daily order demand forecasting dataset

Data	**Model Input / Output**
Week of the month (first, second, third or fourth week); numerical equivalents {1.0, 2.0, 3.0, 4.0, 5.0}	Input
Day of the week (Monday to Friday); numerical equivalents {2.0, 3.0, 4.0, 5.0, 6.0}	Input
Urgent orders	Input
Non urgent orders	Input
Type A orders	Input
Type B orders	Input
Type C orders	Input
Orders from the tax sector	Input
Orders from the traffic controller sector	Input
Banking sector orders (1)	Input
Banking sector orders (2)	Input
Banking sector orders (3)	Input
The daily total demand order (Ground truth data)	Output

Table 2 shows the Mean Relative Error (MRE), Mean Square Error (MSE) and Mean Absolute Error (MAE) performances. Considering the MRE performance, when the results of the studies conducted in (Simsek and Alagoz, 2021) and (Ferreira et al., 2016) are examined, the optimal ANN used in this study may provide a better performance. MRE of the ANN model in (Simsek and Alagoz, 2021) was obtained 0.000015. The MRE value of the ANN model of Ferreira et al. is 0.0006 (Ferreira et al., 2016). The lowest MRE that was obtained in this study is 2.16 10^{-7} by using the optimal ANN-DE model. The MRE is 2.82 10^{-7} for the optimal ANN-DE model. When the values in the table are compared, the optimal ANN models with PSO and DE significantly improve the demand order estimates. The MSEs and MAEs of optimal ANN models with DE and PSO are considerably lower than ANN models in the literature. The ANN model of Simsek et al. implemented Gray Wolf Optimization (GWO) algorithm to perform the rectangular architecture optimization of ANN, where only wide and depth parameters of the rectangular ANN architecture are optimized (Simsek and Alagoz, 2021). This limits data-driving modeling performance of ANNs with GWO to the rectangular shape architectures. This restriction can deteriorate optimality of the resulting networks and reduces the performance. The results in the table apparently show that optimal architecture ANN models can contribute to the data-driven modeling performance of ANNs in this application. For this reason, more advanced architectural optimization based on the modified Carvalho et al.'s objectives is implemented in the current study.

Table 2. Performance of demand order estimation models with metaheuristic optimizations

Parameter			
Training Algorithm	**Mean Square Error (MSE)**	**Mean Absolute Error (MAE)**	**Mean Relative Error (MRE)**
ANN-PSO	1.13$10^{-2}$	6.92 10^{-2}	2.82 10^{-7}
ANN-DE	9.15 10^{-3}	6.31 10^{-2}	2.16 10^{-7}
ANN-GWO (Simsek and Alagoz, 2021)	31.34	3.60	15.0 10^{-6}

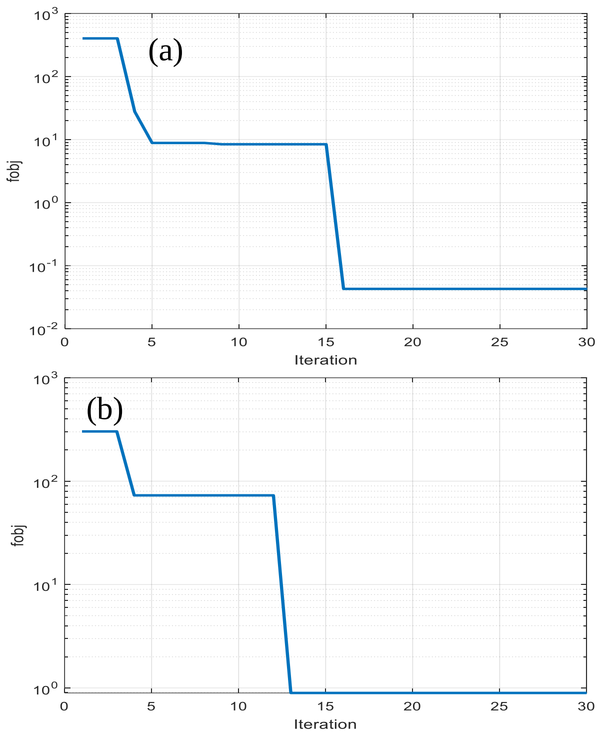

Figure 5. The convergence of the PSO (a) and DE (b) algorithm in ANN optimization

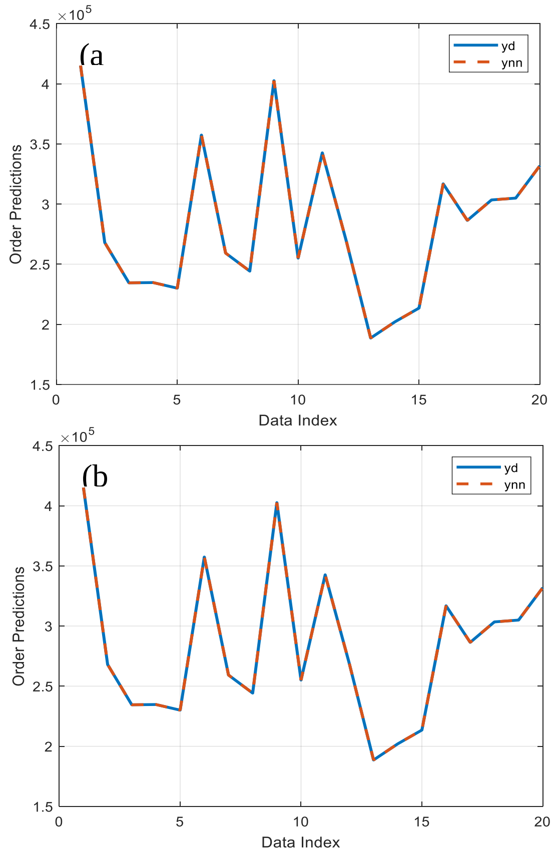

Figure 6. ANN model optimized with PSO (a) and DE (b) algorithm for order demand forecast actual data (yd) and forecast data (ynn)

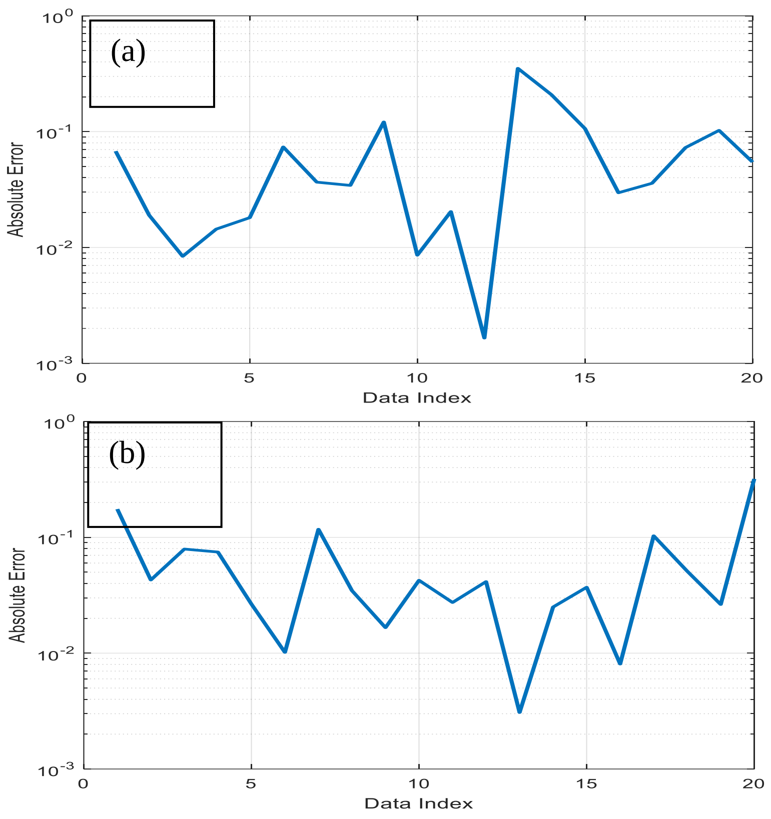

Figure 7. PSO (a) and DE (b) distribution of the absolute error of the ANN estimator model for each measurement according to the ground truth data

CONCLUSION

In this article, a neuroevolution method was demonstrated to obtain ANN models with optimal architecture for the order demand estimation problem. Two different optimization algorithms were implemented for the data-driven architectural optimization of the ANN models. These metaheuristic algorithms are the PSO and the DE algorithms. To enhance the architecture search task by using metaheuristic methods, the neuroevolution objective was enhanced by employing the minimum value policy of the objective (Equation 7). Results indicated that the DE and PSO algorithm could significantly improve the learning performance of ANN based estimation models.

In the experimental study, satisfactory daily demand order estimation results were obtained by using an architectural optimal ANN model. The experimental results show that ANN-PSO model can significantly improve the MSE, MAE and MRE performances compared to the previous work results.

Conflict of Interest

The article authors declare that there is no conflict of interest between them.

Author's Contributions

The authors declare that they have contributed equally to the article.

REFERENCES

Akay, B, Karaboga, D, Akay, R 2021. A comprehensive survey on optimizing deep learning models by metaheuristics. Artificial Intelligence Review 5: 1–66.

Aminian J, Shahhosseini S, 2008. Evaluation of ANN modeling for prediction of crude oil fouling behavior. Applied Thermal Engineering 28(7): 668–674.

Anochi JA, Velho HFC, Furtado HCM, 2014. Self-configuring Two Types of Neural Networks by MPCA. Conference: 2nd International Symposium on Uncertainty Quantification and Stochastic ModelingAt: Rouen - France 5(2). Doi: https://doi.org/10.17265/2159-5275/2015.02.008.

Anochi J, Sambatti S, Luz E, Velho HC, 2016. New learning strategy for supervised neural network: MPCA meta-heuristic approach. 1st BRICS Countries & 11th CBIC Brazilian Congress on Computational Intelligence. Location: Recife, Brasil. Porto de Galinhas Beach, pp:1–6.

Badr EM, Salam MA, Ahmed H, 2019. Optimizing Support Vector Machine using Gray Wolf Optimizer Algorithm for Breast Cancer Detection.

Birs I, Folea S, Prodan O, Dulf E, Muresan C, 2020. An experimental tuning approach of fractional order controllers in the frequency domain. Applied Sciences (Switzerland) 10(7). Doi: https://doi.org/10.3390/APP10072379

Carvalho AR, Ramos FM, Chaves AA, 2011. Metaheuristics for the feedforward artificial neural network (ANN) architecture optimization problem. Neural Computing and Applications 20(8):1273–1284. Doi: https://doi.org/10.1007/s00521-010-0504-3

Carvalho M, Ludermir TB, 2008. Particle Swarm Optimization of Neural Network Architectures and Weights, 336–339. Doi: https://doi.org/10.1109/HIS.2007.45

Caserta M, Voß S, 2009. Metaheuristics: Intelligent Problem Solving. Doi: https://doi.org/10.1007/978-1-4419-1306-7_1

Cho J, Lee K, Shin E, Choy G, Do S, 2015. How much data is needed to train a medical image deep learning system to achieve necessary high accuracy?. arXiv preprint arXiv:1511.06348.

Clerc M, Kennedy J, 2002. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6(1): 58–73. Doi: https://doi.org/10.1109/4235.985692.

Çevik K, Koçer E, 2014. Parçacık Sürü Optimizasyonu ile Yapay Sinir Ağları Eğitimine Dayalı Bir Esnek Hesaplama Uygulaması. Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi 17(2): 39–45. https://dergipark.org.tr/tr/pub/sdufenbed/issue/20801/222008.

Del Valle Y, Venayagamoorthy GK, Mohagheghi S, Hernandez JC, Harley RG, 2008. Particle swarm optimization: Basic concepts, variants and applications in power systems. IEEE Transactions on Evolutionary Computation 12(2): 171–195. Doi: https://doi.org/10.1109/TEVC.2007.896686.

Dhaenens C, Jourdan L, 2022. Metaheuristics for data mining: survey and opportunities for big data. Annals of Operations Research 2022: 1–24. Doi: https://doi.org/10.1007/S10479-021-04496-0.

Eberhart R, Kennedy J, 1995. A new optimizer using particle swarm theory. Undefined 39–43. Doi: https://doi.org/10.1109/MHS.1995.494215.

Egmont-Petersen M, De Ridder D, Handels H, 2002. Image processing with neural networks- A review. In Pattern Recognition (Vol. 35, Issue 10). Doi: https://doi.org/10.1016/S0031-3203(01)00178-9.

Ettaouil M, Ghanou Y, 2009. Neural architectures optimization and Genetic algorithms. Wseas Transactions On Computer 8(3): 526–537.

Ferreira RP, Martiniano A, Ferreira A, Ferreira A, Sassi RJ, 2016. Study on Daily Demand Forecasting Orders using Artificial Neural Network. IEEE Latin America Transactions 14(3): 1519–1525. Doi: https://doi.org/10.1109/TLA.2016.7459644.

Floreano D, Dürr P, Mattiussi C, 2008. Neuroevolution: from architectures to learning. Evolutionary Intelligence 2008 1(1): 47–62. Doi: https://doi.org/10.1007/S12065-007-0002-4.

Hasanien HM, 2011. FPGA implementation of adaptive ANN controller for speed regulation of permanent magnet stepper motor drives. Energy Conversion and Management 52(2): 1252–1257. Doi: https://doi.org/10.1016/j.enconman.2010.09.021.

Ilonen J, Kamarainen JK, Lampinen J, 2003. Differential evolution training algorithm for feed-forward neural networks. Neural Processing Letters 17(1): 93–105. Doi: https://doi.org/10.1023/A:1022995128597.

Imik O, Alagoz BB, 2017. Discretization of fractional order transfer functions by weighted multi-objective particle swarm optimization method. IDAP 2017 - International Artificial Intelligence and Data Processing Symposium. Doi: https://doi.org/10.1109/IDAP.2017.8090162 .

Imik SO, Alagoz BB, 2021. Daily Forecasting of Demand Orders with Optimal Architecture Artificial Neural Network Learning Models. 2021 International Conference on Information Technology, ICIT 2021 - Proceedings, 355–360. Doi: https://doi.org/10.1109/ICIT52682.2021.9491784.

İmik ŞÖ, 2018. Parcacık Sürüsü Optimizasyon Yöntemi ile Kesir Dereceli Filtre Gerçekleşmesi ,İnönü Üniversitesi, Master Thesis (Printed). http://abakus.inonu.edu.tr/xmlui/handle/11616/14805.

Jeppesen JH, Jacobsen RH, Inceoglu F, Toftegaard TS, 2019. A cloud detection algorithm for satellite imagery based on deep learning. Remote sensing of environment 229:247-259.

Kapanova KG, Dimov I, Sellier JM, 2018. A genetic approach to automatic neural network architecture optimization. Neural Computing and Applications 29(5): 1481–1492. Doi: https://doi.org/10.1007/S00521-016-2510-6.

Kaya Y, Hong S, Dumitras T, 2019. Shallow-deep networks: Understanding and mitigating network overthinking. In International conference on machine learning, PMLR, pp: 3301-3310.

Krogh A, 2008. What are artificial neural networks? Nature Biotechnology 2008 26(2): 195–197. Doi: https://doi.org/10.1038/nbt1386.

Landa Becerra R, Coello CAC, 2006. Cultured differential evolution for constrained optimization. Computer Methods in Applied Mechanics and Engineering 195(33–36): 4303–4322. Doi: https://doi.org/10.1016/J.CMA.2005.09.006.

Liu X, Wang GG, Wang L, 2021. LSFQPSO: quantum particle swarm optimization with optimal guided Lévy flight and straight flight for solving optimization problems. Engineering with Computers. Doi: https://doi.org/10.1007/S00366-021-01497-2.

McCulloch WS, Pitts W, 2008. A logical calculus of the ideas immanent in nervous activity 5: 115–133.

Nakisa B, Rastgoo MN, Rakotonirainy A, Maire F, Chandran V, 2018. Long short term memory hyperparameter optimization for a neural network based emotion recognition framework. IEEE Access 6:49325–49338.

Oh SK, Kim WD, Pedrycz W, Joo SC, 2012. Design of K-means clustering-based polynomial radial basis function neural networks (pRBF NNs) realized with the aid of particle swarm optimization and differential evolution. Neurocomputing 78(1): 121–132. Doi: https://doi.org/10.1016/J.NEUCOM.2011.06.031.

Pacal I, Karaboga D, Basturk A, Akay B, Nalbantoglu U, 2020. A comprehensive review of deep learning in colon cancer. Computers in Biology and Medicine 126:104003.

Pacal I, Karaboga D, 2021. A robust real-time deep learning based automatic polyp detection system. Computers in Biology and Medicine 134: 104519.

Pacal I, Karaman A, Karaboga D, Akay B, Basturk A, Nalbantoglu U, Coskun S, 2022. An efficient real-time colonic polyp detection with YOLO algorithms trained by using negative samples and large datasets. Computers in biology and medicine 141: 105031.

Poli R, Kennedy J, Blackwell T, 2007. Particle swarm optimization. Swarm Intell 1(November): 33–57. Doi: https://doi.org/10.1007/s11721-007-0002-0.

Qin AK, Huang VL, Suganthan PN, 2009. Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization. IEEE Transactions on Evolutionary Computation 13:398–417. https://doi.org/10.1109/TEVC.2008.927706.

Shi Y, Eberhart R, 1998. Modified particle swarm optimizer. Proceedings of the IEEE Conference on Evolutionary Computation, ICEC, 69–73. Doi: https://doi.org/10.1109/icec.1998.699146

Slowik A, Bialko M, 2008. Training of artificial neural networks using differential evolution algorithm. 2008 Conference on Human System Interaction, HSI 2008, 60–65. Doi: https://doi.org/10.1109/HSI.2008.4581409.

Storn R, Price K, 1997. Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization 11:341:284–287.

UCI, 2007. Daily Demand Forecasting Orders Data Set. https://archive.ics.uci.edu/ml/datasets/Daily+Demand+Forecasting+Orders

Vijaya G, Kumar V, Verma HK, 1998. ANN-based QRS-complex analysis of ECG. Journal of Medical Engineering and Technology 22(4). Doi: https://doi.org/10.3109/03091909809032534.

Wang D, Tan D, Liu L, 2018. Particle swarm optimization algorithm: an overview. Soft Computing 22(2): 387–408. Doi: https://doi.org/10.1007/S00500-016-2474-6.

Widrow B, Lehr MA, 1990. 30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation. Proceedings of the IEEE 78(9): 1415–1442. Doi: https://doi.org/10.1109/5.58323.

Yu J, Wang S, Xi L, 2008. Evolving artificial neural networks using an improved PSO and DPSO. Neurocomputing 71(4–6): 1054–1060. Doi: https://doi.org/10.1016/J.NEUCOM.2007.10.013.

Zhao W, Chen F, Huang H, Li D, Cheng W, 2021. A new steel defect detection algorithm based on deep learning. Computational Intelligence and Neuroscience 2021: 5592878.