Contents lists available at GrowingScience

Decision Science Letters

homepage: www.GrowingScience.com/dsl

A multilayer feed-forward neural network (MLFNN) for the resource-constrained project scheduling problem (RCPSP)

Amir Golab[a*], Ehsan Sedgh Gooya[a], Ayman Al Falou[a] and Mikael Cabon[a]

[a]*ISEN Yncréa Ouest, Brest, France*

C H R O N I C L E

Article history:
Received April 20, 2022
Received in revised format:
June 25, 2022
Accepted July 22 2022
Available online
July 22, 2022

Keywords:
Project scheduling
Project management
Artificial neural network
Priority rules
RCPSP
Resource constraint

A B S T R A C T

Project management has a fundamental role in national development, industrial development, and economic growth. Schedule management is also one of the knowledge areas of project management, which includes the processes employed to manage the timely completion of the project. This paper deals with the Resource-Constrained Project Scheduling Problem (RCPSP), which is a part of schedule management. The objective of the problem is to optimize and minimize the project duration while constraining the resource quantities during project scheduling. There are two important constraints in this problem, namely resource constraints and precedence relationships of activities during project scheduling. Many methods such as exact, heuristic, and meta-heuristic have been developed by researchers to solve the problem, but there is a lack of investigation of the problem using methods such as neural networks and machine learning. In this article, we develop a multi-layer feed-forward neural network (MLFNN) to solve the standard single- mode RCPSP. The advantage of this method over evolutionary methods or metaheuristics is that it is not necessary to generate numerous solutions or populations. The developed MLFNN learns based on eight project parameters, namely network complexity, resource factor, resource strength, average work per activity, percentage of remaining work, etc., which are calculated at each step of project scheduling, and identified priority rules, which are the outputs of the developed neural network. Therefore, after the learning process, the network can automatically select an appropriate priority rule to filter out an unscheduled activity from the list of eligible activities and schedule all activities of the project according to the given project constraints. Finally, we investigate the performance of the presented approach using the standard benchmark problems from PSPLIB.

1. Introduction

Project management plays an important role in ensuring that projects achieve their strategic goals, promoting economic growth, growing businesses, and creating value or purposeful infrastructure. Thus, project management has a fundamental role in national development and economic improvement (Habibi, Barzinpour, & Sadjadi, 2018). Project management is defined as the application of knowledge, skills, tools, and techniques to project activities to meet project requirements. Project management is composed of various knowledge areas, such as project integration management, project scope management, project schedule management, project risk management, project cost management, etc.

As mentioned earlier, project schedule management is one of the areas of project management that includes the processes used to manage the timely completion of the project. One of these processes is called develop schedule which is the analysis of the sequence of activities, durations, resource requirements, and schedule constraints to create the project schedule for

* Corresponding author. Tel.: +33750014801
E-mail address: amir.golab@isen-ouest.yncrea.fr (A. Golab)

doi: 10.5267/dsl.2022.7.004

project execution. The objective of this process is to create a schedule model, or the identification and sequencing of activities with planned dates for completion of the project activities (PMI, 2017).

A project is a temporary attempt to create a unique product, service, or outcome (PMI, 2017) that can be represented by project schedule network diagrams called activity-on-node (AoN) network. The network represents the order in which activities should be scheduled to address logical precedences between activities. It usually contains nodes representing activities and arrows representing relationships between activities (Koulinas, Kotsikas, & Anagnostopoulos, 2014).

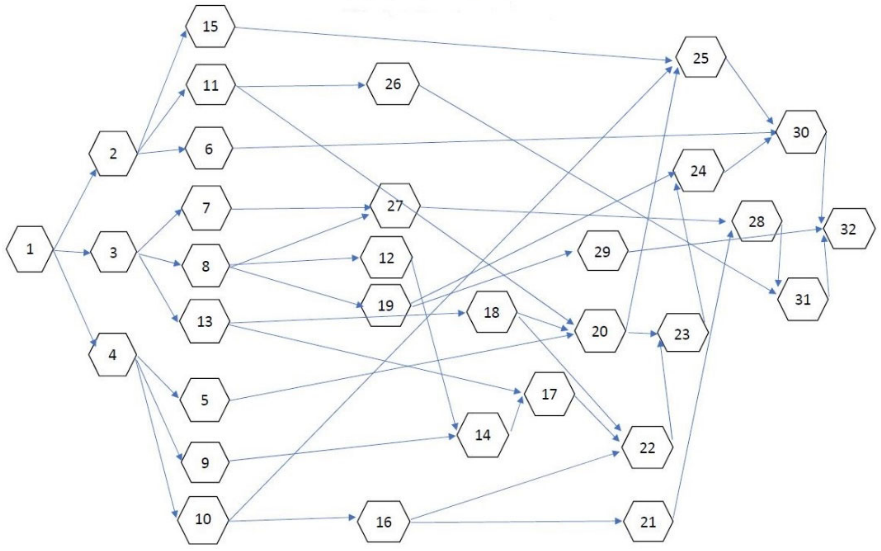

Fig 1. An example of an Activity-on-Node (AoN) network representing a project with 32 activities. Activities 1 and 32 are dummy activities with zero duration. The simple project was selected from the projects of the PSPLIB

The resource-constrained project scheduling problem (RCPSP) is academically similar to the introduced process called develop schedule to obtain project schedule . The RCPSP focuses on creating an activity sequence to optimize and minimize project duration under resource constraints.

The RCPSP is defined by the set $A = \{1, \ldots, i\}$ of activities constrained by two types of constraints, called resource constraints and precedence relations. Eq. (1) states the objective of the problem, which is to minimize or optimize the project duration considering the main constraints. Eq. (2) ensures that all precedence constraints are satisfied during the project schedule. This means that activity i cannot be started until its immediate predecessors are completed. Eq. (3) illustrates that the first and last activities are dummy activities with zero duration on the other word, they are milestones. Moreover, there is a set of renewable resource $R = \{1, \ldots, r\}$ during the execution of the project and each activity consumes u_{irt}units per time for execution. Therefore, Eq. (4) states the respectable constraint of the RCPSP to consider the available resource quantities period by period to obtain resource feasibility during the project scheduling (Kolisch & Hartmann, 2006; Koulinas, Kotsikas, & Anagnostopoulos, 2014; Bouleimen & Lecocq, 2003; Roy & Sen, 2019).

$$\min f_n \qquad (1)$$

subject to

$$f_i \leq f_j - d_j \qquad \forall \left(A_i \; A_j\right) \in Pred \qquad (2)$$

$$f_1 = 0\,, d_1 = 0\,, d_n = 0 \qquad \text{the activities 1 and n are dummy} \qquad (3)$$

$$\textstyle\sum_{i \in P_t} u_{irt} \;\leq\; U_r \qquad \forall\; t = 1, \ldots, f_n \;\text{ and } \forall\;\; r \in R \qquad (4)$$

The defined elements are presented below:

- The set A consists of the project activities with duration d_i and $i = 1, 2, 3, \ldots, n$. the activities 1 and n are dummy.
- The set of $R = \{1, \ldots, r\}$ represents the renewable resources.
- U_r represents the available quantities of renewable resource r
- f_i represents the finish time of activity i
- f_j represents the start time of activity j, which is the immediate successors of activity i.
- The set of $Pred$ consisting of ordered pairs (A_i, A_j) shows that A_j is an immediate successor of A_i
- u_{irt} represents the amount of renewable resource r consumed by activity i in the period t.

Many researchers have developed meta-heuristic methods to solve the problem. However, there is a lack of investigation of the problem using new methods such as machine learning and neural networks (Golab, Sedgh Gooya, Al Falou, & Cabon, 2022). In the following we mention some related articles. The simulated annealing (SA) algorithms were developed for the resource-constrained project scheduling problem (RCPSP) and its multi-mode version (MRCPSP). The objective function considered is to minimize the project duration. In this research, the standard simulated annealing search method is replaced by a new method that takes into account the specificity of the solution space of project scheduling problems (Bouleimen & Lecocq, 2003). A hybrid genetic algorithm with some modifications in the GA model was proposed to solve the RCPSP. The modifications include a specific crossover operator, a local improvement operator, a new way of selecting the parents to be combined, and a two-phase strategy (Valls, Ballestín, & Quintanilla, 2008). A particle swarm optimization (PSO) based hyper heuristic algorithm was proposed for solving the RCPSP. In the developed algorithm, the hyper heuristic works as an upper level algorithm that controls the lower level heuristics that run into the solution space. The schedules are generated according to the serial schedule generation scheme. Also, the double justification operator and a forward-backward improvement procedure are applied to all solutions (Koulinas, Kotsikas, & Anagnostopoulos, 2014). For more information on meta-heuristic methods and related articles, we refer to the review article provided by (Golab, Sedgh Gooya, Al Falou, & Cabon, 2022).

Augmented neural networks was proposed for solving the task-scheduling problem which is a hybrid of the heuristic and the neural networks approaches. The objective of the problem was minimizing the makespan in scheduling n tasks on m machines where the tasks follow a pre-defined precedence relation and task pre-emption is not allowed (Agarwal, Pirkul, & Jacob, 2003). A feed-forward neural network was developed and integrated into the scheduling scheme to automatically select the appropriate priority rules for project scheduling (Shou, 2005). A neurogenetic approach has been developed, which is a hybrid of genetic algorithms (GA) and neural-network (NN) approaches. In this approach, the search process depends on GA iterations for global search and NN iterations for local search. The GA and NN search iterations are interleaved in such a way that the NN can select the best solution from the GA pool. Also, good solutions obtained by the NN search are included in the GA population to use the GA iterations (Agarwal, Colak, & Erenguc, 2011). Project durations were approximated using ANN to minimize or optimize project duration. The study focuses on projects such as residential buildings, schools, and offices that are designed with limited resources. Projects are scheduled according to three priority rules: latest finish time, minimum slack time, and maximum remaining path length (Özkan & Gulçlçek, 2015).

In this research, we use a multilayer feed-forward neural network (MLFNN) approach to solve the standard single-mode resource-constrained project scheduling problem (RCPSP). For this purpose, the developed multilayer feed-forward neural network (MLFNN) is fed with eight project parametric characterizations to select an appropriate priority rule, which is the output of the network. Therefore, at each scheduling step, the system can filter out an appropriate activity selected by priority rule from the list of eligible activities and add it to the project schedule. The algorithm proceeds to schedule all the activities of the project considering the given project constraints.

The rest of the paper is organized as follows. Section 2 explains the priority rules that are the outputs of the MLFNN. Section 3 presents the inputs of MLFNN, called project parametric characterizations. Section 4 mentions the artificial neural networks and explains the developed MLFNN. In Section 5, the serial schedule generation scheme is presented as a decoding function. The computational analysis is presented in section 6. In the last section, a conclusion is drawn.

2. Priority rules

Researchers have introduced priority rules that are used to select an activity from the available decision set or list of eligible activities. A priority rule assigns a value to each activity i in the decision set or the list of eligible activities. The value assignment to the activities is used to filter out the activity with the lowest or highest value. The priority rules are the methods to select the activities according to their selection criteria for project scheduling. Moreover, these rules lead to different consequences depending on the project specifications, such as the number of resources and the existing project conditions, etc. Thus, it is important to select an appropriate priority rule to select an activity from the list of eligible activities, as this action may have an impact on the project duration. If there is more than one activity with the same value, it is possible to select the activity with the smallest activity label. There are different priority rules that can be classified according to different criteria. In the following, we present the different priority rules and their values used in the developed neural network. Also, the selection criteria of the rules are represented in Table 1 (Olaguíbel & Goerlich, 1989; Kolisch & Hartmann, 1999; Ulusoy & Özdamar, 1989; Kolisch, 1996; Shou, 2005; Özkan & Gulçlçek, 2015).

The early start time (EST) and early finish time (EFT) of an activity are calculated using the forward pass technique. They indicate the earliest start and earliest finish that an activity can begin and end in a project. They are calculated according to the formulas below.

$$EST_i = early\ finish\ time\ of\ all\ immediate\ predecessors\ +\ 1 \tag{5}$$

$$EFT_i = EST_i\ +\ d_i \tag{6}$$

However, the backward pass technique is used to determine the latest start time (LST) and the latest finish time (LFT). These are the latest dates that an activity can start or end before it delays the project. Forward-pass and backward-pass are the techniques of the critical path method.

$$\text{LST} = LFT_i - d_i + 1 \tag{7}$$

$$\text{LFT} = latest\ start\ time\ of\ all\ successors\ - 1 \tag{8}$$

Shortest processing time (SPT) is another priority rule that picks up an activity with the shortest duration from the decision set. This means that the activity that takes the least amount of time is completed first.

$$SPT_i = d_i \tag{9}$$

The priority rule of total resource demand (TRD) priority rule selects an activity that requires the least sum of resource units r.

$$TRD_i = \sum_r u_{ir} \tag{10}$$

The total resource scarcity (TRS) is the summation of the required units of resource r by activity i divided by the capacity of resource r.

$$TRS_i = \sum_r u_{ir}/U_i \tag{11}$$

Most total successor (MTS) is the next priority rule that aims to select an activity from the decision set that has the most successors.

$$MTS_i = |\overline{S_i}| \quad , S \text{ is defined the successors of activity } i \tag{12}$$

Slack time or float time refers to the amount of time that an activity can be delayed without affecting the project duration. Thus, the priority rule of minimal slack (MSLK) priority rule filters out an activity with minimum slack time. The value of the rule is determined by the following formula for each activity.

$$MSLK_i = LST_i - EST_i \tag{13}$$

Greatest rank positional weight (GRPW) is another priority rule which is calculated according to the formula.

$$GRPW_i = p_i + \sum_{j \in S_i} p_j \tag{14}$$

Also, the last rule called weighted resource utilization ration and precedence (WRUP) is computed by the following formula.

$$WRUP_i = w|\overline{S_i}| + (1-w)TRS_i \qquad \text{w} = 0.7 \tag{15}$$

Table 1
The priority rules and the selection criteria

Priority rules	Selection criteria
Earliest start time (EST)	Min
Latest start time (LST)	Min
Earliest finish time (EFT)	Min
Latest finish time (LFT)	Min
Shortest processing time (SPT)	Min
Total resource demand (TRD)	Min
Total resource scarcity (TRS)	Min
Most total successors (MTS)	Max
Minimal slack (MSLK)	Min
Greatest rank positional weight (GRPW)	Max
Weighted resource utilization ration and precedence (WRUP)	Max

3. Project parametric characterizations

Three project parametric characterizations, namely network complexity (NC), resource factor (RF), and resource strength (RS) were re-explained by (Kolisch, Sprecher, & Drexl, 1995), which illustrate the structure of a project network. They describe the constraints of the average number of successors per unscheduled activities, the average proportion of resources requested per activity, and the measurement of the availability of resources during the project, receptively (Sprecher & Kolisch, 1997). However, the parameters are used to calculate the factors of a complete project, but they can be modified to outfit the scenarios of partial schedules (Shou, 2005).

$$\text{Network complexity (NC)} = \sum_{i \in US} S_i \Big/ |US| \tag{16}$$

Resource factor (RF) = $\frac{1}{|US|}\frac{1}{|R|}\sum_{i\in US}\sum_{r\in R}\begin{cases}1 & if\ u_{ir} > 0\\ 0 & otherwise\end{cases}$ (17)

Resource strength (RS) = $\sum_{r\in R} {U_r - U_r^{max}} \Big/ {U_r^{max} - U_r^{min}}$ (18)

In addition to three parameters described, we also use other parameters, namely average work per activity, percentage of remaining work, percentage of unscheduled activities, percentage of remaining successors, and average units per day, which can be calculated using the project data. They are also calculated to outfit the scenarios of the partial schedules in this article (Golab, Sedgh Gooya, Al Falou, & Cabon, 2022).

Average work per activity (AWA) = ${\sum_{i\in US} w_i} \Big/ {|US|}$ (19)

Percentage of remaining work (PRW) = ${\sum_{i\in US} w_i} \Big/ {W}$ (20)

Percentage of unscheduled activities (PUA) = ${|US|} \Big/ {|A|}$ (21)

Percentage of remaining successors (PRS) = ${\sum_{i\in US} S_i} \Big/ {\sum_{i\in A} S_i}$ (22)

Average units per day (AUD) = ${\sum_{i\in US} w_i} \Big/ {\sum_{i\in US} d_i}$ (23)

where US is defined as the set of unscheduled activities and the set A consists of the project activities, U_r represents the available quantities of the renewable resource r, and U_r^{max} and U_r^{min} represent the maximum and minimum quantities of the resource r respectively, needed to implement the unscheduled activities, w_i is the work-content of activity i calculated by $w_i = d_i \times u_{ir}$, and W is also the work-content of the project, S_i denotes the immediate successors of activity i and finally, d_i represents the duration of activity i .

These parameters are recalculated at each stage of project scheduling. This means that after scheduling an activity, the parameters are recalculated to characterize the new stage or subproject to be scheduled. These parameters are the inputs of the developed the multilayer feed-forward neural network (MLFNN) which explain in the next section.

4. Artificial neural network

Artificial neural networks (ANN) are flexible, mathematical models and powerful machine learning techniques originally inspired by the functional structure of the human brain that simulate the learning mechanism in the biological organism, where hundreds of billions of interconnected neurons process information in parallel. Artificial neural networks have become a popular and helpful model for optimization, classification, clustering, prediction, etc. in many fields (Svozil, Kvasnicka, & Pospichal, 1997; Wang, 2003; Özkan & Gulçlçek, 2015; Choi, Coyner, Kalpathy-Cramer, Chiang, & Campbell, 2020).

In general, artificial neural networks consist of simple computational units called neurons that act on data while connected by adoptable weight links. Artificial neural networks can be divided into two groups: Feed forward neural networks and feed backward neural networks, but in this context, we focus on feed forward neural networks. A feed forward neural network (FFNN) is an algorithm that consists of arranged layers similar to the neural processing units of the human brain. In FFNN, each unit in one layer is connected to the other units in the other layers. These connections are not all equal, as each connection may have a different weight. The weights of the network connections estimate the possible amount of knowledge of the network. In general, feed-forward neural networks are classified into three types: single-layer, multilayer, and radial basis function networks (Abiodun et al., 2018; Choi, Coyner, Kalpathy-Cramer, Chiang, & Campbell, 2020). In this article, we employ a multilayer neural network, so we will focus on this type of neural network in the following.

Unlike single-layer neural networks, which contain a single input layer and an output node, multilayer neural networks usually consist of the input layer, the output layer, and the layers in between, called hidden layers because they are not visible, and the nodes or neurons organized in said layers. In simple multilayer neural networks, there are one to three hidden layers, but deep neural networks include dozens or hundreds of hidden layers. Multilayer feed-forward neural networks (MLFNN) are trained using the backpropagation learning algorithm. In mathematical science, the concept of learning a neural network or "training a neural network" means adjusting weight coefficients to satisfy certain constraints. A critical aspect of neural networks is the choice of an appropriate network size for a given problem. Network size refers to the number of layers in a network, the number of nodes or neurons per layer, and the number of connections between them. In addition, these types of neural networks are so popular that they are used for a wide variety of problems (Hecht-Nielsen, 1992; Bebis & Georgiopoulos, 1994; Svozil, Kvasnicka, & Pospichal, 1997; Schmidhuber, 2015; Aggarwal, 2018; Choi, Coyner, Kalpathy-Cramer, Chiang, & Campbell, 2020).

The predictive accuracy of a neural network depends on the network size and the type of activation function or transfer function used in the network. Activation functions help in learning and understanding nonlinear and complicated mappings between inputs and corresponding outputs. In other words, they transfer the input signals to the output signals. Thus, the activation function makes the network dynamic and gives it the ability to extract complex and complicated information from the data. Therefore, it is possible to implement the backpropagation optimization strategy to calculate the errors or losses related to the weights and optimize the weights using gradient descent or any other optimization technique to reduce the errors (Sharma, Sharma, & Athaiya, 2017). We use two different activation functions in the developed neural network, which are introduced in the following.

It was explained that multilayer feed-forward neural networks (MLFNNs) are shaped in three types of layers, and each layer consists of a set of interconnected nodes or neurons that benefit from activation functions. In addition, data is fed into the neural network through the input layer. As shown in Figure 2, the developed MLFNN is fed with eight different data. The input layer consists of eight nodes or neurons fed with eight data, which were completely explained in the section of project parametric characterizations. The eight inputs are called average work per activity, percentage of remaining work, percentage of unscheduled activities, percentage of remaining successors, average units per day, network complexity, resource factor, and resource strength. These parameters are computed at each stage of project scheduling to characterize the new step or new subproject required to prepare the conditions for selecting an activity for project scheduling. For example, for the project shown in Fig. 1, there are 32 scheduling phases or stages, so these parameters are calculated 32 times to characterize the subprojects.

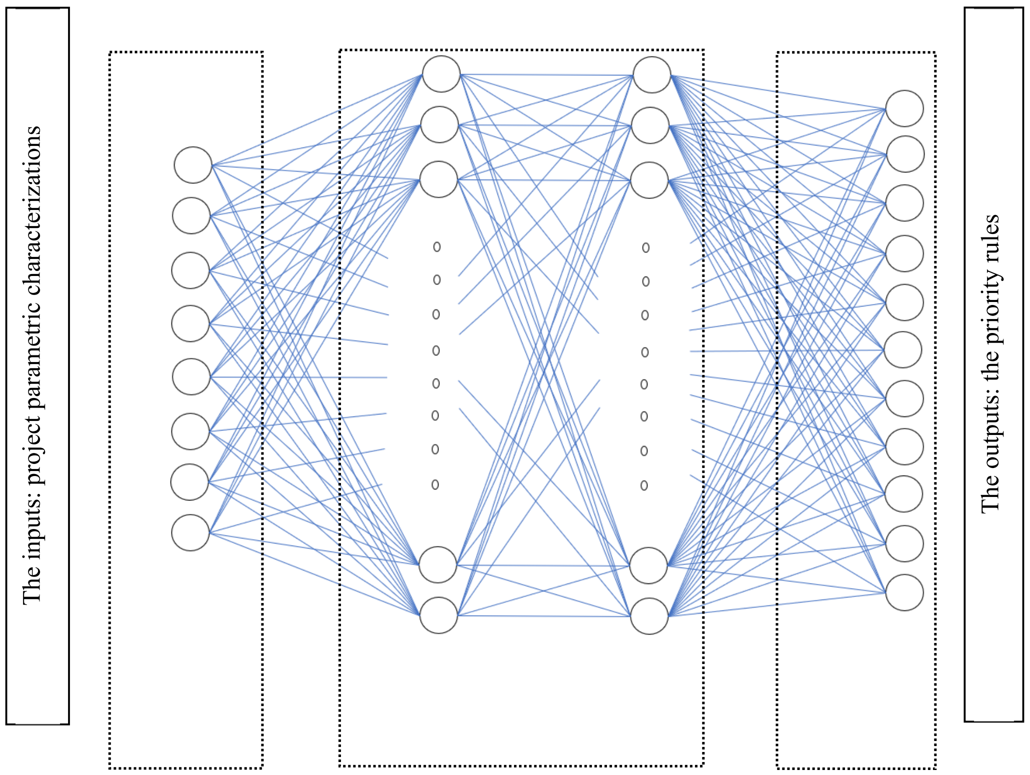
The inputs: project parametric characterizations
The outputs: the priority rules

Fig 2. the developed multilayer feedforward neural network (MLFNN) is composed of eight inputs, two hidden layers and the outputs.

We use hidden layers to create the interconnected neurons of the network. The multilayer feed-forward neural network (MLFNN) is a back-propagation algorithm that consists of two phases. The first is the forward phase and the second is the backward phase. In the first phase, the activation functions are propagated from the input layer to the output layer and in the second phase, the error between the observed actual value and the eligible nominal value in the output layer are propagated backwards in order to improve the weights and bias values (Agarwal, Colak, & Erenguc, 2011; Aggarwal, 2018).

It has been illustrated that if no activation function is used in a neural network, the output would be just a simple linear function. So, we need to apply activations to make the network dynamic and give it the ability to extract complicated data. For this purpose, two different activation functions are embedded in the developed MLFNN. The first type is called relu and is a non-linear activation. This function can be calculated mathematically as and is widely used in neural networks. This activation function is used for the hidden layers of the developed MLFNN. The second type is sigmoid, which is the most widely used activation function. This is a nonlinear function that transforms the values in the range of 0 to 1 thus providing the predictions as outputs. The sigmoid activation function is defined as and we use it for the output layer.

The output layer is composed of priority rules, which were explained in detail in the section of priority rules. Initially, eleven priority rules are provided for the output layer. The duty of the priority rules mentioned above is to select an activity from the list of eligible activities at each step of project scheduling. For example, if the output of the network is minimal slack time (MSLK), the activity with the minimum float or slack time is selected for scheduling. To test the performance of the developed multilayer feed-forward neural network (MLFNN), the output layer is given a different number of neurons or priority rules.

The network is tested with eleven, seven, five, four and three priority rules or neurons for the output layer and the training performance results are reported in Table 2, Table 3 and Table 4.

To learn or train the developed multilayer feed-forward neural network (MLFNN), we use the PSPLIB's projects to create a dataset to feed the developed MLFNN. All eight project parametric characterizations are computed at each stage of the selected PSPLIB's projects. To determine the output column of the dataset, the method or priority rule that provides a minimum project duration to that stage of the scheduling is selected.

It is obvious that the number of priority rules as output columns in the original dataset is not balanced. To achieve better performance of the developed multilayer feed-forward neural network (MLFNN), we balance the initial dataset. This means that the number of priority rules as outputs in the final dataset is the same after the initial dataset is balanced.

It was mentioned that the aim of the learning or training process of the developed MLFNN is to discover the optimal set of weights that ideally gives the correct output. Therefore, the inputs, the outputs, and the test sets must be initialized before training.

5. Serial schedule generation scheme

There are two different schedule generation schemes for generating a feasible schedule. The so-called serial schedule generation scheme (SSGS) and the so-called parallel schedule generation scheme (PSGS). For an understanding of the differences between these two schemes, the reader is referred to (Kolisch & Hartmann, 1999). The serial SGS constructs active schedules that contain at least one optimal solution. In other words, there is a sequence of activities that can be created by the SSGS that leads to an optimal schedule. The serial schedule generation scheme always constructs feasible schedules. In this research, we use SSGS for the decoding function. (Koulinas, Kotsikas, & Anagnostopoulos, 2014).

The serial SGS consists of n stages, where n refers to the stages, iterations of scheduling, and the number of project activities. In the process, there are three set so-called the un-scheduled set US_n, the eligible set E_n, and the scheduled set S_n. These sets are updated at each stage or iteration of scheduling. The unscheduled activities can be a member of the eligible set when their predecessors have been scheduled. (Kolisch & Hartmann, 1999; Koulinas, Kotsikas, & Anagnostopoulos, 2014)

Initialization begins with a partial schedule consisting only of the dummy start activity at time 0. In each stage or iteration of the project scheduling, one activity is selected from the eligible set E_n by the selected priority rule. It has been explained how the priority rule is selected as an output of the multilayer feed-forward neural network (MLFNN). The selected activity is scheduled at the earliest possible time that is feasible in terms of both precedence and resource availability. The process continues to schedule all activities of the project.

6. Computational analysis

In this section we present the results and related dicussions of the research. The results consist of the MLFNN training performance results and the comparative results. In the following, the strengths and weakness of the developed MLFNN are mentioned.

6.1 MLFNN training performance results and discussion

We provided 80% of the data from the balanced dataset for training the developed multilayer feed-forward neural network (MLFNN), and the remaining data, which is 20% of the dataset, was assigned to the testing process. In addition, the batch size and learning rate parameters were set to 64 and 0.0007, respectively. We run the MLFNN 10 times for each mode. Table 2, 3 and 4 represent the training performance of the developed MLFNN. As mentioned earlier, the developed MLFNN was investigated with eleven, seven, five, four, and three priority rules or neurons for the output layer. Since the network size also matters, the developed MLFNN was analyzed with 1, 2, and 3 hidden layers and a different number of epochs of 500, 1000, and 2000. As mentioned earlier, the sigmoid function determines the probability for each output. So that we confirm the priority rules with a probability greater than 0.5 on that stage of scheduling, which are the outputs of the developed neural network. This means that the outputs with probability greater than 0.5 can be an allowable priority rule to select an activity from the list of eligible activities. Also, we approve the maximum probability assigned to a priority rule to be eligible as a priority rule for selecting an activity if no probability greater than 0.5 is observed in the output.

Table 2
Training performance of developed MLFNN with one hidden layer

Priority rules used for output layer	Hidden layers	Epoch	Min accuracy	Max accuracy	Average accuracies
Three priority rules are used as outputs: EST, LST and EFT	1	500	64%	76%	70%
	1	1000	65%	69%	67%
	1	2000	62%	71%	66%
Four priority rules are used as outputs: EST, LST, EFT and LFT	1	500	65%	71%	68%
	1	1000	62%	70%	65%
	1	2000	61%	69%	64%
Five priority rules are used as outputs: EST, LST, EFT, LFT and MTS	1	500	63%	70%	66%
	1	1000	60%	68%	64%
	1	2000	54%	62%	60%
Seven priority rules are used as outputs: EST, LST, EFT, LFT, MTS, TRD and PT	1	500	59%	64%	61%
	1	1000	54%	61%	58%
	1	2000	53%	58%	56%
Eleven priority rules are used as outputs: EST, LST, EFT, LFT, MTS, TRD, PT, GRPW, ST, WRUP and TRS	1	500	48%	57%	51%
	1	1000	43%	52%	48%
	1	2000	42%	46%	44%

Table 3
Training performance of developed MLFNN with two hidden layers

Priority rules used for output layer	Hidden layers	Epoch	Min accuracy	Max accuracy	Average accuracies
Three priority rules are used as outputs: EST, LST and EFT	2	500	62%	67%	65%
	2	1000	63%	68%	66%
	2	2000	64%	70%	66%
Four priority rules are used as outputs: EST, LST, EFT and LFT	2	500	61%	67%	64%
	2	1000	61%	68%	64%
	2	2000	60%	64%	63%
Five priority rules are used as outputs: EST, LST, EFT, LFT and MTS	2	500	59%	63%	61%
	2	1000	58%	62%	61%
	2	2000	58%	61%	59%
Seven priority rules are used as outputs: EST, LST, EFT, LFT, MTS, TRD and PT	2	500	54%	58%	56%
	2	1000	51%	58%	54%
	2	2000	50%	57%	54%
Eleven priority rules are used as outputs: EST, LST, EFT, LFT, MTS, TRD, PT, GRPW, ST, WRUP and TRS	2	500	41%	47%	44%
	2	1000	39%	43%	41%
	2	2000	37%	43%	40%

Table 4
Training performance of developed MLFNN with three hidden layers

Priority rules used for output layer	Hidden layers	Epoch	Min accuracy	Max accuracy	Average accuracies
Three priority rules are used as outputs: EST, LST and EFT	3	500	63%	68%	66%
	3	1000	63%	68%	65%
	3	2000	64%	69%	67%
Four priority rules are used as outputs: EST, LST, EFT and LFT	3	500	60%	65%	64%
	3	1000	61%	65%	63%
	3	2000	60%	65%	63%
Five priority rules are used as outputs: EST, LST, EFT, LFT and MTS	3	500	57%	64%	61%
	3	1000	59%	64%	61%
	3	2000	58%	63%	60%
Seven priority rules are used as outputs: EST, LST, EFT, LFT, MTS, TRD and PT	3	500	55%	61%	57%
	3	1000	51%	59%	55%
	3	2000	54%	58%	56%
Eleven priority rules are used as outputs: EST, LST, EFT, LFT, MTS, TRD, PT, GRPW, ST, WRUP and TRS	3	500	43%	51%	46%
	3	1000	39%	45%	43%
	3	2000	38%	43%	41%

Based on the training performance, it can be observed that the performance of the developed multilayer feed-forward neural network (MLFNN) increases when the fixed number of outputs is reduced. For example, the probability of selecting the appropriate priority rule was higher when three neurons were used as outputs in the developed network than when eleven neurons were used.

6.2 Comparative results and discussion

It was mentioned that the aim of this research is to use the multilayer feed-forward neural network (MLFNN) to solve the RCPCP. The superiority of this method over evolutionary methods or metaheuristics is that it is not necessary to generate many solutions or populations. The MLFNN only needs to train the weights only once, and the obtained weights are then used to schedule all projects or all instances.

In the previous sections, the developed MLFNN and training performance were explained. After training the MLFNN, we apply the algorithm to schedule the standard instances. The standard problem instances used consist of projects with four renewable resource types and 60 and 120 activities respectively. The standard test instances were selected from the project planning instance library PSPLIB.

The procedure was coded in Python 3 and the instances were run on a laptop with Intel Core i5 vPro 2.30GHz and 16.0 GB.

Results are presented in terms of an average percentage of deviations from the lower bound based on the critical path for the project instances with 60 activities and 120 activities. The critical path lower bound refers to sequencing all the activities of a project according to their precedence relations . Table 5 summarizes the obtained results for 181 instances of J60 and 160 instances of J120. The results presented in Table 5 show that the average deviations from the critical path lower bound are better when the performance of the developed multilayer feed-forward neural network (MLFNN) increases. In the previous section, it was mentioned that the performance of the MLFNN increases when the fixed number of outputs is reduced. The average deviations from the lower bound of the critical path obtained 15.97% and 37.77% for J60 and J120, respectively, which are better than the other results reported in Table 5.

Table 5
Percentage of average deviations from critical path lower bound for the J60 and J120

Priority rules used for output layer	**Number of activities**	
	60	**120**
Three priority rules are used as outputs: EST, LST and EFT	15.97	37.77
Four priority rules are used as outputs: EST, LST, EFT and LFT	16.28	39.74
Five priority rules are used as outputs: EST, LST, EFT, LFT and MTS	47.04	89.61
Seven priority rules are used as outputs: EST, LST, EFT, LFT, MTS, TRD and PT	45.59	86.39
Eleven priority rules are used as outputs: EST, LST, EFT, LFT, MTS, TRD, PT, GRPW, ST, WRUP and TRS	58.96	117.42

Table 6
Percentage of average deviations from critical path lower bound for the J60

Reference	**Algorithm**	**Deviation**
Our algorithm	MLFNN **(**three neurons used as outputs)	15.97
Our algorithm	MLFNN (four neurons used as outputs)	16.28
(Valls, Ballestin, & Quintanilla, 2008)	Hybrid GA	11.56
(Alcaraz & Concepción, 2001)	Genetic Algorithm	NA
(Hartmann, 2002)	Genetic Algorithm	12.21
(Nonobe & Ibaraki, 2002)	Tabu Search	12.97
(Koulinas, Kotsikas, & Anagnostopoulos, 2014)	Particle Swarm Optimization-HH	11.74
(Bouleimen & Lecocq, 2003)	Simulated Annealing	12.75
(Mendes, Gonçalves, & Resende, 2009)	Genetic Algorithm	11.72
(Chen, Shi, Teng, Lan, & Hu, 2010)	Hybrid (ACO and SS)	11.75
(Agarwal, Colak, & Erenguc, 2011)	Neurogenetic	11.51
(Mobini, Mobini, & Rabbani, 2011)	Artificial Immune Algorithm	11.17
(Gonçalves, Resende, & Mendes, 2011)	Genetic Algorithm	11.56
(Wang & Fang, 2012)	Hybrid Estimation of Distribution Algorithm	11.44
(Chen R.-M. , 2011)	Particle swarm optimization	12.03
(Ziarati & Akbari, 2011)	Bee Algorithms	12.55
(Proon & Jin, 2011)	Genetic Algorithm with Neighborhood Search	11.35
(Liu, Liu, Shi, & Li, 2020)	Genetic Algorithm	11.74
(Zamani, 2017)	Genetic Algorithm	11.61
(Lim, Ma, Rodrigues, & Tan, 2013)	Hybrid Genetic Algorithm	11.73

Table 6 and Table 7 present comparative results for J60 and J120 standard instances. Our method achieves average deviations of 15.97% and 16.28% when the MLFNN uses three neurons as outputs and four neurons as outputs for J60, respectively. They are not among the best but can be competitive. Table 7 shows that our method achieves better performance. The average deviations are 37.77% and 39.74% when the MLFNN uses three neurons and four neurons as outputs for J120 instances, respectively. The comparative results in Table 7 present that the choice of priority rules by the developed MLFNN leads to sufficiently competitive results.

The results show that the performance of the procedure and the solutions obtained can be improved. We suggest that this can be done by choosing appropriate priority rules as outputs and developing more neural networks. The major advantage of using neural networks such as the multilayer feed-forward neural network (MLFNN) used in this study is that the project can be scheduled only by generating a sequence activity. This method prevents the generation of a large number of solutions.

Table 7
Percentage of average deviations from critical path lower bound for the J120

Reference	Algorithm	Deviation
Our algorithm	MLFNN (three neuron used as outputs)	37.77
Our algorithm	MLFNN (four neuron used as outputs)	39.74
(Valls, Ballestin, & Quintanilla, 2008)	Hybrid GA	34.07
(Alcaraz & Concepción, 2001)	Genetic Algorithm	39.36
(Hartmann, 2002)	Genetic Algorithm	37.19
(Nonobe & Ibaraki, 2002)	Tabu Search	40.86
(Koulinas, Kotsikas, & Anagnostopoulos, 2014)	Particle Swarm Optimization-HH	35.20
(Bouleimen & Lecocq, 2003)	Simulated Annealing	40.82
(Mendes, Gonçalves, & Resende, 2009)	Genetic Algorithm	35.87
(Chen, Shi, Teng, Lan, & Hu, 2010)	Hybrid (ACO and SS)	35.19
(Agarwal, Colak, & Erenguc, 2011)	Neurogenetic	34.65
(Mobini, Mobini, & Rabbani, 2011)	Artificial Immune Algorithm	30.04
(Gonçalves, Resende, & Mendes, 2011)	Genetic Algorithm	35.94
(Wang & Fang, 2012)	Hybrid Estimation of Distribution Algorithm	34.83
(Chen R.-M. , 2011)	Particle swarm optimization	35.71
(Ziarati & Akbari, 2011)	Bee Algorithms	37.72
(Proon & Jin, 2011)	Genetic Algorithm with Neighborhood Search	33.45
(Liu, Liu, Shi, & Li, 2020)	Genetic Algorithm	34.88
(Zamani, 2017)	Genetic Algorithm	34.59
(Lim, Ma, Rodrigues, & Tan, 2013)	Hybrid Genetic Algorithm	34.95

7. Conclusion

In this research, a multilayer feed-forward neural network (MLFNN) was developed for solving the resource-constrained project scheduling problem (RCPSP). The advantage of this method over evolutionary methods or metaheuristics is that it is not necessary to generate numerous solutions or populations, on the contrary, this method generates a solution according to the training of the MLFNN. For this purpose, the MLFNN uses the eight parameters as inputs and a different number of priority rules as outputs. It is obvious that the project specifications may change during the scheduling of the project. Therefore, the performance of the priority rules depends on the project specifications such as the number of resources, existing project constraints, and so on. Therefore, different priority rules are suitable for different types of projects or sub-projects. So, in this context, we used the MLFNN with a mixture of priority rules as outputs to improve the performance of project scheduling.

The PSPLIB projects were used to create the dataset for the MLFNN training and testing procedure. After training the network, the MLFNN selects a priority rule as output. Finally, the appropriate activity can be picked up by the selected priority rule from the list of eligible activities to be added to the project schedule at each stage of the schedule. The training performance of the MLFNN was investigated using the balanced data set after setting the parameters of the network. It was found that the performance of the network increases when the number of outputs is reduced. In the following, we applied the algorithm for scheduling the problem instance library PSPLIB after training the MLFNN. The computational results show that the results are competitive.

Consequently, we believe that the developed multilayer feed-forward neural network has a reasonable performance to deal with project scheduling problems. Although the obtained results are not yet the best, this study should encourage researchers to exploit potential improvements to develop neural networks or other machine learning methods to deal with project scheduling problems.

References

Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., & Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. *Heliyon, 4*(11).

Agarwal, A., Colak, S., & Erenguc, S. (2011). A neurogenetic approch for the resource-constrained project schedulig problem. *Computers & operations research, 38*(1), 44 - 50.

Agarwal, A., Pirkul, H., & Jacob, V. S. (2003). Augmented neural networks for task scheduling. *European Journal of Operational Research, 151*(3), 481-502.

Aggarwal, C. C. (2018). *Neural networks and deep learning.* springer.

Alcaraz, J., & Concepción, M. (2001). A robust genetic algorithm for resource allocation in project scheduling. *Annals of operations Research, 102*(1), 83-109.

Bebis, G., & Georgiopoulos, M. (1994). Feed-forward neural networks. *IEEE Potentials, 13*(4), 27-31.

Bouleimen, K., & Lecocq, H. (2003). A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European journal of operational research, 149*(2), 268-281.

Chen, R.-M. (2011). Particle swarm optimization with justification and designed mechanisms for resource-constrained project scheduling problem. *Expert Systems with Applications, 38*(6), 7102-7111.

Chen, W., Shi, Y.-j., Teng, H.-f., Lan, X.-p., & Hu, L.-c. (2010). An efficient hybrid algorithm for resource-constrained project scheduling. *Information Sciences, 180*(6), 1031-1039.

Choi, R. Y., Coyner, A. S., Kalpathy-Cramer, J., Chiang, M. F., & Campbell, J. P. (2020). introduction to machine learning, neural networks, and deep learning. *Translational Vision Science & Technology, 9*(2), 14 - 14.

Golab, A., Sedgh Gooya, E., Al Falou, A., & Cabon, M. (2022). Investigating the performance of an artificial neural network for solving the resource constrained project scheduling problem (RCPSP). *In Pattern Recognition and Tracking XXXIII. 12101*, pp. 78-83. Florida: SPIE. doi:https://doi.org/10.1117/12.2618499

Golab, A., Gooya, E., Falou, A & Cabon, M. (2022). Review of conventional metaheuristic techniques for resource-constrained project scheduling problem.Journal of Project Management, 7(2), 95-110. doi:10.5267/j.jpm.2021.10.002

Gonçalves, J. F., Resende, M. G., & Mendes, J. J. (2011). A biased random-key genetic algorithm with forward-backward improvement for the resource constrained project scheduling problem. *Journal of Heuristics, 17*(5), 467-486.

Habibi, F., Barzinpour, F., & Sadjadi, S. J. (2018). Resource-constrained project scheduling problem: review of past and recent developments. *Journal of Project Management*, *3*(2), 55-88.

Hartmann, S. (2002). A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics (NRL), 49*(5), 433-448.

Hecht-Nielsen, R. (1992). Theory of the backpropagation neural network. *Neural networks for perception*, 65-93.

Kolisch, R. (1996). Efficient priority rules for the resource-constrained project scheduling problem. *Journal of Operations Management, 14*(3), 179-192.

Kolisch, R., & Hartmann, S. (1999). heuristic algorithms for solving the resource-constrained project scheduling problem: classification and computational analysis. *Project scheduling* (pp. 147 - 178). Boston: Springer.

Kolisch, R., & Hartmann, S. (2006). Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research, 174*(1), 23-37.

Kolisch, R., Sprecher, A., & Drexl, A. (1995). characterisation and generation of a general class of resource-constrained project scheduling problem. *management science, 41*(10), 1693 - 1703.

Koulinas, G., Kotsikas, L., & Anagnostopoulos, K. (2014). A particle swarm optimization based hyper-heuristic algorithm for the classic resource constrained project scheduling problem. *Information Sciences, 277*(1), 680-693.

Lim, A., Ma, H., Rodrigues, B., & Tan, S. T. (2013). New meta-heuristics for the resource-constrained project scheduling problem. *Flexible Services and Manufacturing Journal, 25*(1), 48-73.

Liu, J., Liu, Y., Shi, Y., & Li, J. (2020). Solving resource-constrained project scheduling problem via genetic algorithm. *Journal of Computing in Civil Engineering, 34*(2).

Mendes, J. J., Gonçalves, J. F., & Resende, M. G. (2009). Mendes, Jorge JM, José Fernando Gonçalves, and Mauricio GC Resende. "A random key based genetic algorithm for the resource constrained project scheduling problem. *Computers & operations research, 31*(1), 92-109.

Mobini, M., Mobini, Z., & Rabbani, M. (2011). An Artificial Immune Algorithm for the project scheduling problem under resource constraints. *Applied Soft Computing, 11*(2).

Nonobe, K., & Ibaraki, T. (2002). Formulation and tabu search algorithm for the resource constrained project scheduling problem. *In Essays and surveys in metaheuristics* (pp. 557-588). Boston: Springer.

Olaguíbel, R. A., & Goerlich, J. M. (1989). Heuristic algorithms for resource-constrained project scheduling: A review and an empirical analysis. *Advances in project scheduling*, 113-134.

Özkan, Ö., & Gulçlçek, Ü. (2015). A neural network for resource constrained project scheduling programming. *Civil Engineering and Management, 21*(2), 193 - 200.

PMI. (2017). *Project management body of knowledge (PMP Guide) SIXTH Edition.*

Proon, S., & Jin, M. (2011). A genetic algorithm with neighborhood search for the resource-constrained project scheduling problem. *Naval Research Logistics (NRL), 58*(2), 73-82.

Roy, B., & Sen, A. K. (2019). Meta-heuristic Techniques to Solve Resource-Constrained Project Scheduling Problem. In *International conference on innovative computing and communications* (pp. 93-99). Springer.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 85-117.

Sharma, S., Sharma, S., & Athaiya, A. (2017). Activation functions in neural networks. *towards data science, 6*(12), 310 - 316.

Shou, Y. (2005). A neural network based heuristic for resource-constrained project scheduling. *International Symposium on Neural Networks* (pp. 794- 799). Berlin, Heidelberg: Springer.

Sprecher, A., & Kolisch, R. (1997). PSPLIB - A project scheduling problem library: OR Software - ORSEP Operations Research Software Exchange Program. *European Journal of Operational Research, 6*(1), 205-216.

Svozil, D., Kvasnicka, v., & Pospichal, J. (1997). Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems, 39*(1), 43-62.

Ulusoy, G., & Özdamar, L. (1989). Heuristic performance and network/resource characteristics in resource-constrained project scheduling. *Journal of the operational research society, 40*(12), 1145 - 1152.

Valls, V., Ballestín, F., & Quintanilla, S. (2008). A hybrid genetic algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research, 185*(2), 495-508.

Wang, L., & Fang, C. (2012). A hybrid estimation of distribution algorithm for solving the resource-constrained project scheduling problem. *Expert Systems with Applications, 39*(3), 2451-2460.

Wang, S.-C. (2003). Artificial neural network. In *Interdisciplinary computing in java programming* (pp. 81-100). Boston: springer.

Zamani, R. (2017). An evolutionary implicit enumeration procedure for solving the resource-constrained project scheduling problem. *International Transactions in Operational Research, 24*(6), 1525-1547.

Ziarati, K., & Akbari, R. ,. (2011). On the performance of bee algorithms for resource-constrained project scheduling problem. *Applied Soft Computing, 11*(4), 3720-3733.