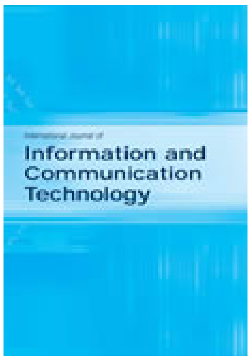


International Journal of Information and Communication Technology

ISSN online: 1741-8070 - ISSN print: 1466-6642
https://www.inderscience.com/ijict

Application of evolutionary neural model in path optimisation of simulation system

Meng Xu, Cheng Xu, Luyao Pei, Bo Wang, Zhixuan Lv

DOI: 10.1504/IJICT.2025.10074323

Article History:

Received:	30 August 2025
Last revised:	13 September 2025
Accepted:	14 September 2025
Published online:	12 November 2025

Application of evolutionary neural model in path optimisation of simulation system

Meng Xu*, Cheng Xu, Luyao Pei,
Bo Wang and Zhixuan Lv

Energy Development Research Institute,
China Southern Power Grid,
Guangzhou, Guangdong, 510700, China
Email: xu.meng2007@163.com
Email: xucheng_mail@126.com
Email: peiluyao1@163.com
Email: ashley1989131@163.com
Email: 13609704215@163.com
*Corresponding author

Abstract: This research introduces an evolving neural model that combines learning from neural networks with evolutionary computation techniques to optimise paths in simulation systems. The suggested approach overcomes the shortcomings of traditional algorithms when faced with dynamic, complicated situations by merging structural and parametric optimisation. To improve flexibility, convergence speed, and generalisation performance, the model uses an actor-critic reinforcement learning scheme, evolutionary field optimisation, and neural architecture search. Analyses of experimental data show that the suggested method is more efficient, stable, and accurate than more conventional methods like Q-learning and DQN. Path planning, risk minimisation, and live system simulation are three areas where the results show evolving neural models could improve intelligent decision-making.

Keywords: topics covered include digital twins; computational intelligence; evolutionary algorithms; EAs; evolutionary neural networks; path optimisation; simulation systems; and actor-critical models.

Reference to this paper should be made as follows: Xu, M., Xu, C., Pei, L., Wang, B. and Lv, Z. (2025) 'Application of evolutionary neural model in path optimisation of simulation system', *Int. J. Information and Communication Technology*, Vol. 26, No. 39, pp.92–113.

Biographical notes: Meng Xu is a senior researcher at the Energy Development Research Institute, China Southern Power Grid, Guangzhou, Guangdong, China. Her research focuses on renewable energy systems, low-carbon power transition strategies, and intelligent optimisation in smart grids.

Cheng Xu is a Research Engineer at the Energy Development Research Institute, China Southern Power Grid. His primary research interests include energy system modelling, distributed energy management, and grid stability analysis.

Luyao Pei is an associate researcher specialising in clean energy technologies and sustainable power development. Her work emphasises energy efficiency improvement and integrated resource planning.

Bo Wang is a research associate at the Energy Development Research Institute. His research areas include renewable energy integration, power system simulation, and smart energy solutions.

Zhixuan Lv is an engineer focusing on data-driven analysis for power system operation and low-carbon energy transition.

1 Introduction

Reverse design (RD) is a method that is utilised extensively in nearly every area of engineering. It involves digitising a physical component to generate a computational or virtual representation of it. One way to showcase a digital twin is by mimicking this digital or virtual model. A novel idea for characterising a new wave of modelling and simulation, the digital twin has only just begun to see widespread use. In the past, simulation tools were only employed during the design phase. However, presently, many forms of simulated tools are utilised for testing, validation, and optimisation, which is also known as the digital twin. Industry, and the manufacturing sector in particular, is fixated on this new technology, which is still in its infancy. An ideal definition of a digital twin would be the seamless bidirectional integration of data from real and virtual machines. Monitoring performance, optimising progress, simulating results, and predicting possible faults are some of the many uses of digital twins in various industries (Fuller et al., 2020), particularly manufacturing. This method has several applications throughout a product's lifespan, from conception to manufacturing, distribution, use, and eventual disposal. With digital twins, creativity, design, and development can reap enormous benefits in the future, particularly considering the increasing desire for personalised products and the adoption of Industry 4.0.

In a nutshell, RE is a set of principles used to glean product design details. As a result, it is possible to reverse-engineer components used in many different industries, such as engineering, medicine, software, consumer electronics, and define systems (Lo et al., 2021). According to this strategy, it is helpful at many points throughout the product life cycle, including ideation, production, shipping, usage, and finally, disposal. The future of innovation, design, and development stands to gain a great deal from digital twins, especially when thinking about the. Two processes, evolution and learning, interact to allow species to adapt to their circumstances. In contrast to the species-based Darwinian model that underpins evolution, the connectionist framework for the human brain underpins learning. The fundamental structures that a species is determined by evolution, which is a progressive, unpredictable process at the species level, and by learning, which is a process of fine-tuning the structure of an individual to adapt to its environment (Harpham et al., 2004). Several stages of a product's life cycle can benefit from this approach, including concept, manufacturing, transportation, use, and disposal. Digital twins have a lot of potential to improve the future of development, design, and innovation, particularly when considering that fuzzy systems operate at a higher level of abstraction from human cognition, in contrast to the connectionist model-focused learning approach. Among the many approaches to AI in soft computing, three stand out: neurones, fuzzy systems, and genetic algorithms. Some of the most popular methods for simulating complex systems include neuronal networks and fuzzy logic. Two optimisation challenges must be solved to adapt neural networks with fuzzy systems:

Two types of optimisation: structural and quantitative. The initial stage in determining the best system architecture is structural optimisation (Castillo et al., 2000). This is a discrete combinatorial optimisation issue, and it is notoriously difficult to solve using the standard optimisation methods based on calculus. The optimal parameters of the system in a continuous parameter space are found by applying parametric optimisation after the system configuration has been identified. Traditional optimisation methods can tackle parametric optimisation, but they often stall at poor local optimal solutions. When it comes to learning and adaptation, neural and fuzzy models are perfect candidates for evolutionary computation. One of the most active fields of study when it comes to optimisation and adaptability is evolutionary computation. The idea behind the method is based on natural selection, which Darwin proposed. According to the Darwinian paradigm, it is not possible for an individual's learnt skills to be incorporated into its DNA and subsequently transmitted down through generations. Adaptability to a changing environment is enhanced by a combination of knowledge acquisition and evolution, as seen by developing neural networks (Hang et al., 2011). The impact of learning on evolution speeds up the process, which can manifest as either the Baldwin effect or Lamarckian evolution. The Lamarckian method permits the insertion of a person's acquired features into their genetic code, allowing their descendants to inherit those traits.

Despite the lack of biological support for the Lamarckian theory, it does a good job of describing how human civilisations have developed and can be helpful for EAs that analyse biological systems. However, it is physiologically reasonable to expect that learning will improve individuals' adaptation to their circumstances, leading to an increase in the likelihood that they will reproduce (the Baldwin effect). In order to find the best answer, optimisation techniques known as evolutionary algorithms (EAs) mimic the way evolution works in nature, drawing inspiration from Charles Evolution, according to Darwin. In the early stages of an algorithm's run, it generates individuals to be subjected to selection, mutation, and crossover in order to keep the strongest and get rid of the weakest. After a few iterations, it becomes capable of efficiently solving complicated optimisation problems and producing high-quality optimisation strategies. The great searching capacity and adaptable representations of EAs have led to their extensive application in solving a wide range of issues (Jamshidi et al., 2020), particularly in settings where non-convex or non-differentiable functions are present. Due to their potential as alternate approaches to NAS issue solving, EAs have recently garnered a lot of attention.

One interesting technique for optimising CNN architectures is the use of EAs, which have a great searching capacity and a flexible encoding strategy. Successful design of the network architecture's evolution strategy is the primary determinant of the evolution procedure, which is followed by the larger structure of evolutionary convolutional neural network models (ECNNs) (Roshani et al., 2021). Nevertheless, the majority of ECNN approaches disregard the relation between classification tasks in favour of enhancing the performance of previously identified CNN architectures. Whenever there is a shift in the learning environment, evolution must begin anew. Evolutionary transfer learning, which makes use of information from previously completed tasks to help solve the present task, shows promise for NAS challenges in this context. Better EAs have often made use of evolutionary transfer learning, a human-like method of learning, to address unrelated but related problems (Wang et al., 2021). Here is the outline of the article: Section 2 summarises previous research on using evolving neural models in this context; Section 3 describes the methods that will be used to optimise paths in simulation systems; Section 4 gives the experimental results and discusses their implications, with the study's concluding portion serving as its conclusion.

1.1 Contribution of this study

This research contributes to the area of remote sensing by suggesting a framework for semantic division of images from high-resolution remote sensing using a multi-scale convolutional deep neural network (MSCNN). The proposed method improves accuracy in identifying complicated land cover patterns and geographic details by integrating multi-scale feature extraction, in contrast to existing single-scale models. The study enhances segmentation efficiency in varied and heterogeneous situations by tackling issues like intra-class variation and inter-class similarity. The study also adds a solid approach to the field by integrating deep neural networks with optimised training strategies; this improves computing efficiency and generalisability. When tested against current state-of-the-art methods, the MSCNN model outperforms them all in terms of accuracy, recall, and precision. Applications for town planning, monitoring of the environment, and handling land use can be derived from these contributions, which lay a solid groundwork for intelligent interpretation of remote sensing data.

2 Related work

2.1 Graphical algorithms

Although graphical approaches offer a foundation for modelling, they typically lack search capabilities and must be supplemented by specialised search algorithms when applied to real-world situations. This is in contrast to traditional algorithms, which frequently struggle with challenging modelling. Both C-space and grid algorithms are examples of graphical algorithms; in the former, the obstacles are represented as polygons in movement space, and the latter searches for the shortest possible route by considering the beginning and ending points as well as a feasible straight line connecting every one of the polygons. While the c-space algorithm's ease of use and intuitiveness make it a good alternative to the spatial approach for finding the shortest path, the algorithm's inflexibility arises from the fact that changing the starting and ending points requires reconstructing the visible graph. Grid algorithms employ encoded rasters to depict the map; rasters with obstacles are called obstacle rasters, and rasters without obstacles are called free rasters. The free rasters are utilised as the foundation for path search.

2.2 Intelligent bionic algorithm

Natural disclosures can frequently be helpful when tackling path planning challenges involving complex dynamic environmental data. Algorithms like genetic algorithms, with the last section of the study acting as the conclusion, found in bionic research, are examples of intelligent bionic algorithms (Chappell et al., 2021). The ant colony algorithm mimics the foraging behaviour of real ant colonies through iterative processes. Although it can be enhanced with the addition of elite ants and other ways, it is computationally demanding and easily falls into local optimal settings, despite its good global optimisation capabilities, inherent parallelism, and computer implementation ease. Even while neural network algorithms are great for AI, they have not been able to successfully apply them to path planning since mathematical formulas can't capture the complexities and changes that occur in this domain 12 (Wu et al., 2023). Neural networks are great at learning, but they can't generalise well enough to be useful. Its resilience and high learning capabilities, however, have made its integration with other algorithms an exciting area of study in path planning.

2.3 Evolutionary algorithms for learning path generation

While searching for the best possible fit between a learner's attributes and a learning path, the generation of learning routes is seen as an optimisation problem within the evolutionary algorithm technique. Optimal strategies for ant colonies, genetic algorithms, particle swarms, etc, differential evolution are some of the most prominent examples of EAs. To handle the learning path's varied length, we used the variable-length genetic algorithm. Optimises a Java course's learning path using an integer-encoded evolutionary algorithm (Wu et al., 2022). In order to avoid creating solutions that are not viable, operators like partially mapped crossing and cycle crossover can be used to construct children from parent solutions. The next generation of solutions is selected using a simulated annealing mechanism, and new solutions are created using a discrete version of the hill-climbing method (Zhang et al., 2022a). Supposedly, the best results were obtained by combining cycle crossover with random initialisation and tournament selection. Forms a cooperative optimisation algorithm by merging both GA and ACO. When making adjustments, factors including the learning object as well as the learner's emotional condition, cognitive ability, and performance are taken into account.

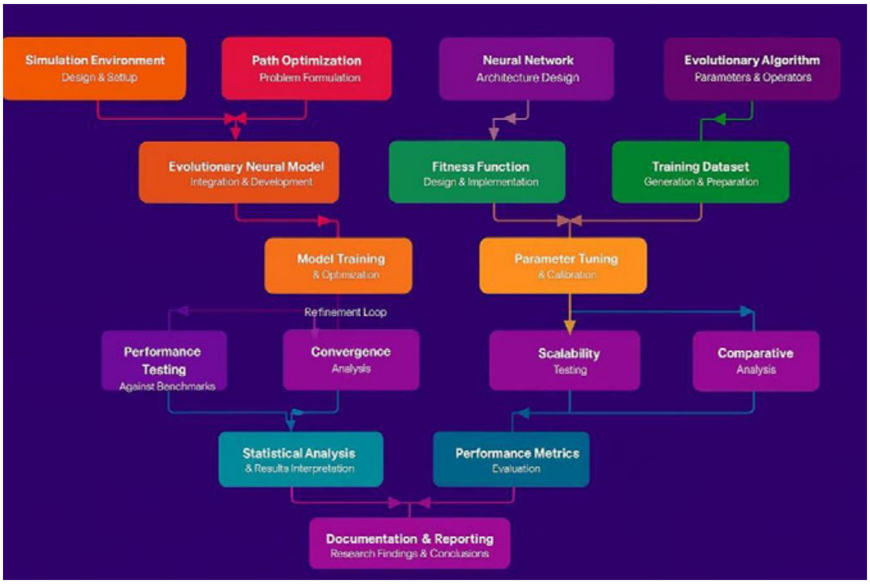
A multiobjective optimisation approach is used to recommend the learning pathway of MOOCs. Various learner metrics are taken into account, including enrolment, learning time, and popularity. Additionally, GA and ACO solve the model. Since the learning path is basically just a list of learning resources, creating one is a challenging combinatorial optimisation issue that is also NP-hard. In order to keep the new solutions feasible, EAs like GA's crossover operator, DE's mutation operator, and PSO's velocity update are explicitly developed for this kind of situation (Zhang et al., 2022b). But, with the right solution representation scheme, the continuous variation of these genetic algorithms ensures that a new solution is feasible. We will talk about differential evolution, a class of efficient continuous EAs (Chappell et al., 2021), and how it represents solutions in the next section. Because of their remarkable capacity to learn and identify patterns in data, neural networks are beneficial for solving complicated problems. Neural networks can accomplish sophisticated function approximations, prediction making, process optimisation, and anomaly detection through training on massive datasets. Their adaptability and ability to learn make them useful in many contexts, including pattern identification, prediction, optimisation, and the management of intricate systems. For successful complex system modelling, there are a plethora of options already accessible (Iqbal et al., 2023a). Using fractional order linear equations, Iqbal et al. examined a model of a computer virus pandemic. Additionally, we analysed the Newell-Whitehead-Segel equation in its stochastic form. To model the distribution of temperatures in heating tanks, we used the Runge-Kutta method, the exponential matrix algorithm, and the differential transformation algorithm (Iqbal et al., 2023b). Presents a new approach for efficiently and accurately solving quantum mechanical models, the conformable Shehu transform decomposing method (CSTDM).

Additionally, there are a plethora of published papers that make use of neural networks to comprehend and analyse intricate systems. Provides an overview of the numerical results obtained by employing stochastic scaled conjugate slope neural networks and fractional derivatives in the FKFS model (Haritha et al., 2023). Using LMBNNs as its foundation, propagation model using artificial neural networks trained with Levenberg-Marquardt backpropagation. An approach that makes use of artificial neural networks that have been optimised utilising particle swarm optimisation and active-set methods.

3 Materials and methods

The whole process flow of an ENN architecture is shown in Figure 1. The first steps include configuring the simulation environment, optimising the path, designing the architecture of the neural network, and setting the evolutionary process's parameters. After that, you'll need to prepare your training dataset, build your fitness functions, and incorporate an evolutionary neural model. Model training, parameter adjustment, and assessment phases, like performance testing, scalability testing, convergence analysis, and comparative analysis, follow. The last step in documenting and reporting study findings and conclusions is to undertake statistical analysis along with achievement measures.

Figure 1 Diagrammatic representation of the steps involved in creating and assessing an evolutionary neural network model, from initialisation of the simulation to preparation of the dataset and training of the model, and finally to reporting on the results after testing and parameter tuning (see online version for colours)



3.1 *Path optimisation*

In the Critic's first layer, the input layer, two vectors are created: one includes data about the learner's present state and data about the Actor's present learning pathway output (Ding et al., 2025):

$$Z_t = conca(st, at) \tag{1}$$

Figure 2 shows the internal framework of the second layer, long short-term memory (LSTM). This paper uses the LSTM layer to extract time series information because learning is a time- dependent activity. The capacity to remember what has been learnt and how it influenced subsequent decisions is possessed by this layer. Both the learner's current status and their learning trajectory are inputs to the LSTM layer that corresponds to it:

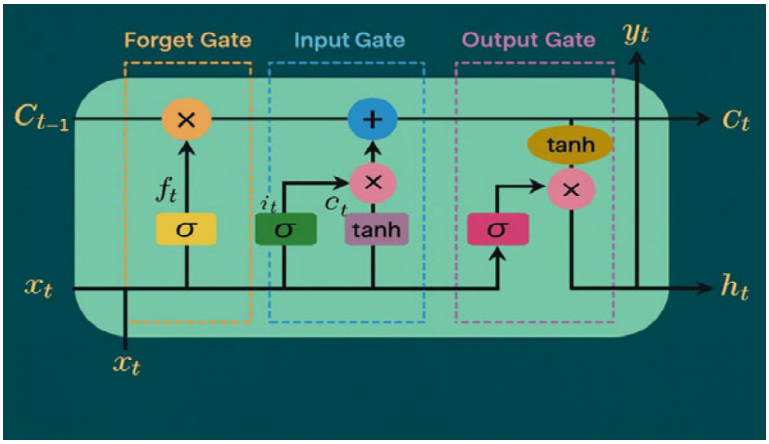
$$ht = LSTM\left(zt, ht - 1\right) \tag{2}$$

Equation (12) represents the LSTM layer's hidden state, which integrates the learner's present-state learning trajectory with their prior learning experience. A fully connected layer makes up the third level. In this research, we investigate and develop A fully-connected three-layer network that can take the LSTM layer's output data and produce an index for the evaluation of state-path pairs, or Q value:

$$Q(st, at) = Wq \cdot ht + bq \tag{3}$$

In equation (13), the weight matrix, the bias term, and the outcome of LSTM, the final equation is composed of the evaluation Q value of that specific state-path pair and the output layer is the final layer. In order to determine how beneficial the present learning trajectory is, the critical component looks at the Q value. Given the present situation, the Q number indicates the likelihood of selecting this learning path. In this layer, the Q value is represented as a scalar.

Figure 2 Brain diagram of an LSTM (see online version for colours)



3.2 Neural network

Figure 3 displays an EFO-DNN technique block diagram for optimising meta-heuristic architecture; this diagram is relevant to the neurological design search for the DNN model. The metaheuristic architectural optimisation process is addressed by this approach, which is implemented by. Using their method, neural architecture search can make use of continuous search spaces and metaheuristic optimisation techniques. This study optimised the DNN model's neural architecture using objective functions implemented with the mean-value search policy (Alagoz et al., 2025).

Figure 3 depicts a computational approach that optimises the architecture by combining the back propagation technique with the EFO algorithm. Parametric learning makes use of the backpropagation algorithm to train models, and structural learning makes use of the EFO algorithm to optimise architectures. When it comes to managing DNN training and architectural optimisation, both algorithms are complex at work simultaneously. To train the DNN, the EFO algorithm first creates descriptions of potential architectures, and then backpropagation is used. In order to determine whether the DNN architecture is suitable, the objective function compares it to established neuroevolutionary goals. This work optimises the architecture to produce nearly the best generalising DNN framework for the cooling demands of the dataset by using an objective function that delivers solely values for generalisation performance to the EFO algorithm. One way to adapt continuous-space metaheuristic approaches to a discrete search space of acceptable architecture descriptions is by using the block of integerisation and neural repair. In Figure 3, we can see an example of processing that uses the EFO's continuous candidate outcome vectors to get valid descriptions of the neural architecture.

Figure 3 An optimisation technique for data-driven DNN architecture and its primary parts (see online version for colours)

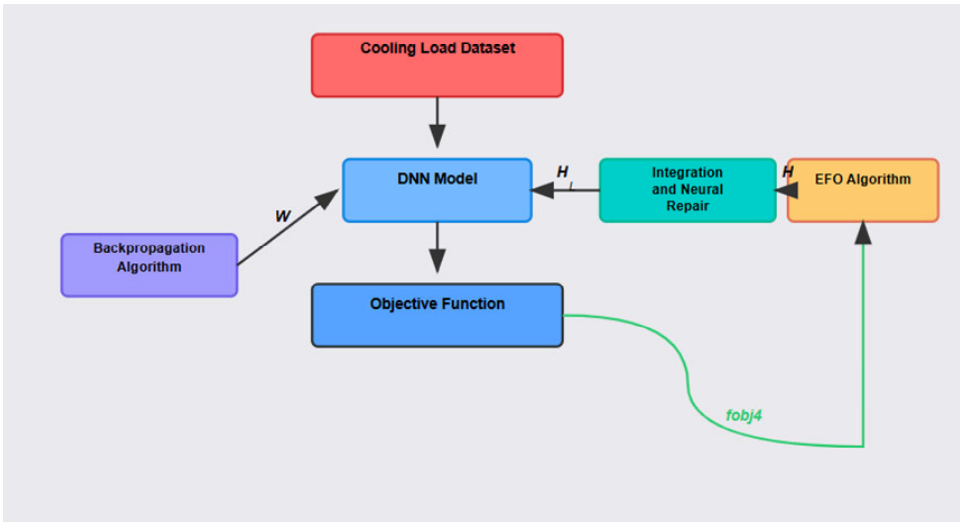
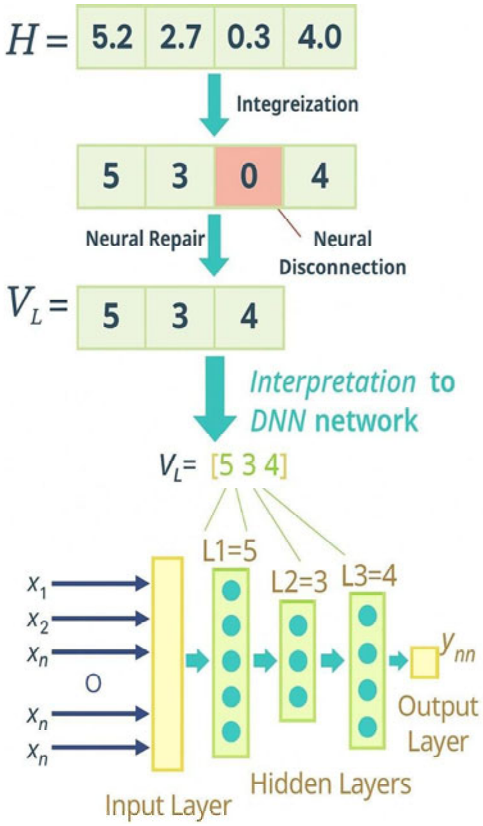


Figure 4 Justification for the valid architecture specification scheme proposed by Simsek et al. (see online version for colours)



In this case, the EFO technique can give continuous vectors that describe potential architectures. A legitimate description of a DNN architecture, however, requires that the neurons in the buried layers must contain integers that do not equal zero. Pre-processing possible architectural descriptions yielded valid neural design descriptor vectors. To begin, we round all of the elements to the nearest integer in what is called the integerisation state. Step two, ‘neural repair,’ involves removing all zeros from the vector. There is a range of neural depths that can be achieved through different neural restoration techniques. Thanks to this adaptability of neural depth, DNNs can have the number of hidden layers modified to achieve optimal model complexity. After that is done, the vector elements will show you how many neurons are in each layer of these DNN networks. The process of preparing the vector to provide an appropriate architectural description is illustrated in Figure 4. With five neurons in the most significant hidden portion, three in the middle hidden layer, and eight at the bottom, this example vector shows a three-layer architecture of the neural network. A reduction of 0.3 brings the network depth down to 3 hidden layers. The optimisation of deep neural network design is made easier by using the architecture coding format, which specifies the number of neurons for hidden layers.

3.3 Evolutionary algorithm

- Methods for reducing systemic risk are detailed in this section of the article. A vector of benchmarks indicating the appropriate levels of reserves for the corporations to maintain can be used as a solution to this issue. This paper compared three different methodologies (Michalak et al., 2024).
- The thresholds have been optimised through evolution (Figure 5). Here, the evolutionary algorithm minimises systemic risk by operating on a search space that is identical to the decision region of the corresponding problem. This is the standard method for solving graph protection problems in the literature, hence it serves as a comparison point for MOEA/D-NN. This paper compares MOEA/D-NN to four other evolutionary processes, such as MOEA/D, NSGA-II, and NSGA-III.
- SPEA2. These algorithms were developed by Deb and Jain in 2014 and 2010, respectively, and are based on decomposition-based EAs. Algorithm 1 is one of these EAs that incorporates a local search; in the rest of the article, algorithms that employ a local search are denoted by an additional '+LS' after their names. The NSGA-II method, for instance, can be enhanced with the local search to form NSGA-II+LS.
- Deciding on thresholds for future problem cases by training an automatic learning model with data obtained while running the optimisation algorithm on a specific problem instance (Figure 6). The results discussion uses the notation 'NN only' to describe this method. In this study, the proposed strategy is applied utilising neural networks that have been trained using the modified conjugate gradient (SCG) technique
- Changing the parameters of ML models over time and then utilising those models to select each node's threshold in the optimisation run (Figure 7). The authors of this article suggest a novel method, MOEA/D-NN (in which NN stands for 'neural net').

Figure 5 Optimising the thresholds through evolution. a vector of node thresholds represents each population solution. the method is defined in great depth (see online version for colours)

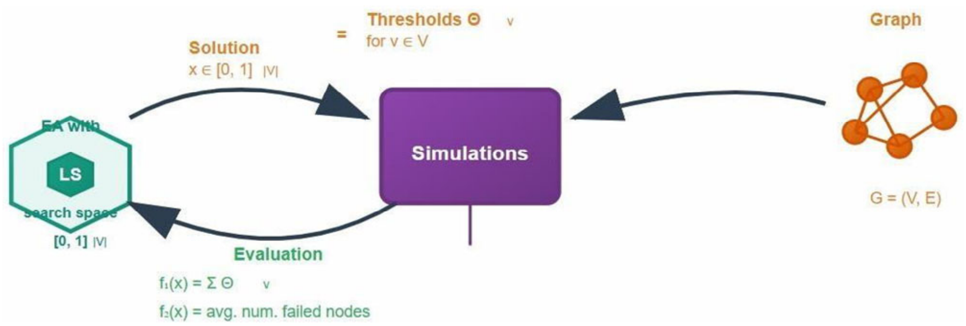
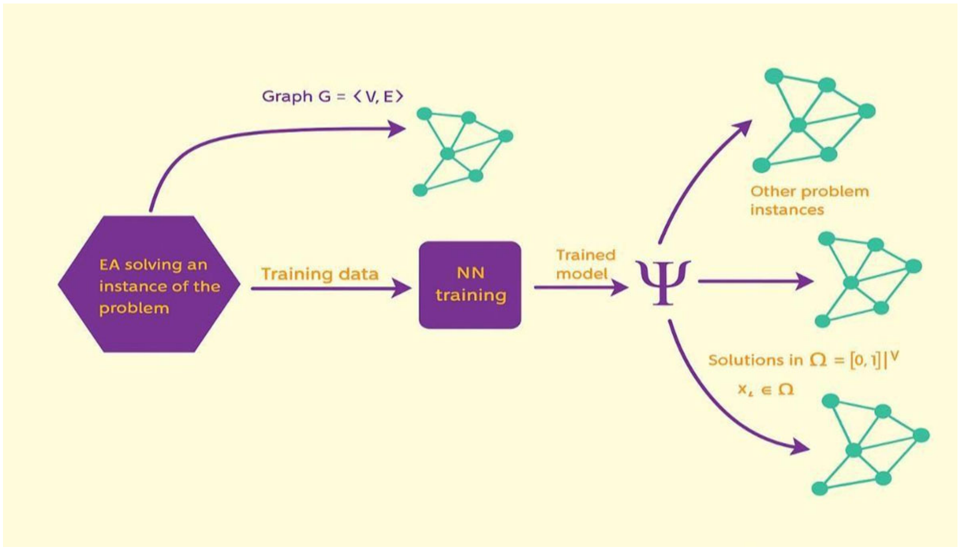
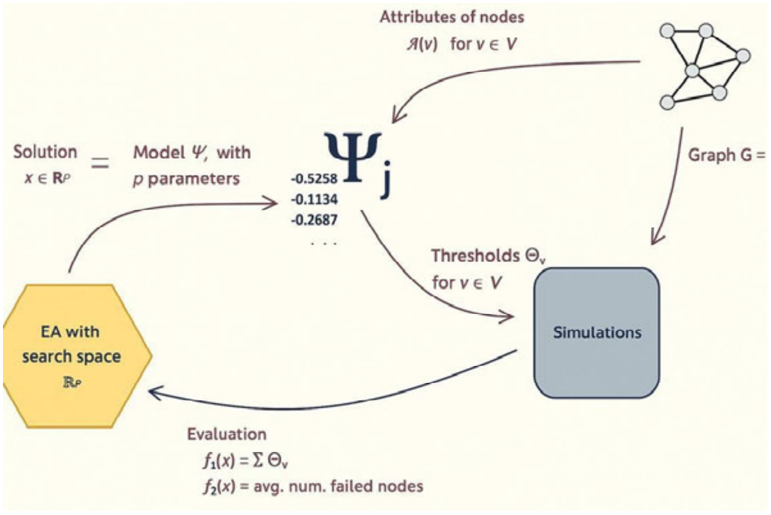


Figure 6 To solve new cases, it is necessary to train an algorithm for machine learning utilising the data obtained while running the optimisation method (see online version for colours)



Notes: Whenever an evolutionary algorithm discovers a better answer, it creates samples of training data. it is the training dataset that is used to train the model. in order to solve new problem instances, we feed the ml model all of the attributes and let it determine the thresholds. the method is defined in great depth.

Figure 7 Genetic algorithm for optimising weights in neural networks (see online version for colours)



The neural network's p weights are contained in each solution vector in the population. The evaluation of a solution is achieved by simulating the propagation of information using a vector of thresholds generated by the neural network without weights taken from *x*. The method is defined in great depth or more in-depth analysis, we employ the following methods: the Rand function takes a probability distribution *P* as input and uses it to draw an element. As an example, the uniform probability distribution of the range [0, 1] is drawn by Rand (U[0, 1]). Starting with randomly chosen nodes assigned to reach the failing state, the program uses the solution's thresholds to mimic the graph's bankruptcy spread according to the failure process outlined in. Returns, at the end of the simulations, a vector with the failure percentages of all the nodes in *V* and an average of those percentages.

3.4 Evolutionary neural model

The growth field is a space with several dimensions that contains agent property codes. In this space, the agent's attributes are represented by a property coding $ak \in RD$, where D serves as the dimension of the property code and k is the agent index value. These codes change over time, as described in reference (Alagoz et al., 2022). Search agents, defined by the property they own code, are individuals that represent individuals in a solution atmosphere to solve optimisation issues and can only behave within that environment. The suitability of the agent to its natural environment is typically used as a selection criterion for its chances of survival. Hence, the goal function (*Xk*) evaluates how well agent property codes reflect an ideal solution to the environment. The value of (*Xk*) represents the field measurement of the real estate codes, and the field quantity has been extensively utilised for repositioning agent property codes to fit the evolution field. As a result, agent selection and evolution take the field value into account. Essentially, goals in the evolving field assign an appropriateness value to the search agent's property code. An evolutionary field can be described by an objective function and a minimum set of asset codes that can be used to develop strategies for the field-based evolution of agent characteristics; this set is closed and has the form (*ak*, *Fk*). Figure 8 shows the connection between the solution environment agents and the field of property code evolution. The vector *bk* represents the *j*th property of the search agent *k*, where the constituent parts of the vector *xq*, make up the characteristic code.

$$Xk = [xk, 1\ xk, 2, \ldots xk, D] \tag{4}$$

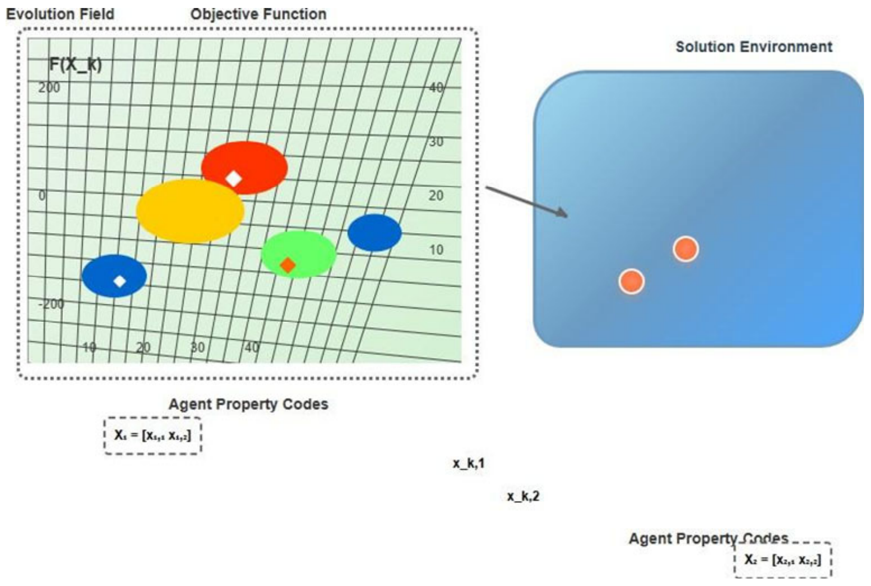
A system of Cartesian coordinates is extensively used to govern the geometrical evolution methods of property codes, which in turn affect agent properties. Because of this, distance measures are a legitimate way to convey the evolutionary relationships between the agent behaviour codes. Therefore, the space of metrics (*Xk*,) is defined by property codes in Linear coordinates, where the function d stands for a metric that ensures: The metric *d*(*Yi*, *Yb*) expresses the dissimilarity of agents' attributes in the given measure space.is greater than or equal to zero. In the solution environment, agents *i* and *j* are considered to be identical agents since the equality state (*Yi*, *Xb*) = 0. The values of (*Xi*, *Xb*) help determine how much the agent's property code has changed over time and allow for the expression of an indication for agent property differentiation. Agent attributes do not impose any priority, hence

$$d(Yi, Yb) = d(Xb, Xi) \tag{5}$$

Triangle inequality satisfies agent attributes and permits the definition of geometrical evolution strategies, as shown by the disparity $d(Yi, Yb) \leq d(Yi, Yk) + d(Yk, Yb)$. Avoiding code space deflection is essential for the shortest evolutionary path. Agent property changes are the outcome of relocating the evolution field's code for the agent's properties, which is also known as agent evolution. A distance measurement in the advancement field can be used to express the amount of evolution. Assume for the sake of argument that in instance *n*, the agent k's property code (*X*[*n*]) changes to *Xk*[n + 1] at instance *n* + 1. Seasonal evolution that meets is a good proxy for the amount of dynamic evolution at this instance.

$$(Xk[n], Xk[n+1] \geq 0 \tag{6}$$

Figure 8 The optimisation problem's solution space and the evolution field's property codes are shown schematically (see online version for colours)



Agent K's property code indicates the pace of seasonal evolution.

$$E_r\left(X_k[n]\right) = \frac{d\left(X_k[n], X_k[n+]\right)}{\left\|X_k[n]\right\|} \tag{7}$$

where the norm of the *Y*[*n*] vector is represented by the operator $// ak[n]$. Agents operating in a solution environment may evolve some traits that help them survive, while others may hinder their chances of survival. The field value (*Xk*[*n*]) and evolutionary energy density are both increased as the value of *Fk*[*n*] at code *Xk* grows larger. The advantageous evolution becomes apparent when (*Xk*[*n*+1]) < *F*(*Xk*[*n*]). Then, in order to minimise agent k's field value (its evolutionary energy density), the Lyapunov stability provides the basis for writing the negative derivative requirement as.

$$\Delta Fk[n] = (Xk[n+1]) - F(Xk[n]) < 0 \tag{8}$$

Natural selection processes may not necessarily need to know or be cognisant of the best course of action over the long term of evolution. Natural selection operates as a Markovian process, and the present value of the field provides an expression for beneficial transitions of agent attributes, i.e., seasonal advantageous development of code for the property.

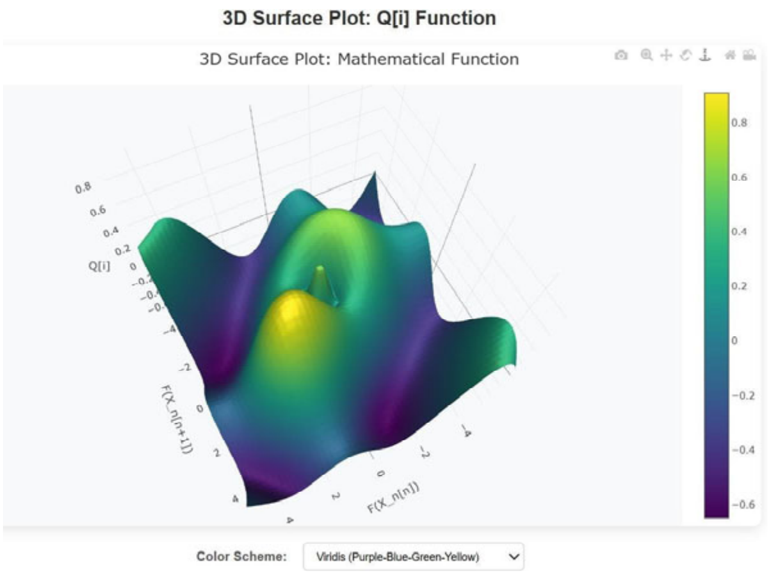
$$X_k[n+1] = \begin{Bmatrix} X_k(n+) & F\left(X_k[n+1]\right) - F\left(X_k[n]\right) < 0 \\ X_k[n] & F\left(X_k[n-1]\right) - F\left(X_k[n] \geq 0\right) \end{Bmatrix} \tag{9}$$

Depletion of evolutionary potential is a good proxy for the quality of evolution; this proxy can be represented as.

$$Q[n] = \frac{F\left(X_k[n+1] - F\left(X_k[n]\right)\right)}{\left|F\left(X_k[n-1]\right)\right| + \left|F\left(X_k[n]\right)\right|'} \tag{10}$$

The value of the seasonal grade indicator [*n*] could range from –1 to 1. A rating of –1 denotes poor quality in seasonal evolution, while a value of +1 shows exceptional quality. Figure 9 displays the values of the seasonal quantity index [*n*] for the randomly selected digits *F*(*Xk*[*n*+1]) or *F*(*Xk*[*n*]) in the set of values from [–5, 5].

Figure 9 The seasonal grade index [*n*] is decided using the consequent values of *F*(*Xk*[*n*+1]) or *F*(*Xk*[*n*]) (see online version for colours)



Color Scheme: Viridis (Purple-Blue-Green-Yellow)

4 Result analysis

In Figure 10, we can see the training-related reduction in the model that was the subject of this paper's convergence investigation. The loss value is shown vertically, and the total amount of training rounds is displayed horizontally.

The actor and critic losses are examined separately in this research as a result of the actor-critic structure of the model. The blue line depicts changes in the actor's loss value, while the yellow line shows changes in the critic's loss value. The model's loss is effectively decreased by changing its parameters in training, since the loss curves for either the actor and the critic level out in the second stage (60–100 rounds) after a steep decline in the first stage (0–40 rounds). This implies that the model is able to learn and converge in this stage. Given how near the model's final loss value is to 0, overfitting is a distinct possibility.

Figure 10 Analysis of convergence (see online version for colours)

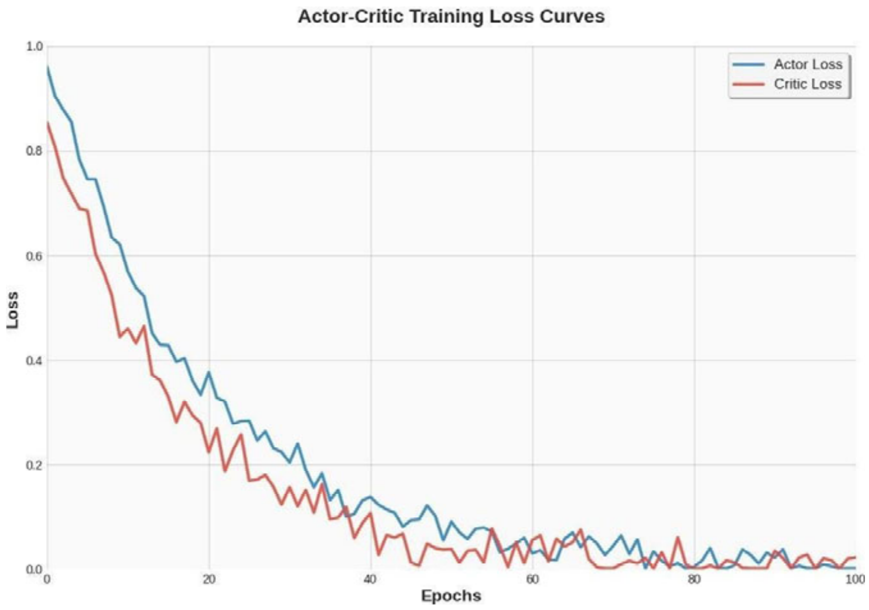


Figure 11 shows how the loss value convergence compares to the standard models that perform the same task, including Q-learning and DQN, and the model that is being studied in this article. The blue line shows the loss of significance of the study's proposed model with time, whereas both the green and red lines represent the changes in loss in Q-learning as DQN, respectively, for this job. This paper's analysis of the actor-critic model reveals that it outperforms Q-learning and DQN in terms of learning route optimisation stability and efficiency, thanks to its much quicker convergence time and more refined loss reduction method.

Some underfitting may occur if there are provisions in the multiobjective function to simplify the model. The multiobjective function, which takes into account a loss function of the training dataset, might be affected by extrapolation from training loss while optimising the architecture, somewhat impairing generalisation performance in the building optimisation job. Because the generalisation error may be further minimised with a single objective that solely concerns test loss in architectural optimisation, it can produce nearly the best generalising model. These observations are validated by the training and exam performances shown in Table 1. Our working hypothesis is that adjusting the neural architecture so that the test dataset's loss function provides nearly optimal generalisation is possible, given a complicated enough model. Our experiments on the cooling load dataset showed that, out of all the objectives examined, the one that takes the test dataset's mean square error into account had the best chance of enhancing the DNN model's generalisation property.

Figure 11 Comparison of model errors (see online version for colours)

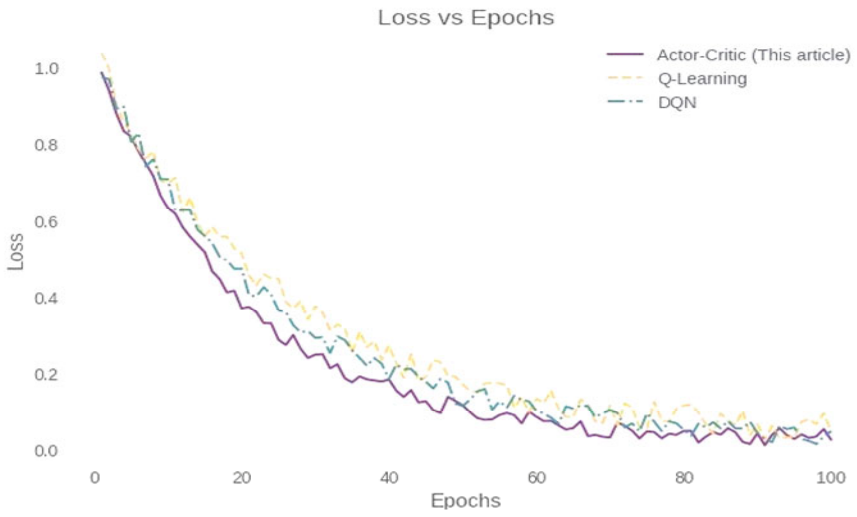


Table 1 DNN model complexity points confirmed by RMSE outputs classify models according to their use in training and testing.

Objective functions	*Average training performance*	*Average test performance*	*Sum of average training and test performances*	*Explanations*
$fobj_1$	0.9251	1.4794	2.4045	Underfitting propensity due to the model reduction term
$fobj_2$	1.0854	1.3215	2.4069	Underfitting propensity due to the model reduction term
$fobj_3$	0.4143	0.9178	1.3321	Enhanced
$fobj_4$	0.3795	0.8648	1.2443	Reduced

Figure 12 provides a better illustration of the objective product designs by describing how the objectives of the model complexity optimisation characteristics of the goal functions are shaped by the tasks. The picture depicts three separate types of these goals. The curves depict three different loss functions: one for single-objective testing, one for multiobjective training and testing, and one for multiobjective training and testing with a model reduction (MR) component. Attempts to reduce objective functions lead to architectural design occurring in the design zone, which is the grey shaded region in the illustration, where objective curves intersect. Since the model reducing term shifts the minimum point under this sort of objective function towards low-complexity models, the underfitting modelling region tends to contain models with a minimum of the curve labelled 3. As the total number of layers increases, the value of the model's reducing term also climbs, bringing the function's minimum closer to the underfitting zone. Because the training and test losses move the minimum point of the multiobjective function closer to the overfitting region, the model with label two as its minimum is typically one that is near the overfitting simulation region. There may be a dip in value before it begins to climb again. Suppose we take the minimum of a curve labelled 3, which is the smallest of the generalisation errors. In that case, we have a simple model that is quite similar to the best generalising model. The model achieves the lowest possible test loss, which is the reason behind this. Therefore, the purpose lets you get to the ideal model complexity that practically guarantees top notch generalisation performance. The cooling load assessment problem's Table 2 confirms these findings.

Figure 12 This graphic depicts the features of three different kinds of objective functions, as well as design zones (grey zones) for optimising neural architecture (see online version for colours)

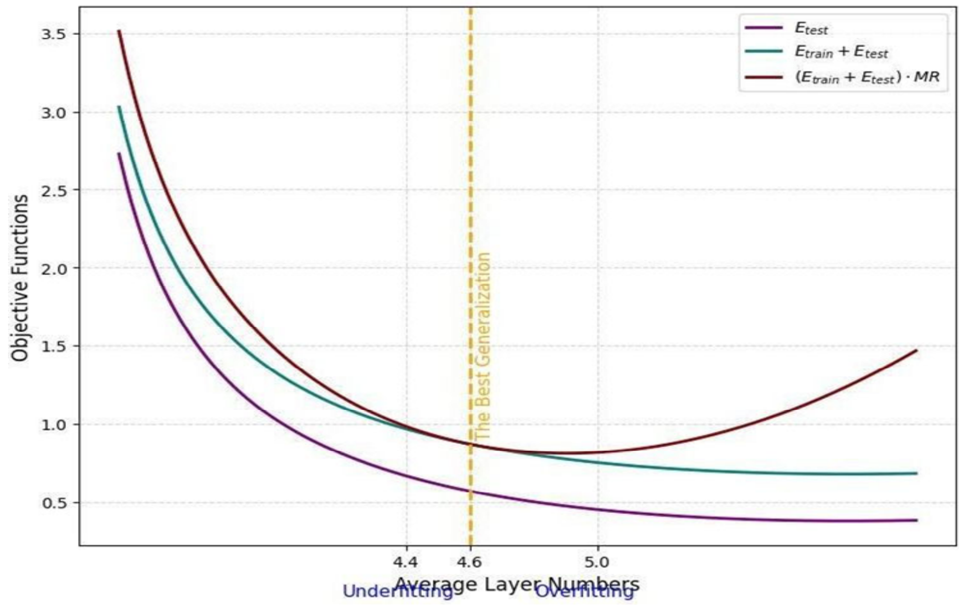


Table 2 Averaging the RMSE and the number of layers in each group of models

Objective functions	*Characteristic type*	*Average layer numbers*	*Average RMSE for test dataset (generalisation error)*
$fobj_1$	$(E_{train} + E_{test}).MR$	4.4	1.47
$fobj_2$	$(E_{train} + E_{test}).MR$	4.5	1.32
$fobj_3$	$(E_{train} + E_{test}).MR$	5	0.91
$fobj_4$	E_{test}	4.6	0.86

What follows is a discussion of the studies that compared optimisation without machine learning models to systemic risk minimisation using machine learning models. Figure 13 shows the overall experimental procedure. Above the dark blue line in the picture, in the experimental preparation phase, we trained machine learning models. We fine-tuned the parameters associated with the tested procedures – thirty examples of optimisation problems meant for 'training' machine learning models. To prevent parameters and model training from being overfit to the tested issue cases, the tested methods were then applied to a distinct set of 'testing' optimisation problem examples with $|V|$ = nodes. With the initial shock size set to and the values listed in Tab. 3, all examples were based upon REDS graphs. We opted for the REDS network model because its topology is intriguing;

it mimics real-world networks with densely linked groupings of nodes as well as less populated regions. R determines whether connections may be formed remotely, E specifies how many connections a node can establish, and S dictates the propensity to cluster (due to the lowering of edge costs as a function of the number of shared neighbours), which are all parameters that control the graph generation process. This paves the way for evaluating the produced ML models on graphs that differ in properties from the training ones. The generation of a REDS graph involves randomly placing |V| nodes on the unit square. The next step is to slowly increase the number of edges connecting randomly chosen pairs of nodes that are no more than R distance apart. At the outset, every node has an amount of social energy E, and the cost of creating an edge to another node is the same as the distance D between them. Having shared neighbours reduces the price of the edge between two nodes by a certain amount, where S is a parameter that regulates how powerful the synergistic effect is. Constructing connections between nodes with shared neighbours is made more cost-effective by this cost discounting method. All of this is based on actual experiences: people are more likely to form new friendships through shared acquaintances, and Collaborators facilitate the formation of new commercial ties. This study established the values for the REDS model's parameters. The total number of nodes |V| and the parameters R, E, and S are detailed in Table 3 for each instance used in the testing. The average node degree is calculated from the generated graphs and shown in the last column of Table 1, even though it is not a variable that can be modified in the REDS model. When plotting with R, E, and S, this number is required. With the parameters shown in Tab. 3, we ensured that the mean quality of the nodes was constant across all graph sizes, keeping the conditions for the occurrence of defaults stable. With the parameters listed in Table 3, Figure 13 shows a REDS graph with |V| = 1000.

Table 3 Experimental parameters for REDS graphs

$\|V\|$	R	E	S	
1,000	0.1000	0.5	1.0000	28.86
1,250	0.0894	0.5	0.8944	29.07
1,500	0.0817	0.5	0.8165	29.22
1,750	0.0756	0.5	0.7559	29.44
2,000	0.0707	0.5	0.7071	29.56
2,250	0.0667	0.5	0.6667	29.64

We offer here both the actual and complicated-valued neuron modes that comprise the PWM-EFO method for NOx estimation. The input data from the sensors array, which were originally all positive-valued, had to be given a negative sign to activate the intricate-valued mode. Dendritic bifurcation of PWM generates complex numbers in line with equation (23), and all input values are transformed into low real numbers as a consequence. Complex magnitudes are no match for the PWM neuron. In order to switch to the mode of complex-valued neurones, the training dataset was organised as ($-x1$, −x2, $-x3$, $-x4$, $-x5$, $-x6$, $-x7$, $-x8$, yd). In the previous section, it functioned in the real-valued PWM neuron mode since the training dataset was structured as ($x1$, 2, x 3, $x4$, $x5$, $x6$, $x7$, $x8$, b.

Figure 13 The settings were used to construct a reds architecture with 1,000 nodes, as an example (see online version for colours)

Figure 14 Values of the test dataset's real-valued and complex-valued PWM neurones are estimated (see online version for colours)

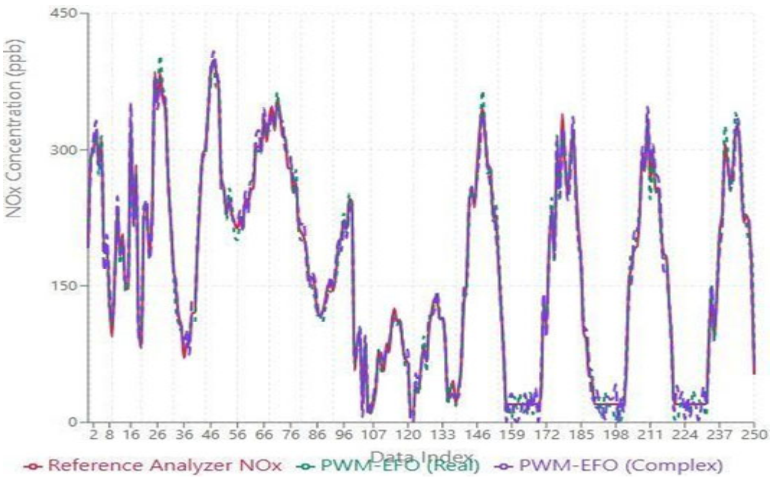
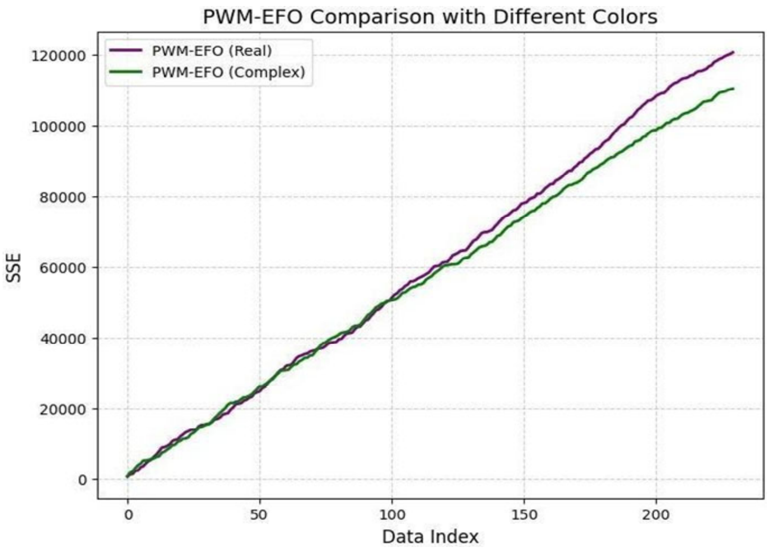


Figure 14 displays the estimated modes of different PWM neurons for your test dataset. Estimation performance indices are shown in Table 4. The results demonstrate that the real-valued method operation slightly enhances the estimate performance when contrasted with the complex-valued mode. The results could be explained by the dataset's real-valued relational nature and the absence of computing requirements in the complicated domain.

Table 4 Model performance metrics, including R^2, MSE, MAE, and MRE (mean relative error), were evaluated throughout testing

Estimation models	*MSE*	*MAE*	*MRE*	*R^2*
PWM-EFO (real)	725.52	20.8	0.16	0.89
PWM-EFO (complex)	842.46	22.9	0.18	0.87

Figure 15 An increase in the sum of square errors (SSE) in estimations performed by PWM neurones operating in real and complex modes (see online version for colours)



After using both real-valued and complex-valued PWM neurons, 241 hourly measure predictions were performed, and the change in the total amount of the squared error (SSE) is shown in Figure 15. Even while it performs better in the complicated value mode up past the 100th measurement, its SSE performance begins to fall at the 190th measurement. The results demonstrate that when it comes to the real-valued form, the training sample generalisation is better. NOx calibration problem, and that real-valued neurones have better long-term SSE performance.

5 Conclusions

In order to optimise paths using simulations, this study introduces a new evolutionary neuronal model that successfully combines learning from neural networks with evolutionary optimisation. The model outperforms traditional learning models in terms of convergence speed, stability, and generalisation. The system guarantees strong performance in complicated and ever-changing contexts by resolving overfitting and underfitting with the help of evolutionary field optimisation and actor-critic reinforcement learning. The outcomes validate the efficacy and dependability of evolutionary neural models in solving engineering system optimisation, risk management, and path planning problems. This study paves the way for the suggested framework to be used in various real-world applications, including digital twin systems, intelligent transportation, and Industry 4.0, by enhancing the synergy between neural networks and evolutionary computation.

Declarations

All authors declare that they have no conflicts of interest.

Fund information

Project name: Research on power simulation system in a cloud-based multi-core environment; Project source: Company-level project of China Southern Power Grid Energy Development Research Institute Co., Ltd.(No.EDRI-PS-ZLYJ-2024-103)

References

Alagoz, B.B., Keles, C., Ates, A. et al. (2025) 'Optimal deep neural network architecture design with improved generalisation for data-driven cooling load estimation problem', *Neural Comput. Appl.*, 2 May, pp.1–20.

Alagoz, B.B., Simsek, O.I., Ari, D., Tepljakov, A., Petlenkov, E. and Alimohammadi, H. (2022) 'An evolutionary field theorem: evolutionary field optimisation in training of power-weighted multiplicative neurons for nitrogen oxides-sensitive electronic nose applications', *Sensors*, 18 May, Vol. 22, No. 10, p.3836.

Castillo, P.A., Merelo, J.J., Rivas, V., Romero, G. and Prieto, A. (2000) 'G-Prop: global optimisation of multilayer perceptrons using GAs', *Neurocomputing*, 1 November, Vol. 35, Nos. 1–4, pp.149–163.

Chao, Y., Augenstein, P., Roennau, A., Dillmann, R. and Xiong, Z. (2023) 'Brain-inspired path planning algorithms for drones', *Front. Neurorobot.*, 3 March, Vol. 17, p.1111861.

Chappell, D., Crofts, J.J., Richter, M. and Tanner, G. (2021) 'A direction preserving discretization for computing phase-space densities', *SIAM J. Sci. Comput.*, Vol. 43, No. 4, pp.B884–B906.

Ding, X., Ding, H., Zhou, F. et al. (2025) 'Personalised learning path optimisation based on enhanced deep neural network: higher education teaching model integrating learner behaviour and cognitive style', *Discov. Artif. Intell.*, December, Vol. 5, No. 1, pp.1–9.

Fuller, A., Fan, Z., Day, C. and Barlow, C. (2020) 'Digital twin: enabling technologies, challenges and open research', *IEEE Access*, 8 May, Vol. 8, pp.108952–108971.

Hang, B. et al. (2011) 'Evolutionary computation and its applications in neural and fuzzy systems', *Appl. Comput. Intell. Soft Comput.*, 8 July, Vol. 13, No. 1, p.938240.

Haritha, K., Shailesh, S., Judy, M.V. et al. (2023) 'A novel neural network model with distributed evolutionary approach for big data classification', *Sci. Rep.*, Vol. 13, No. 3, p.11052.

Harpham, C., Dawson, C.W. and Brown, M.R. (2004) 'A review of genetic algorithms applied to training radial basis function networks', *Neural Comput. Appl.*, Vol. 13, No. 3, pp.193–201.

Iqbal, M.S. et al. (2023a) 'Numerical simulations of nonlinear stochastic Newell-Whitehead-Segel equation and its measurable properties', *J. Comput. Appl. Math.*, 15 January, Vol. 418, p.114618.

Iqbal, Z. et al. (2023b) 'A finite difference scheme to solve a fractional order epidemic model of computer virus', *AIMS Math.*, Vol. 8, No. 11, pp.2337–2359.

Jamshidi, M., Lalbakhsh, A., Lotfi, S., Siahkamari, H., Mohamadzade, B. and Jalilian, J. (2020) 'A neuro-based approach to designing a Wilkinson power divider', *Int. J. Microw. Comput. Aided Eng.*, March, Vol. 30, No. 3, p.e22091.

Lo, C., Chen, C. and Zhong, R.Y. (2021) 'A review of digital twin in product design and development', *Adv. Eng. Informatics*, 1 April, Vol. 48, p.101297.

Michalak, K. (2024) 'Evolutionary algorithm with a regression model for multiobjective minimisation of systemic risk in financial systems', *Soft Comput.*, March, Vol. 28, No. 5, pp.3921–3939.

Pourmostaghimi, V., Heidari, F., Khalilpourazary, S. and Qazani, M.R.C. (2023) 'Application of evolutionary optimisation techniques in reverse engineering of helical gears: an applied study', *Axioms*, 1 March, Vol. 12, No. 3, p.252.

Roshani, S., Jamshidi, M.B., Mohebi, F. and Roshani, S. (2021) 'Design and modelling of a compact power divider with squared resonators using artificial intelligence', *Wirel. Pers. Commun.*, April, Vol. 117, No. 3, pp.2085–2096.

Wang, Z., Lu, D., Wang, H., Liu, T. and Li, P. (2021) 'Evolutionary convolutional neural network optimisation with cross-tasks transfer strategy', *Electronics*, 2 August, Vol. 10, No. 15, p.1857.

Wu, J., Li, H., Li, B., Zheng, X. and Zhang, D. (2023) 'Optimisation of robotic path planning and navigation point configuration based on convolutional neural networks', *Front. Neurorobot.*, 4 June, Vol. 18, p.1406658.

Wu, M.Y., Ke, C.K. and Lai, S.C. (2022) 'Optimizing the routing of urban logistics by context-based social network and multi- criteria decision analysis', *Symmetry*, 1 September, Vol. 14, No. 9, p.1811.

Zhang, J., Zhang, J., Zhang, Q. and Wei, X. (2022a) 'Obstacle avoidance path planning of space robot based on improved particle swarm optimisation', *Symmetry*, 5 May, Vol. 14, No. 5, p.938.

Zhang, Y-W., Xiao, Q., Song, Y-L. and Chen, M-M. (2022b) 'Learning path optimisation based on multi-attribute matching and variable length continuous representation', *Symmetry*, 9 November, Vol. 14, No. 11, p.2360.