# A Neural Net-Based Approach for CPU Utilization

Sibel SENAN

Department of Computer Engineering, İstanbul Üniversity, Istanbul, Turkey
ssenan@istanbul.edu.tr
(Geliş/Received:04.01.2017; Kabul/Accepted:11.07.2017)
DOI: 10.17671/gazibtd.331037

***Abstract*—** CPU scheduling is an important subject to maximize CPU utilization in the context of operating systems. Multiprogramming operating systems need CPU scheduling for organization of processes to be executed. The order of process execution is determined by a CPU scheduling policy in use. The utilization of CPU depends on the selection of scheduling algorithms. There are several scheduling policies in the literature such as First-Come, First-Served scheduling, Shortest-Job-First scheduling, Last-Come, First-Served scheduling, Priority scheduling. On the other hand, there are some criteria (waiting time, throughput number, turnaround time, response time) to measure the eficiency of these policies. It is important that we choose the scheduling policy which has the minimum waiting time as this is crucial stage of utilizing CPU efficiently. This paper explores an alternative, neural network approach to build a CPU scheduling model to obtain the waiting time measure. In this paper, we will show that neural networks can be used to model scheduling policies and can predict the waiting time of processes. Three learning algorithms and three different neuron numbers in the hidden layer of the network are studied to boost the eficiency of neural network model for waiting time prediction. A comparison between Neural-Network Based Model and First-Come, First-Served scheduling, Shortest-Job-First scheduling, Last-Come, First-Served scheduling are provided. The results reveal the effectiveness of neural networks in predicting waiting times, and thus suggest that it can be useful and practical addition to the framework of operating systems.

***Keywords*—** CPU Scheduling, Operating Systems, Neural Networks.

# CPU Kullanımı için Yapay Sinir Ağı Tabanlı Yaklaşım

***Özet*—** CPU kullanımını maksimize etmek için kullanılan CPU zamanlama, işletim sistemlerinde önemli bir konudur. Çoklu programlamalı işletim sistemleri, çalıştırılacak işlemlerin organize edilmesi için CPU zamanlamasına ihtiyaç duyarlar. Çalıştırılacak işlemlerin sırası kullanılan CPU zamanlama politikası tarafından belirlenir. CPU kullanımı seçilen zamanlama algoritmasına bağlıdır. Literatürde, İlk Gelen Önce zamanlama, En Kısa İş zamanlama, Son Gelen Önce zamanlama, Öncelikli zamanlama gibi farklı algoritmalar bulunmaktadır. Diğer yandan, bu zamanlama politikalarının verimliliğini ölçmek için kullanılan bazı kriterler (bekleme süresi, iş üretimi sayısı, dönüş süresi, yanıt süresi) vardır. Minimum bekleme süresine sahip zamanlama algoritmasının seçimi CPU kullanımının verimli olması açısından önemli bir aşamadır. Bu çalışma, yapay sinir ağı yaklaşımının CPU zamanlama modeli oluşturarak bekleme süresi ölçütünü elde etmede kullanılabilirliğini araştırmaktadır. Bu çalışma ile yapay sinir ağlarının zamanlama politikalarının modellenmesinde kullanılabileceği ve işlemlerin bekleme zamanını tahmin edebileceği gösterilecektir. Yapay sinir ağı modelinin bekleme süresi tahmini için verimliliğini artırmak amacı ile üç farklı öğrenme algoritması ve her öğrenme algoritması için üç farklı gizli katman nöron sayısı çalışılmıştır. Yapay sinir ağı tabanlı model ile İlk gelen önce zamanlama, En kısa iş zamanlama, Son gelen önce zamanlama algoritmalarının karşılaştırılması sağlanmaktadır. Sonuçlar yapay sinir ağı yaklaşımının bekleme süresi tahminindeki etkinliğini göstermektedir ve böylece işletim sistemleri çerçevesine faydalı ve pratik bir katkı sağlanmaktadır.

***Anahtar Kelimeler*—** CPU Zamanlama, İşletim Sistemleri, Yapay Sinir Ağları.

## 1. INTRODUCTION

Scheduling is the way of sharing computer resources between multiple processes by operating system. Operating system switches CPU among the processes according to a scheduling policy in use [1].

One of the purposes of the operating system is keeping CPU as busy as possible to maximize the performance of the CPU and to make the computer more productive. The objective is to have some process running at all times, to maximize CPU utilization. Because, maximizing CPU utilization means maximizing the overall performance of the computer system. Therefore, CPU scheduling is considered as a fundamental topic in operating system concept. Since the execution order of processes is determined by scheduling policies, CPU utilization depends on the selection of the scheduling policy to be used. There are several scheduling policies in the literature such as First-Come, First-Served policy, Shortest-Job-First policy, Last-Come, First-Served policy, Priority policy. Scheduling policies are generally classified into preemptive and non-preemptive scheduling disciplines[1-3].

Preemptive Scheduling : running task may be interrupted for some time and resumed later when the priority task has finished its execution.

Non-preemptive Scheduling : running task is executed without interruption. It cannot be interrupted until terminated.

The choice of a particular policy may favor one class of processes over another. In choosing which policy to use in a particular situation, we must consider the algorithmic properties of these policies. There are several criteria to measure the efficiency of these algorithms [2] :

*1. CPU utilization* : It is defined as the value of time CPU is in use. The goal of the CPU scheduling is to maximize the CPU utilization.

*2. Throughput* : It is defined as the number of processes that are completed per time unit.

*3. Turnaround Time* : It is defined as a total time which is spend to complete the process from the time of submission to the time of completion.

*4. Response Time* : It is defined as the time passed until the first response is produced for a process execution.

*5. Waiting Time* : It is defined as the total time a process has been waiting in ready queue.

*6. Context Switch* : It is defined as a computing process of storing and restoring state of a CPU so that execution can be resumed from same point at a later time. Context switch are usually computationally intensive, lead to wastage of time, memory, scheduler overhead so much of the design of operating system is to optimize these switches.

The goal of the CPU scheduling is to maximize CPU utilization and throughput and to minimize turnaround time, waiting time, and response time. Therefore, minimum waiting time is one of the characteristics of the effective scheduling algorithm.

Recently, a great deal of papers have conducted important resercahes into the CPU scheduling algorithms and their performances. In [3], a new CPU scheduling algorithm called MIN-MAX has been proposed, focusing on the comparative study of the existing algorithms on the basis of various scheduling parameters with the proposed algorithm MIN-MAX. In [4]-[6], the review of different scheduling algorithms has been performed with different parameters, such as running time, burst time and waiting times etc. The CPU scheduling algorithm with improved performance has been presented in [7]. The technique used in this paper for increasing the speed up factor is 'Pipelining'. This technique can be applied to any CPU scheduling algorithm to improve its performance. [8] has carried out a comparative study of various scheduling algorithms for a single CPU and determines which algorithm may be the best for a specific situation. [9] has proposed a new CPU scheduling algorithm called Combinatory that combines the functions of some basic scheduling algorithms. In [10], the author has developed an interactive Java-based simulator that uses graphical animation to convey the concepts of various CPU scheduling algorithms. A CPU scheduling algorithm that can handle all types of processes with optimum scheduling criteria has been proposed in [11]. In [12], a new Round Robin scheduling algorithm has been given. The authors of [13] have realised the implementation of a new CPU scheduling algorithm called An Optimum Multilevel Dynamic Round Robin Scheduling (OMDRRS) in order to improve Round Robin scheduling algorithm using dynamic time slice concept. They have also simulated the behavior of various CPU scheduling algorithms. The authors of [14] have simulated different scheduling algorithms and evaluated their performances (throughput, latency, utilization, turnaround time, and waiting time) in a multi-processor environment.

Neural networks have the ability to model a function without knowing the exact character of this function. Therefore, a neural network can be considered as a black box that needs to be well-defined for the problem interested. Unlike the traditional methods, in neural networks, the only thing we need is to determine the endpoints (inputs and outputs). Therfore, neural network models do not need to derive metrics from some certain types of equations. Since selecting a scheduling algorithm according to a minimum waiting time is a fundamental step to utilize CPU efficiently, it is important to determine whether a neural network could be used as a tool to generate waiting time. The aim of this paper is to evaluate neural networks as a computational tool to estimate the waiting time of process sets. Hence, we first establish neural networks as a method for modeling scheduling policies by showing that we can model three widely accepted policies; First Come-First Served (FCFS) scheduling policy, Last-Come, First-Served (LCFS) scheduling policy and Shortest-Job-First (SJF) scheduling policy. We use three different training algorithms and three different neuron numbers in the hidden layer of the network to evaluate the estimation results according to the network architecture and training method. The results of FCFS, LCFS and SJF scheduling algorithms and the designed neural network model are compared to show the effectiveness of the proposed method. Obtained results show that the estimation ability of designed neural network model is highly powerful. We also showed that the neural network approach can be considered as a successful alternative method to calculate waiting time of processes to analyze CPU utilization when compared with the traditional methods.

## 2. SCHEDULING POLICIES

Scheduling policies decide which of the processes in the ready queue is to be allocated the CPU. In this study, we considered the following scheduling policies that all are non-preemptive scheduling algorithms to calculate the waiting time criterion and to train the designed neural network according to these results.

2.1 First-Come, First-Served Scheduling
FCFS, allows a process that requests CPU first, holds the CPU first. Process in ready queue is executed on the basis of arrival time, without any preemption. Once CPU has started executing a process, it cannot be interrupted unless completed.

2.2 Last-Come, First-Served Scheduling
LCFS, allows a process that requests CPU last, holds the CPU first. The last process in ready queue is executed without any preemption.

2.3 Shortest-Job-First Scheduling
Burst time is exact time that is required to complete execution of particular process. CPU scheduling algorithms require burst time as input. SJF, allows a process with the minimum burst time holds the CPU first. FCFS policy is used, when two processes have the equal CPU burst time.

## 3. NEURAL NETWORKS

Neural networks are powerful mathematical and computational tools widely used for estimating engineering problems due to their ability of learning. In recent years, neural network systems have become a popular solving technique and are used in many fields such as accounting, civil engineering, mine engineering, environmental engineering, medicine, etc. Designing an appropriate neural network model for the system in interest is a fundamental issue in neural network approach [15-19]. Therefore, designing an estimating model for CPU scheduling by Neural Networks can be considered as a significant concept in both operating systems and neural networks fields. Since selecting a scheduling algorithm according to a minimum waiting time is a crucial step to utilize CPU efficiently, it can be considered as a intuitive decision making method to design a neural network model to estimate total waiting time of processes.
A typical feed-forward neural network model consists of three layers of neurons that are input, hidden, and output. When a feed-forward neural network is being trained, the connection weights are updated to minimize the error between the desired and estimated values of the system variables [20].

In this study, we have designed a feed-forward neural network with one hidden layer to estimate waiting time of processes in Table 1. Three distinct training algorithms (Levenberg - Marquardt (L-M) Algorithm, Conjugate - Gradient (C-G) Algorithm with Polak-Ribiere updates, Gradient – Descent (G-D) Algorithm) are used to train the network [21-25]. The number of hidden neurons is chosen to be 10, 20 and 30 for each algorithm to evaluate the optimum result of the network. Input layer consists of 10 nodes, and an output layer consists of 1 node. The inputs are the burst times for 10 processes, shown in Table 1, that are randomly generated. The output is the estimated waiting time value.

Mathematically, a hidden or output unit operates as follows :

$$y_j = f\left(\sum_i w_{ji}x_i + b_j\right), i = 1,2,\dots,m, j = 1,2,\dots,n \quad (1)$$

where *m* is the number of the inputs, *n* is the number of the outputs, $y_j$ is transformed output by the *j* th hidden or output node, *f* is an activation function, $w_{ji}$ is the synaptic weight from the *i*th node to *j*th node, $x_i$ is an input node, $b_j$ is bias at *j*th node.

50 column input vectors are used with the dimension (10x1), and 50 outputs are derived as a scalar value. In MATLAB simulation of this case study, inputs are given as a matrix of size (50x10) and the output is obtained as (50x1) column vector for 50 queues of 10 processes.

$$\begin{aligned} x_p &= [x_1, x_2, \dots, x_{10}]^T, \qquad p = 1,2,\dots,50 \\ w &= [w_1, w_2, \dots, w_{10}]^T \\ y_p &= f(w_i x_i) = f(w^T x), \qquad p = 1,2,\dots,50 \end{aligned} \quad (2)$$

where p is a set number of 10 processes given in Table 1., components of input vector, $x_p$, are burst times of processes for *p*th set and an output $y_p$ is a waiting time for the *p*th set.

## 4. PROPOSED WORK

In this study, we randomly generated burst times for 10 processes as given in Table 1 since there is not a database including real burst time values. On the other hand, this situation does not effect the accuracy of the results of the proposed network because once we train the network it will work for every value. Then, we used these values to find the waiting time of each set according to FCFS, LCFS and SJF policies as given in Table 2. These obtained results are used to train the designed network model under three different training algorithms (Levenberg - Marquardt Algorithm, Conjugate - Gradient Algorithm with Polak-Ribiere updates and Gradient – Descent Algorithm) as 70% for training, 15% for validation, and 15% for testing. The node number in hidden layer is applied as three different values (10, 20 and 30) for each training algorithm to see the optimum result. Hence, 9 different estimated results are calculated for each scheduling policy. The block diagram of the proposed system is given in Figure 1. Table 3 shows the estimated results of 9 distinct neural network model for FCFS policy for each set. Table 4 shows the estimated results of 9 distinct neural network model for LCFS policy. Table 5 shows the estimated results of 9 distinct neural network model for SJF policy.

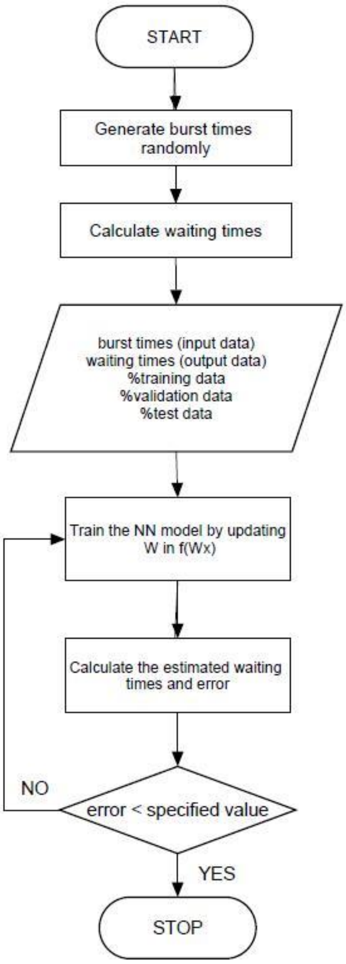


Figure 1. The block diagram of the proposed system

## 5. RESULTS AND ANALYSIS

We analyzed three different CPU scheduling algorithm by intending to design a proper neural network model. Hence, we used three different training algorithms and the neuron number of the hidden layer of the network is taken into account for three different values to see the optimum results of the neural network model. Therefore, we get 9 results for each scheduling policy and 27 results totally. Table 6. shows the average difference values of the results of the scheduling policies and the designed neural network for process sets.

Table 1. Burst Times of processes for 50 Different Process Set

| PROCESS SET | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 6 | 8 | 17 | 4 | 22 | 5 | 24 | 19 | 37 | 20 |
| 2 | 11 | 7 | 26 | 30 | 32 | 15 | 9 | 10 | 16 | 8 |
| 3 | 9 | 11 | 17 | 18 | 24 | 46 | 12 | 14 | 28 | 10 |
| 4 | 42 | 47 | 16 | 52 | 38 | 14 | 22 | 21 | 16 | 36 |
| 5 | 85 | 32 | 43 | 50 | 17 | 18 | 21 | 7 | 12 | 24 |
| 6 | 12 | 14 | 28 | 10 | 37 | 60 | 64 | 12 | 35 | 10 |
| 7 | 22 | 21 | 16 | 36 | 25 | 5 | 12 | 22 | 5 | 24 |
| 8 | 21 | 7 | 12 | 24 | 13 | 44 | 35 | 32 | 15 | 9 |
| 9 | 18 | 19 | 20 | 24 | 36 | 38 | 45 | 22 | 21 | 55 |
| 10 | 12 | 38 | 14 | 22 | 21 | 32 | 33 | 90 | 77 | 54 |
| 11 | 25 | 17 | 18 | 21 | 58 | 38 | 42 | 41 | 82 | 37 |
| 12 | 34 | 37 | 60 | 64 | 65 | 24 | 76 | 55 | 37 | 19 |
| 13 | 8 | 25 | 5 | 12 | 17 | 42 | 34 | 85 | 24 | 23 |
| 14 | 11 | 13 | 44 | 35 | 48 | 16 | 32 | 48 | 64 | 11 |
| 15 | 14 | 9 | 22 | 95 | 37 | 48 | 14 | 42 | 75 | 33 |
| 16 | 26 | 45 | 53 | 14 | 33 | 16 | 36 | 80 | 8 | 17 |
| 17 | 98 | 12 | 52 | 54 | 36 | 12 | 24 | 49 | 48 | 59 |
| 18 | 5 | 82 | 89 | 37 | 71 | 35 | 10 | 42 | 25 | 80 |
| 19 | 46 | 11 | 12 | 55 | 42 | 5 | 24 | 15 | 62 | 41 |
| 20 | 34 | 25 | 32 | 35 | 37 | 14 | 22 | 5 | 20 | 6 |
| 21 | 76 | 28 | 74 | 52 | 9 | 41 | 32 | 15 | 8 | 11 |
| 22 | 77 | 75 | 88 | 90 | 51 | 40 | 24 | 46 | 10 | 9 |
| 23 | 90 | 37 | 20 | 68 | 87 | 39 | 38 | 14 | 36 | 42 |
| 24 | 88 | 42 | 23 | 37 | 85 | 62 | 17 | 18 | 24 | 85 |
| 25 | 15 | 90 | 84 | 61 | 51 | 40 | 37 | 60 | 10 | 12 |
| 26 | 25 | 96 | 16 | 36 | 42 | 84 | 25 | 5 | 24 | 18 |
| 27 | 26 | 14 | 12 | 24 | 17 | 55 | 13 | 44 | 60 | 27 |
| 28 | 28 | 34 | 35 | 10 | 24 | 26 | 36 | 38 | 50 | 35 |
| 29 | 32 | 55 | 5 | 24 | 38 | 61 | 21 | 32 | 34 | 65 |
| 30 | 45 | 52 | 64 | 84 | 31 | 42 | 56 | 77 | 41 | 20 |
| 31 | 48 | 17 | 9 | 24 | 28 | 39 | 45 | 51 | 11 | 80 |
| 32 | 91 | 14 | 42 | 37 | 42 | 22 | 24 | 41 | 14 | 52 |
| 33 | 84 | 18 | 85 | 85 | 49 | 21 | 25 | 18 | 26 | 65 |
| 34 | 65 | 60 | 12 | 64 | 57 | 7 | 64 | 46 | 98 | 80 |
| 35 | 12 | 5 | 18 | 19 | 77 | 19 | 38 | 14 | 5 | 25 |
| 36 | 26 | 44 | 27 | 28 | 64 | 38 | 36 | 18 | 46 | 13 |
| 37 | 36 | 38 | 35 | 39 | 60 | 17 | 25 | 60 | 34 | 9 |
| 38 | 5 | 32 | 21 | 7 | 97 | 94 | 26 | 55 | 76 | 45 |
| 39 | 22 | 38 | 22 | 34 | 45 | 27 | 19 | 84 | 77 | 12 |
| 40 | 27 | 19 | 20 | 21 | 48 | 24 | 64 | 55 | 90 | 82 |
| 41 | 91 | 10 | 25 | 5 | 37 | 9 | 9 | 26 | 88 | 11 |
| 42 | 45 | 9 | 26 | 44 | 33 | 18 | 42 | 61 | 15 | 16 |
| 43 | 65 | 51 | 36 | 38 | 36 | 52 | 85 | 30 | 7 | 12 |
| 44 | 37 | 87 | 21 | 32 | 71 | 22 | 12 | 64 | 11 | 15 |
| 45 | 56 | 85 | 9 | 77 | 42 | 24 | 84 | 78 | 47 | 36 |
| 46 | 78 | 51 | 42 | 51 | 37 | 98 | 66 | 41 | 32 | 40 |
| 47 | 87 | 42 | 10 | 24 | 26 | 36 | 38 | 50 | 14 | 11 |
| 48 | 34 | 17 | 24 | 38 | 61 | 21 | 32 | 34 | 65 | 24 |
| 49 | 16 | 42 | 23 | 37 | 38 | 61 | 21 | 75 | 39 | 50 |
| 50 | 99 | 90 | 51 | 40 | 24 | 46 | 10 | 9 | 28 | 37 |

**Remark 1.** This case study is applied for 50 process sets including 10 processes whose burst times are randomly generated. It should be noted that, the proposed model can be applied to process set of any length.

**Remark 2.** In this case study, 9 different neural network models are designed to analyze the results of the network for each scheduling policy. Among these models, the best solution is obtained by the network with 10 hidden nodes under L-M algorithm for FCFS policy. The best solution for LCFS policy is obtained by the network with 10 hidden nodes under G-D algorithm. The best solution for SJF policy is obtained by the network with 10 hidden nodes under L-M algorithm. The regression graphics of these network models are given by Figure 3, Figure 3, and Figure 4, respectively. The regression values are obtained to be very close to 1 which is a desired case for modeling by NNs.

Table 2. Waiting Times of Process Sets Under FCFS, LCFS, SJF Scheduling Policies

| PROCESS SET | WAITING TIME UNDER FCFS | WAITING TIME UNDER LCFS | WAITING TIME UNDER SJF |
|---|---|---|---|
| 1 | 538 | 920 | 455 |
| 2 | 800 | 676 | 495 |
| 3 | 792 | 909 | 578 |
| 4 | 1548 | 1188 | 991 |
| 5 | 1868 | 913 | 819 |
| 6 | 1152 | 1386 | 747 |
| 7 | 924 | 768 | 597 |
| 8 | 898 | 1010 | 627 |
| 9 | 1130 | 1552 | 1015 |
| 10 | 1231 | 2306 | 1090 |
| 11 | 1345 | 2066 | 1197 |
| 12 | 2202 | 2037 | 1601 |
| 13 | 928 | 1547 | 691 |
| 14 | 1281 | 1617 | 950 |
| 15 | 1500 | 2001 | 1033 |
| 16 | 1554 | 1398 | 911 |
| 17 | 2112 | 1884 | 1347 |
| 18 | 2180 | 2104 | 1357 |
| 19 | 1310 | 1507 | 864 |
| 20 | 1277 | 793 | 720 |
| 21 | 2081 | 1033 | 880 |
| 22 | 3038 | 1552 | 1470 |
| 23 | 2423 | 1816 | 1455 |
| 24 | 2295 | 2034 | 1396 |
| 25 | 2465 | 1675 | 1303 |
| 26 | 1976 | 1363 | 944 |
| 27 | 1066 | 1562 | 860 |
| 28 | 1287 | 1557 | 1157 |
| 29 | 1502 | 1801 | 1154 |
| 30 | 2459 | 2149 | 1773 |
| 31 | 1319 | 1849 | 1010 |
| 32 | 1913 | 1498 | 1147 |
| 33 | 2471 | 1813 | 1377 |
| 34 | 2228 | 2749 | 1781 |
| 35 | 996 | 1092 | 564 |
| 36 | 1605 | 1455 | 1137 |
| 37 | 1704 | 1473 | 1167 |
| 38 | 1615 | 2507 | 1155 |
| 39 | 1495 | 1925 | 1098 |
| 40 | 1389 | 2661 | 1324 |
| 41 | 1492 | 1307 | 635 |
| 42 | 1423 | 1358 | 942 |
| 43 | 2183 | 1525 | 1239 |
| 44 | 1986 | 1362 | 974 |
| 45 | 2470 | 2372 | 1707 |
| 46 | 2599 | 2225 | 1887 |
| 47 | 1835 | 1207 | 942 |
| 48 | 1456 | 1694 | 1168 |
| 49 | 1549 | 2069 | 1328 |
| 50 | 2588 | 1318 | 1172 |

**Remark 3.** The optimum results of the neural network models are obtained when the number of processes and the number of hidden nodes are equal.

**Remark 4.** The minimum waiting time is provided by SJF policy for each process set and the results of the designed neural network model show the same tendency.

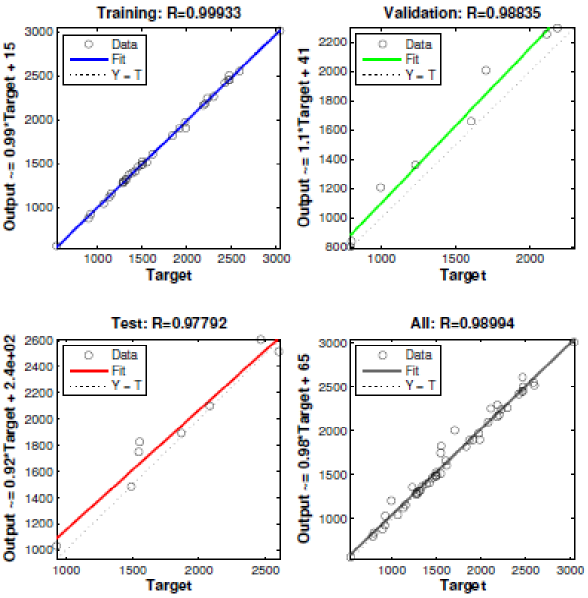


Figure 2. The regression coefficients of NN model with 10 hidden nodes under L-M Alg. For FCFS policy

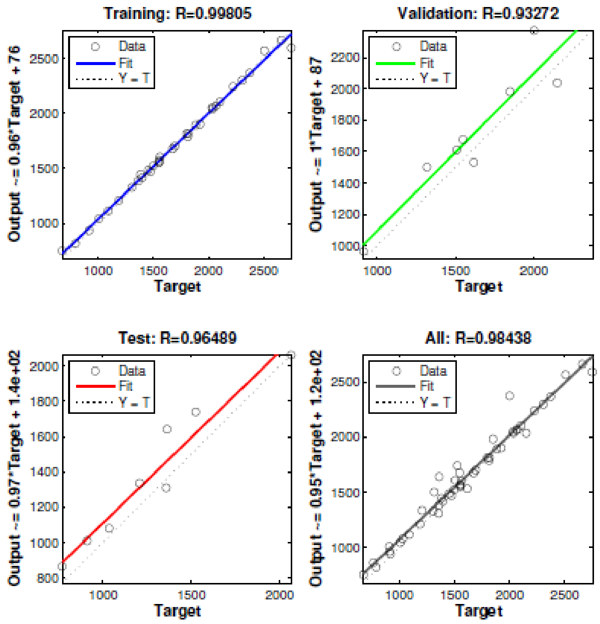


Figure 3. The regression coefficients of NN model with 10 hidden nodes under G-D Alg. For LCFS policy

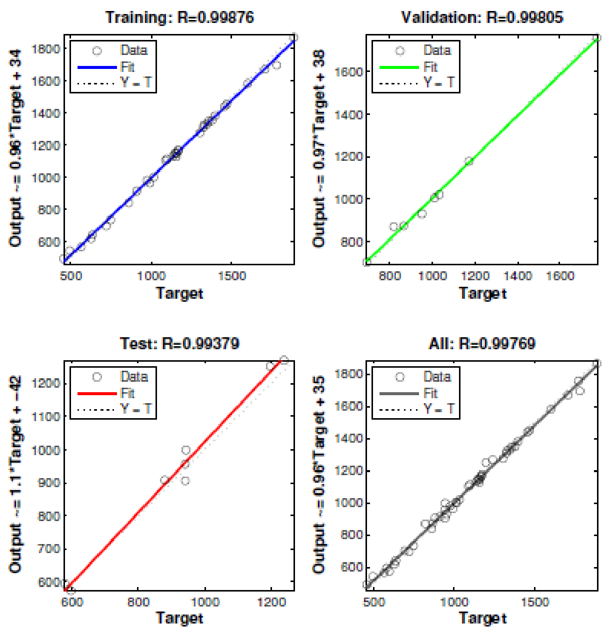


Figure 4. The regression coefficients of NN model with 10 hidden nodes under L-M Alg. For SJF policy

## 6. CONCLUSIONS

Minimum waiting time is one of the characteristics of efficient CPU scheduling policies. Selecting an appropriate scheduling policy according to a minimum waiting time is a fundamental step to utilize CPU efficiently. The aim of this paper is to explore a neural network approach to build a CPU scheduling model to generate waiting time. This study establishes neural networks as a method for modeling scheduling policies by showing that we can model three widely accepted policies; First Come-First Served (FCFS) scheduling policy, Last-Come, First-Served (LCFS) scheduling policy and Shortest-Job-First (SJF) scheduling policy. For this purpose, different training algorithms and various number of hidden neurons are applied in the process of designing neural network model. The optimum results have been acquired when the number of processes and hidden neurons are equal. In our proposed neural network system, the training results have shown that the L-M algorithm best fits for FCFS and SJF policies, and G-D training algorithm best fits for LCFS policy. The proposed model can be applied to the process set of any length. This paper proved that applying neural network approach to CPU utilization can be an alternative successful method to obtain waiting time criterion.

## REFERENCES

[1] A. Silberschatz, P. B. Galvin, G. Gagne, **Operating System Concepts, Sixth Edition,** John Wiley and Sons Inc., New York, A.B.D., 2004.

[2] M. Milenkovic, **Operating Systems Concepts and Design, Second Edition,** McGraw-Hill, Computer Science Series, 2010.

[3] K. Sukhija, N. Aggarwal, M. Jindal, "An Optimized Approach to CPU Scheduling Algorithm:Min-Max", *Journal Of Emerging Technologies inWeb Intelligence*, 6, 420-428, 2014.

[4] M. Saini, N. Kumar, "A Survey on CPU Scheduling", *International Journal Of Research In Computer Applications and Robotics*, 3, 7-12, 2015.

[5] S.Jain and S.Jain, "A Review study on the CPU Scheduling Algorithms", *International Journal of Advanced Research in Computer and Communication Engineering* , 5, no.8, 2016.

[6] Imran Qureshi, "CPU Scheduling Algorithms: A Survey", *Int. J. Advanced Networking and Applications*, 5, no.4, 1968-1973, 2014.

[7] H. Arora, D. Arora, B.Goel, P. Jain, "An Improved CPU Scheduling Algorithm", *International Journal of Applied Information Systems*, 6, 7-9, 2013.

[8] N. Goel, R.B. Garg, "A Comparative Study of CPU Scheduling Algorithms", *International Journal of Graphics and Image Processing*, 2, 245-251, 2012.

[9] S. Bibi, F. Azam, Y. Chaudhry, "Combinatory CPU Scheduling Algorithm", *International Journal of Computer Science and Information Security*,8, 39-43, 2010.

[10] S. Suranauwarat, "The Design and Development of a CPU Scheduling Algorithm Simulator", *In Proc. of 12th WSEAS International Conference on Applied Computer Science*, 164-170, 2012.

[11] M. Sindhu, R. Rajkamal, P. Vigneshwaran, "An Optimum Multilevel CPU Scheduling Algorithm", *International Conference on Advances in Computer Engineering*, 90-94, 2010.

[12] Raman, P.K.Mittal, "An Efficient Dynamic Round Robin CPU Scheduling Algorithm", *International Journal of Advanced Research in Computer Science and Software Engineering*, vol.4, no.5, 2014.

[13] N.Goel, R.B.Garg, "Simulation of an Optimum Multilevel Dynamic Round Robin Scheduling Algorithm", *International Journal of Computer Applications*, 76, 7, 42-46, 2013.

[14] S. Almakdi, M. Aleisa, M. Alshehri, "Simulation and Performance Evaluation of CPU Scheduling Algorithms", *International Journal of Advanced Research in Computer and Communication Engineering*, 4, 2015.

[15] S. Yang, D. Wang, T. Chai, G. Kendall, "An improved constraint satisfaction adaptive neural network for job-shop scheduling", *Journal of Scheduling*, 13, 17-38, 2010.

[16] M.D. Richard, R.P. Lippmann, "Neural Network Classifiers Estimate Bayesian a posteriori Probabilities", *Neural Computation*, 3, 461-483, 1991.

[17] A.A. Collister, O. Lahav, "ANNz: Estimating Photometric Redshifts Using Artificial Neural Networks", *Publications of the Astronomical Society of the Pacific*, 116, 345-351, 2004.

[18] M. Kumar, N. S. Raghuwanshi, R. Singh, W. Wallender, W. Pruitt, "Estimating Evapotranspiration using Artificial Neural Network", *Journal of Irrigation And Drainage Engineering*, 128, 224-233, 2002.

[19] M. Uzunoglu, C. Kocatepe, R. Yumurtaci, "An Artificial Neural Network Based Preestimation Filter For Bad Data Detection Identification and Elimination In State Estimation", *Mathematical and Computational Applications*, 1, 159-164, 1996.

[20] S. Haykin, **Neural Networks and Learning Machines (3rd Edition)**, Prentice Hall., 2009.

[21] K. Levenberg, “A method for the solution of certain problems in least squares”, *Quarterly of Applied Mathematics*, 5, 164-168, 1944.

[22] D. Marquardt, “An algorithm for least-squares estimation of nonlinear parameters”, *SIAM Journal on Applied Mathematics*, 11, 431-441, 1963.

[23] H. Yuand, B. M. Wilamowski, **Intelligent Systems, Chapter 12, LevenbergMarquard Training,** CRC Press., 2011.

[24] M. Hestenes, E. Stiefel, “Methods of Conjugate Gradients for Solving Linear Systems”, *Journal of Research of the National Bureau of Standards*, 49, 409-436, 1952.

[25] M. Avriel, **Nonlinear Programming: Analysis and Methods**, Dover Publishing., 2003.

Table 3. All Results of 9 NN model for FCFS Policy

| Process Set | 10-Node L-M Alg. | 20-Node L-M Alg. | 30-Node L-M Alg. | 10-Node C-G Alg. | 20-Node C-G Alg. | 30-Node C-G Alg. | 10-Node G-D Alg. | 20-Node G-D Alg. | 30-Node G-D Alg. |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 563.3358 | 515.7816 | 847.3712 | 590.5264 | 893.4663 | 538.0000 | 537.1191 | 628.6688 | 538.0000 |
| 2 | 838.3434 | 768.6529 | 1196.8678 | 820.7253 | 800.0000 | 800.0000 | 799.4765 | 875.6751 | 800.0000 |
| 3 | 801.1347 | 760.7168 | 790.3533 | 826.3715 | 792.0000 | 792.0000 | 859.8623 | 604.8718 | 792.0000 |
| 4 | 1648.1870 | 1533.8790 | 1547.0681 | 1565.7740 | 1548.0000 | 1548.0000 | 1546.6650 | 1564.9830 | 1548.0000 |
| 5 | 1891.7890 | 1849.8259 | 1867.5340 | 1931.4770 | 1868.0000 | 1170.6650 | 2038.5440 | 1615.4410 | 1868.0000 |
| 6 | 1158.0150 | 1129.4895 | 1773.8321 | 1191.9850 | 1579.6980 | 1343.3420 | 1151.4450 | 1188.9230 | 1380.3650 |
| 7 | 925.8820 | 900.5603 | 1552.5876 | 967.9123 | 945.5376 | 696.6961 | 912.6073 | 1100.5910 | 866.6014 |
| 8 | 878.7025 | 868.1511 | 894.6057 | 936.6642 | 898.0000 | 898.0000 | 896.8202 | 967.0436 | 845.9406 |
| 9 | 1122.4230 | 1115.0268 | 1128.2988 | 1143.2050 | 1285.4900 | 1130.0000 | 1229.3670 | 955.0412 | 1130.0000 |
| 10 | 1350.1810 | 1228.8870 | 1230.2027 | 1267.4680 | 1231.0000 | 1231.0000 | 1231.2750 | 886.9156 | 1653.1440 |
| 11 | 1369.5910 | 1332.8506 | 1344.3096 | 1353.6780 | 1345.0000 | 1345.0000 | 1301.6430 | 1367.3590 | 1345.0000 |
| 12 | 2178.9060 | 2188.1394 | 2201.9207 | 2224.1190 | 2386.4820 | 2390.1720 | 2200.7600 | 2216.2720 | 2202.0000 |
| 13 | 1030.9370 | 924.7103 | 925.7904 | 1016.1730 | 997.0957 | 928.0000 | 982.5115 | 986.9197 | 928.0000 |
| 14 | 1294.7680 | 1272.3975 | 1279.9523 | 1315.7240 | 1281.0000 | 1281.0000 | 1218.0370 | 1146.3340 | 1281.0000 |
| 15 | 1523.5930 | 1525.6027 | 1499.4663 | 1495.3830 | 1524.2570 | 1500.0000 | 1395.9510 | 1139.2190 | 1500.0000 |
| 16 | 1706.7700 | 1546.1120 | 2155.9733 | 1688.6860 | 1425.2770 | 2747.7540 | 1553.0010 | 1587.1460 | 1554.0000 |
| 17 | 2253.7120 | 2101.4670 | 1571.4588 | 2245.8990 | 2112.0000 | 2112.0000 | 2111.2510 | 2118.7970 | 2112.0000 |
| 18 | 2160.2430 | 2148.2474 | 2180.0743 | 2163.1520 | 3429.5310 | 1445.4230 | 2579.0410 | 2186.7520 | 2180.0000 |
| 19 | 1318.0530 | 1297.8887 | 726.5543 | 1316.5360 | 1310.0000 | 1310.0000 | 1322.0800 | 1320.4350 | 1310.0000 |
| 20 | 1278.9840 | 1179.7643 | 1274.8862 | 1317.6240 | 1272.5610 | 1277.0000 | 1276.3260 | 1308.6990 | 1277.0000 |
| 21 | 2099.7710 | 2063.4917 | 1427.3614 | 2002.4420 | 2081.8180 | 2669.8810 | 2194.7050 | 2067.7820 | 2217.4310 |
| 22 | 3008.0080 | 2764.0809 | 2795.8841 | 2758.3790 | 3038.0000 | 3038.0000 | 3037.6430 | 3008.8620 | 2272.7320 |
| 23 | 2417.6530 | 2420.0576 | 1668.5868 | 2281.8380 | 2423.0000 | 2423.0000 | 2422.7030 | 2435.9260 | 2336.6040 |
| 24 | 2260.8330 | 2176.5101 | 2294.7961 | 2263.1910 | 2640.2400 | 2295.0000 | 2394.6240 | 2705.3370 | 2295.0000 |
| 25 | 2607.5340 | 2530.3026 | 2267.3111 | 2474.3310 | 2526.4860 | 2465.0000 | 2464.1680 | 2380.6760 | 2803.5360 |
| 26 | 1970.4360 | 1983.5091 | 1975.7587 | 1970.9870 | 1976.0000 | 1976.0000 | 2011.2540 | 1974.5730 | 1976.0000 |
| 27 | 1042.8250 | 1056.4903 | 1063.6889 | 1119.7340 | 1066.0000 | 883.2344 | 1065.5960 | 1009.4500 | 1066.0000 |
| 28 | 1281.0590 | 1310.6423 | 1285.3303 | 1410.6940 | 1287.0000 | 1658.4120 | 1286.7280 | 1315.6350 | 1287.0000 |
| 29 | 1488.5740 | 1378.8923 | 1519.2737 | 1517.4670 | 1502.0000 | 2064.3360 | 1582.5690 | 2031.2370 | 1502.0000 |
| 30 | 2447.4670 | 2447.5120 | 2458.7987 | 2451.8590 | 2459.0000 | 2459.0000 | 2269.1700 | 2230.5780 | 2459.0000 |
| 31 | 1317.3540 | 1321.3305 | 1319.4513 | 1311.5290 | 1463.7700 | 1319.0000 | 1278.6230 | 1336.2790 | 1319.0000 |
| 32 | 1898.9960 | 1904.0010 | 1912.5567 | 1916.0770 | 1913.0000 | 1913.0000 | 1912.6100 | 1916.2410 | 1240.8950 |
| 33 | 2449.3060 | 2703.5959 | 2470.6856 | 2362.3550 | 2471.0000 | 2471.0000 | 2470.8260 | 1945.8580 | 2471.0000 |
| 34 | 2243.6540 | 2036.2273 | 2207.9979 | 2172.0220 | 2228.0000 | 2228.0000 | 2287.2310 | 2223.3210 | 2228.0000 |
| 35 | 1006.2010 | 992.1551 | 2200.5614 | 915.8697 | 996.0000 | 996.0000 | 996.1771 | 1050.6350 | 1239.6700 |
| 36 | 1658.4790 | 1592.5319 | 1604.2166 | 1622.9930 | 1605.0000 | 1781.7750 | 1603.9660 | 1630.2560 | 1605.0000 |
| 37 | 1786.0190 | 1699.0770 | 1703.1819 | 1770.9780 | 1704.0000 | 1704.0000 | 1703.2680 | 1828.3580 | 1441.1900 |
| 38 | 1605.3800 | 1612.8619 | 1615.4448 | 1640.1610 | 1615.0000 | 1615.0000 | 1632.2480 | 1629.0180 | 2390.8330 |
| 39 | 1488.8620 | 1390.4569 | 1494.0077 | 1524.7790 | 1495.0000 | 1495.0000 | 1494.4050 | 1548.7000 | 935.7278 |
| 40 | 1394.1040 | 1381.1707 | 1389.3388 | 1431.3240 | 1389.0000 | 1389.0000 | 1389.3830 | 1406.1870 | 1905.4430 |
| 41 | 1484.6870 | 1503.7724 | 1162.6850 | 2105.3590 | 1492.0000 | 1353.1930 | 1491.9060 | 1503.2430 | 1492.0000 |
| 42 | 1410.5000 | 1390.8623 | 1422.7309 | 1494.7570 | 1423.0000 | 1423.0000 | 1314.8920 | 1469.1090 | 1469.1090 |
| 43 | 2293.8210 | 2158.3797 | 2183.7101 | 2179.7600 | 2183.0000 | 2183.0000 | 2236.0400 | 3195.1270 | 3195.1270 |
| 44 | 1900.8330 | 1894.1307 | 1985.6707 | 1795.9260 | 2442.7040 | 2797.4250 | 1985.7230 | 2001.3150 | 2001.3150 |
| 45 | 2500.8090 | 2228.6573 | 2469.7746 | 2474.2020 | 2470.0000 | 2470.0000 | 2469.2640 | 2427.5200 | 2427.5200 |
| 46 | 2515.0930 | 2358.3229 | 2599.1299 | 2504.9000 | 2754.5030 | 2682.0910 | 2599.4560 | 2580.8710 | 2580.8710 |
| 47 | 1818.2540 | 1820.8530 | 1836.4277 | 1963.4790 | 1835.0000 | 2757.6190 | 1705.9680 | 1844.4360 | 1844.4360 |
| 48 | 1463.6340 | 1452.8767 | 1467.5955 | 1472.1140 | 1456.0000 | 1456.0000 | 1455.2030 | 1479.1040 | 1479.1040 |
| 49 | 1512.3280 | 1559.6771 | 1567.9290 | 1568.0700 | 1549.0000 | 1549.0000 | 1549.0750 | 1578.7470 | 1578.7470 |
| 50 | 2549.8920 | 2584.8221 | 2587.7862 | 2529.7380 | 2588.0000 | 2110.0730 | 2698.6620 | 2564.8170 | 2564.8170 |

Table 4. All Results of 9 NN model for LCFS Policy

| Process Set | 10-Node L-M Alg. | 20-Node L-M Alg. | 30-Node L-M Alg. | 10-Node C-G Alg. | 20-Node C-G Alg. | 30-Node C-G Alg. | 10-Node G-D Alg. | 20-Node G-D Alg. | 30-Node G-D Alg. |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 937.8915 | 920.0000 | 944.2013 | 950.5981 | 898.0092 | 938.0062 | 929.3108 | 920.1202 | 920.0000 |
| 2 | 722.6814 | 676.0000 | 743.7531 | 675.9177 | 664.2900 | 849.0036 | 675.8433 | 579.6319 | 676.0000 |
| 3 | 990.7845 | 1087.1640 | 944.5030 | 908.9595 | 898.3317 | 909.0000 | 899.9115 | 909.1029 | 909.0000 |
| 4 | 1210.9230 | 1264.9470 | 900.4425 | 1117.82 | 1204.0420 | 1188.0000 | 1187.9820 | 1188.0780 | 1188.0000 |
| 5 | 962.9023 | 739.9513 | 536.2939 | 1094.82 | 921.5349 | 1072.4360 | 953.0079 | 1180.0810 | 913.0000 |
| 6 | 1444.7580 | 1386.0000 | 1278.7950 | 1385.642 | 1703.3030 | 1386.0000 | 1512.6050 | 1357.7190 | 1386.0000 |
| 7 | 842.8548 | 768.0000 | 677.4106 | 767.9866 | 700.1255 | 768.0000 | 767.9472 | 768.0173 | 768.0000 |
| 8 | 1044.1680 | 1010.0000 | 760.0232 | 1010.002 | 773.7024 | 1010.0000 | 1009.9980 | 1010.2100 | 1010.0000 |
| 9 | 1551.4660 | 1433.3660 | 1398.7040 | 1552.008 | 1547.9830 | 1552.0000 | 1551.9940 | 1552.0500 | 1552.0000 |
| 10 | 2295.6850 | 2350.7070 | 2429.8920 | 2305.917 | 2598.1290 | 2306.0000 | 2406.0000 | 1839.9190 | 1893.0530 |
| 11 | 2062.1480 | 2078.8760 | 1870.3700 | 2065.873 | 2044.4240 | 2066.0000 | 2065.9890 | 2066.0290 | 2066.0000 |
| 12 | 2037.1910 | 2037.0000 | 2122.0230 | 2036.954 | 2026.6050 | 2037.0000 | 2037.0000 | 2348.5980 | 2015.3250 |
| 13 | 1676.3520 | 1547.0000 | 1521.0950 | 1776.283 | 1506.1980 | 1547.0000 | 1367.0580 | 1547.1480 | 1558.1000 |
| 14 | 1532.3060 | 1617.0000 | 1388.9390 | 1616.763 | 1576.1500 | 1617.0000 | 1616.9290 | 1759.7290 | 1617.0000 |
| 15 | 2171.8190 | 2001.0000 | 2211.7090 | 2000.955 | 1993.4600 | 2001.0000 | 2000.9470 | 2000.8090 | 2772.3960 |
| 16 | 1415.3150 | 1398.0000 | 1441.7780 | 1397.835 | 1381.3420 | 1961.7670 | 1297.9090 | 1416.5160 | 1404.1750 |
| 17 | 1891.3920 | 1884.0000 | 1716.5550 | 1883.963 | 1881.9550 | 1842.7660 | 1883.9920 | 2183.6230 | 1715.1370 |
| 18 | 2103.3050 | 2104.0000 | 2149.6300 | 2103.985 | 2097.0860 | 2104.0000 | 2094.1170 | 2103.9820 | 2104.0000 |
| 19 | 1609.5160 | 1507.0000 | 958.6304 | 1506.84 | 1508.4670 | 1507.0000 | 1506.9660 | 1506.8430 | 1507.0000 |
| 20 | 818.0318 | 793.0000 | 717.0918 | 792.9026 | 795.0093 | 793.0000 | 792.9388 | 747.3302 | 793.5628 |
| 21 | 1078.5070 | 1033.0000 | 944.8952 | 1032.889 | 1142.4810 | 1033.0000 | 1033.0240 | 1033.0540 | 1033.0000 |
| 22 | 1572.1660 | 1552.0000 | 1836.7400 | 1560.042 | 1606.6980 | 302.1984 | 1552.0490 | 1551.9760 | 1552.0000 |
| 23 | 1811.9360 | 1799.3860 | 1568.7750 | 1815.932 | 1941.8410 | 1454.0360 | 2171.6100 | 1815.8970 | 1397.8940 |
| 24 | 2049.8840 | 2331.4780 | 1694.7330 | 2022.578 | 2057.8220 | 2034.0000 | 2176.5210 | 2034.0950 | 2199.5180 |
| 25 | 1675.4540 | 1887.0800 | 844.0410 | 1767.331 | 1532.7340 | 1675.0000 | 1886.9170 | 2124.8730 | 1675.0000 |
| 26 | 1440.8450 | 1363.0000 | 1238.3680 | 1362.981 | 1382.5280 | 1363.0000 | 1362.9120 | 1363.0130 | 1363.0000 |
| 27 | 1602.9220 | 1562.0000 | 1424.7830 | 1561.713 | 1521.4210 | 1537.8680 | 1663.0000 | 1562.0930 | 1922.9870 |
| 28 | 1560.1090 | 1557.0000 | 1491.3380 | 1556.757 | 1542.1900 | 1711.9570 | 1556.9740 | 1557.1350 | 1700.2240 |
| 29 | 1813.8380 | 1801.0000 | 1677.2940 | 1801.04 | 1797.7610 | 1801.0000 | 1801.0090 | 1851.1360 | 2226.5680 |
| 30 | 2035.8580 | 1549.2360 | 2621.0260 | 2215.179 | 2148.2770 | 2149.0000 | 2148.9830 | 2148.8380 | 2149.0000 |
| 31 | 1981.7090 | 1849.0000 | 2573.1440 | 1848.999 | 1867.3540 | 1849.0000 | 1849.0000 | 1849.6600 | 1849.0000 |
| 32 | 1522.4950 | 1359.0160 | 1672.6750 | 1497.876 | 1472.0990 | 1498.0000 | 1498.0080 | 1849.1440 | 1160.1360 |
| 33 | 1786.4390 | 1602.9220 | 2038.0500 | 1812.931 | 1796.0060 | 1909.4460 | 1804.0900 | 2028.3540 | 1813.0000 |
| 34 | 2590.6770 | 2608.7580 | 2218.4120 | 2749.078 | 2748.4050 | 2749.0000 | 2657.9810 | 2748.9560 | 2005.1340 |
| 35 | 1114.8430 | 1092.0000 | 873.7934 | 1477.199 | 900.7235 | 889.4609 | 1091.9250 | 1091.9560 | 1092.0000 |
| 36 | 1485.6260 | 1455.0000 | 1556.8000 | 1583.272 | 1433.3610 | 1455.0000 | 1454.9650 | 1455.0620 | 1455.0000 |
| 37 | 1469.2550 | 1473.0000 | 1369.7700 | 1472.849 | 1663.2250 | 1607.8880 | 1472.9790 | 1472.9020 | 1308.3270 |
| 38 | 2565.1590 | 2507.0000 | 1961.9090 | 2506.871 | 2047.9320 | 2507.0000 | 2506.9630 | 2507.0040 | 2507.0000 |
| 39 | 1900.7610 | 1775.3960 | 2027.1120 | 1861.454 | 2225.8970 | 1925.0000 | 1922.4980 | 1924.9950 | 1925.0000 |
| 40 | 2660.6400 | 2670.0000 | 2059.4970 | 2661.005 | 2913.6340 | 2661.0000 | 2661.0120 | 2661.0230 | 2661.0000 |
| 41 | 1327.0480 | 1387.4500 | 589.2895 | 1520.68 | 1304.6320 | 1334.3590 | 1406.9810 | 1307.2010 | 1307.0000 |
| 42 | 1309.4590 | 1358.0000 | 1142.3020 | 1447.833 | 1343.8010 | 1238.6680 | 1358.0140 | 1357.8780 | 1358.0000 |
| 43 | 1670.4200 | 1525.0000 | 1187.8210 | 1363.776 | 1417.5630 | 1525.0000 | 1722.8320 | 1524.8760 | 1525.0000 |
| 44 | 1383.6860 | 1362.0000 | 1286.7210 | 1362.004 | 1368.9350 | 1477.4980 | 1609.4050 | 1362.1000 | 1362.0000 |
| 45 | 2364.0810 | 2372.0000 | 2274.6680 | 2372.053 | 2367.2710 | 2372.0000 | 2342.2400 | 2371.8970 | 2372.0000 |
| 46 | 2239.4360 | 2225.0000 | 2876.7390 | 2224.981 | 2268.8140 | 2225.0000 | 2224.9670 | 2224.9450 | 2225.0000 |
| 47 | 1335.3340 | 1207.0000 | 724.4970 | 1294.51 | 1000.1670 | 1128.3140 | 1382.2970 | 1206.9940 | 1207.0000 |
| 48 | 1700.3970 | 1790.3340 | 1460.1070 | 1693.817 | 1666.1420 | 1694.0000 | 1693.9620 | 1679.5140 | 1694.0000 |
| 49 | 2067.8170 | 2069.0000 | 2028.4030 | 2023.585 | 2026.8470 | 2069.0000 | 2068.9820 | 2068.9630 | 2069.0000 |
| 50 | 1431.4620 | 1361.7520 | 1338.2990 | 1317.997 | 1319.1560 | 1318.0000 | 1242.8950 | 1671.1680 | 1286.2100 |

Table 5. All Results of 9 NN model for SJF Policy

| Process Set | 10-Node L-M Alg. | 20-Node L-M Alg. | 30-Node L-M Alg. | 10-Node C-G Alg. | 20-Node C-G Alg. | 30-Node C-G Alg. | 10-Node G-D Alg. | 20-Node G-D Alg. | 30-Node G-D Alg. |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 492.5537 | 411.4987 | 421.9262 | 562.9937 | 698.1689 | 481.7583 | 481.7583 | 554.9838 | 571.0441 |
| 2 | 543.1882 | 432.4990 | 508.3105 | 499.0482 | 620.3964 | 615.5979 | 515.5979 | 505.4336 | 561.4124 |
| 3 | 595.9677 | 497.4110 | 697.4678 | 573.6829 | 680.8813 | 588.5218 | 588.5218 | 597.9939 | 639.4268 |
| 4 | 962.7445 | 1011.0640 | 817.8590 | 989.8972 | 989.6485 | 767.2904 | 810.8669 | 1390.9930 | 1067.1930 |
| 5 | 871.2825 | 785.8293 | 863.5327 | 817.0076 | 817.2058 | 833.1024 | 734.8605 | 918.9908 | 925.1859 |
| 6 | 734.2130 | 675.4574 | 747.7734 | 668.5055 | 594.8056 | 659.9185 | 677.6117 | 700.9942 | 839.1765 |
| 7 | 574.4386 | 551.1917 | 605.5168 | 695.5730 | 537.6758 | 608.9937 | 612.8375 | 696.9850 | 738.0142 |
| 8 | 619.1873 | 553.7844 | 632.7223 | 621.5709 | 622.6311 | 644.8910 | 709.1861 | 696.9909 | 706.3515 |
| 9 | 1000.4630 | 961.8336 | 1183.2620 | 1000.5890 | 1013.9910 | 1003.2740 | 1011.6100 | 1114.9920 | 1043.9230 |
| 10 | 1103.1790 | 1087.3310 | 1098.2900 | 1068.6240 | 1088.6700 | 1096.8530 | 1195.3300 | 1189.9950 | 1144.5070 |
| 11 | 1251.4030 | 1182.3440 | 1166.7310 | 1188.2880 | 1187.1860 | 1203.1510 | 1113.3650 | 1326.2540 | 1221.4000 |
| 12 | 1583.6870 | 1591.9170 | 1601.3440 | 1728.0000 | 1636.8240 | 1321.6180 | 1672.5210 | 1701.0010 | 1481.9390 |
| 13 | 702.5545 | 621.0868 | 691.8826 | 718.0733 | 689.1517 | 723.4783 | 943.0833 | 728.5265 | 845.5146 |
| 14 | 930.6376 | 901.0494 | 949.4356 | 947.4288 | 949.0155 | 964.4218 | 1016.8640 | 985.8364 | 1020.4140 |
| 15 | 1021.3630 | 1032.9920 | 1075.9460 | 1051.6730 | 1031.9230 | 1039.2100 | 891.2174 | 1055.3100 | 1107.0560 |
| 16 | 913.8994 | 894.4158 | 910.5151 | 910.8187 | 909.3431 | 926.7889 | 981.6817 | 990.9919 | 1037.8990 |
| 17 | 1331.0130 | 1420.8930 | 1348.0660 | 1335.1490 | 1297.6920 | 1276.7440 | 1356.6530 | 1732.7060 | 1721.6000 |
| 18 | 1348.6060 | 1409.0260 | 1610.0220 | 1308.5750 | 1358.3960 | 1350.0230 | 1355.0210 | 1490.9960 | 1525.7460 |
| 19 | 873.6953 | 866.5141 | 879.3327 | 864.7402 | 862.1920 | 751.6447 | 841.7512 | 893.9958 | 970.7055 |
| 20 | 696.5530 | 676.7407 | 731.2695 | 792.0228 | 717.7358 | 718.9481 | 873.7510 | 767.9837 | 795.8770 |
| 21 | 907.8507 | 880.3707 | 879.8113 | 889.6364 | 878.6407 | 896.9196 | 963.3410 | 779.9898 | 956.2853 |
| 22 | 1454.1070 | 1474.9710 | 1472.3880 | 1458.6950 | 1467.8850 | 1466.3370 | 1574.7330 | 1525.9980 | 1598.8590 |
| 23 | 1440.1470 | 1455.8440 | 1457.4890 | 1439.5860 | 1460.7250 | 1515.5380 | 1369.4030 | 1654.9970 | 1583.8190 |
| 24 | 1381.8480 | 1390.4390 | 1398.1390 | 1363.8520 | 1448.5090 | 1139.0030 | 1300.3410 | 1378.5470 | 1525.2910 |
| 25 | 1276.3700 | 1308.3720 | 1304.6120 | 1345.4210 | 1300.8230 | 1302.8890 | 939.7524 | 1402.9980 | 1260.4310 |
| 26 | 998.5577 | 949.3576 | 984.4049 | 940.4065 | 942.6237 | 937.9478 | 861.1636 | 998.9952 | 1049.4960 |
| 27 | 839.9656 | 815.5705 | 899.8967 | 853.6171 | 857.8741 | 859.9797 | 960.7563 | 828.7761 | 999.2051 |
| 28 | 1146.1890 | 1142.9240 | 1156.9800 | 1154.9740 | 1156.0930 | 1153.0060 | 1133.0360 | 1196.9990 | 1233.2610 |
| 29 | 1152.3870 | 1165.5970 | 1174.1420 | 1150.6750 | 1153.2870 | 1102.0590 | 1266.6250 | 1253.9960 | 1250.1170 |
| 30 | 1758.5260 | 1767.2630 | 2021.6820 | 1785.4070 | 1772.2720 | 1767.7740 | 1606.0670 | 1587.7610 | 1851.5880 |
| 31 | 1005.4010 | 1019.7920 | 1059.9820 | 981.6633 | 1009.1610 | 1022.9880 | 1125.9740 | 1086.6870 | 1101.3110 |
| 32 | 1148.2190 | 1134.7570 | 1147.3080 | 1138.0140 | 1146.9370 | 1154.1590 | 1274.3400 | 1240.9770 | 1222.6430 |
| 33 | 1352.2120 | 1334.9990 | 1379.0490 | 1359.4600 | 1373.4620 | 1364.0330 | 1863.9880 | 1477.0010 | 1308.9270 |
| 34 | 1696.3920 | 1818.7050 | 1343.2150 | 1779.6020 | 1550.3290 | 1772.3970 | 1551.4770 | 1980.9970 | 1835.5090 |
| 35 | 566.9703 | 491.8064 | 569.0225 | 561.4405 | 560.7351 | 592.8054 | 632.9525 | 599.5945 | 663.4965 |
| 36 | 1133.5810 | 1120.5190 | 1136.8210 | 1130.9230 | 1127.1200 | 1144.1670 | 1195.9900 | 1236.9960 | 1214.2140 |
| 37 | 1163.8940 | 1220.1170 | 1167.5240 | 1158.3880 | 1167.0830 | 1134.7290 | 1164.2550 | 1245.9970 | 1021.7730 |
| 38 | 1127.1180 | 1152.8300 | 1256.4520 | 1134.4680 | 1151.8300 | 1166.9300 | 1095.2060 | 1254.9930 | 1309.8180 |
| 39 | 1114.1670 | 1096.3070 | 1098.0490 | 1084.7650 | 1083.5040 | 1102.7260 | 1323.0720 | 1054.5340 | 1183.0110 |
| 40 | 1309.8380 | 1317.4720 | 1334.8050 | 1299.4060 | 1322.0240 | 1315.3580 | 1237.0530 | 1524.0010 | 1460.6390 |
| 41 | 641.2256 | 642.3794 | 634.3464 | 647.8603 | 633.3793 | 632.0134 | 733.9766 | 834.9984 | 760.2512 |
| 42 | 956.9164 | 892.8212 | 941.6143 | 940.4842 | 889.6043 | 953.8073 | 936.9060 | 1141.9930 | 1033.7160 |
| 43 | 1268.9700 | 1255.3170 | 1270.0200 | 1225.8510 | 1238.2740 | 1243.9630 | 1197.7730 | 1438.9980 | 1293.5380 |
| 44 | 981.0678 | 963.2432 | 983.9494 | 942.3786 | 972.1737 | 979.4640 | 898.6088 | 915.5674 | 1030.9460 |
| 45 | 1670.7840 | 1704.5120 | 1710.9470 | 1708.7000 | 1713.3280 | 1694.7600 | 1883.3770 | 1649.8680 | 1796.3070 |
| 46 | 1869.4200 | 1875.3520 | 1887.2350 | 1770.4470 | 1886.5030 | 1876.2370 | 1936.1090 | 1999.9100 | 1976.8690 |
| 47 | 906.2981 | 913.8651 | 972.1981 | 939.4036 | 941.4739 | 959.1903 | 1063.9870 | 986.0455 | 1034.6440 |
| 48 | 1168.2890 | 1277.1910 | 1167.5590 | 1162.5770 | 1167.4280 | 1171.6720 | 1326.9980 | 1267.9980 | 1238.4660 |
| 49 | 1324.6140 | 1328.6580 | 1328.7830 | 1322.4420 | 1334.1640 | 1330.1040 | 1348.2480 | 1527.9990 | 1388.9500 |
| 50 | 1176.9920 | 1190.9300 | 1172.6750 | 1145.7240 | 1170.4780 | 1176.8970 | 1171.9950 | 1371.9950 | 1201.9010 |

Table 6. Average Differences of the Results

| Scheduling Policy | Neural Network Model | Average Difference, % |
|---|---|---|
| **FCFS** | **10 Hidden Node, L-M training Alg.** | **2,271795674** |
| FCFS | 20 Hidden Node, L-M training Alg. | 2,50928458 |
| FCFS | 30 Hidden Node, L-M training Alg. | 11,31792847 |
| FCFS | 10 Hidden Node, C-G training Alg. | 4,138162165 |
| FCFS | 20 Hidden Node, C-G training Alg. | 5,204217314 |
| FCFS | 30 Hidden Node, C-G training Alg. | 8,849168829 |
| FCFS | 10 Hidden Node, G-D training Alg. | 2,437718382 |
| FCFS | 20 Hidden Node, G-D training Alg. | 7,285583699 |
| FCFS | 30 Hidden Node, G-D training Alg. | 7,410640241 |
| LCFS | 10 Hidden Node, L-M training Alg. | 3,185375557 |
| LCFS | 20 Hidden Node, L-M training Alg. | 3,211717432 |
| LCFS | 30 Hidden Node, L-M training Alg. | 15,09407928 |
| LCFS | 10 Hidden Node, C-G training Alg. | 3,07761742 |
| LCFS | 20 Hidden Node, C-G training Alg. | 4,630245757 |
| LCFS | 30 Hidden Node, C-G training Alg. | 5,167792174 |
| **LCFS** | **10 Hidden Node, G-D training Alg.** | **2,982247861** |
| LCFS | 20 Hidden Node, G-D training Alg. | 4,114307727 |
| LCFS | 30 Hidden Node, G-D training Alg. | 4,360410091 |
| **SJF** | **10 Hidden Node, L-M training Alg.** | **2,028381072** |
| SJF | 20 Hidden Node, L-M training Alg. | 3,421056389 |
| SJF | 30 Hidden Node, L-M training Alg. | 3,503745147 |
| SJF | 10 Hidden Node, C-G training Alg. | 2,491615607 |
| SJF | 20 Hidden Node, C-G training Alg. | 3,300253264 |
| SJF | 30 Hidden Node, C-G training Alg. | 3,424767992 |
| SJF | 10 Hidden Node, G-D training Alg. | 9,290039653 |
| SJF | 20 Hidden Node, G-D training Alg. | 10,04691753 |
| SJF | 30 Hidden Node, G-D training Alg. | 10,00751593 |