Abstract: Computers in the current times are indispensable to mankind. They perform many tasks in much less time, when compared to humans, opening up new paths of exploration. One important aspect of computers is multi-tasking, which enables a system to execute multiple tasks simultaneously. The scheduler causes the processor to perform context switching among the various processes. It primarily divides the CPU time to various processes, based on a specific policy related to response time, throughput, and user interactivity. Some of the criteria which are considered while selecting a good scheduling algorithm include CPU utilization time, Load Average, Response time, Turnaround time, and throughput. Ideally, one should view this as an optimization problem. Throughput and CPU utilization should be maximized; where as other factors should be minimized. Occasionally, it requires to put the variance of any aspect, to a minimum value, rather than minimizing the aspect itself. Consistency of the aspect is preferred over inconsistency.

Keywords: Dynamic Time quantum, waiting time, turnaround time, Neural Network, Process Scheduling, MLFQ scheduling.

I. INTRODUCTION

1.1 Scheduling

The process scheduler is an imperative factor of operating systems and it is liable for the drill of CPU time slice to processes, whereas maintaining scheduling policy to obtain user interactivity, throughput, real time responsiveness, and more. The most accepted scheduling algorithms that adhere to user interactivity are priority based [1][17][18]. The fundamental plan is to maintain the CPU busy as much as possible by executing a process until it must wait for an event, and then switch to another process. Processes exchange among overriding CPU cycles (CPU-burst) and performing I/O (I/O-burst) [2][19][20]. Scheduling is a difficult choice making action since of inconsistent goals, finite capital and the complexity in precisely modelling actual world scenarios [13][21][22].

1.2 Scheduling Criteria

There are a number of different criteria to think when trying to select the "best" scheduling algorithm for a exacting conditions and situation, together with, CPU utilization, Throughput, Turnaround time, Waiting time, Load average, Response time. In universal one needs to optimize the average importance of a criteria (Maximize CPU utilization and throughput, and minimize all the others). Conversely from time to time one needs to do somewhat dissimilar, such as to minimize the maximum response time [2][23][6][7]. Every so often it is most attractive to decrease the variation of criteria than the real charge [8][9][10][11].

1.3 Choice of scheduling algorithm

When scheduling a commodity operating system, code author has to predict the best algorithm to enforce the desired task that should be accomplished. The best algorithm has to finish the desired process with intended throughput or execution time with the combination of several mechanisms. For example windows Vista/XP/NT uses a mixture of predetermine priority pre-emptive scheduling, a multilevel feedback queue, first come first serve, and round robin.

1.4 MLFQ Scheduling

Multilevel feedback queue (MLFQ) algorithms are extensions to the more general MLQ algorithms. Silberchatz, Galvin and Gagne [2][3][4], algorithm dividing the ready queue into numerous break up queues. The processes are appointed to one queue. Based on appropriate of the process, such as memory size, process priority or process type. Each queue has its individual scheduling algorithm. The major distinction relies on the fact that, in MLFQ algorithms, jobs are allowed to move from one queue to another, according to their runtime performance [15][24]. By stirring jobs from highest priority queues to lowest priority ones, the algorithm guarantees that short jobs and I/O-intensive processes get to the CPU much earlier. After a quantum of slice jobs are decreased in importance and moved to a lower priority queue [5].
II. ARTIFICIAL INTELLIGENCE AND SCHEDULING

Artificial Intelligence (AI) aims at constructing artifacts (machines, programs) that have the ability to study, adjust and show human-like aptitude. Hence, education algorithms are imperative for realistic applications of AI. The field of appliance knowledge is the study of methods for programming computers to learn. Research shows that Artificial neural networks (RNs) can be used as a appliance knowledge tool to learn the programming process [12] [13]. An ANN is a data modelling tool that is able to imprison and represent composite and non-linear input / output associations. Neural networks have numerous applications in scheduling. These are: searching networks (Hopfield net), probabilistic networks (Boltzmann machine), error-correcting networks (multilayer perceptron), competing networks and self-organizing networks. Individual of the tradeoffs of using neural networks is that they can be trained for precise use cases by experts and sophisticated users and then reused by less theoretically inclined users without requiring any knowledge about neural networks or programmers. The formation of a network is done totally offline. This income that it is not necessary to have a kernel customized to train the networks. In fact, networks can be trained (as well as tested) on any other operating system if this is essential. The feature data is extracted from a convention kernel that works and is written to a file that can be used for training and testing.

2.1 Neural Network in MLFQ

Attractive feature of Neural Network is the capability to learn. This provides a level of flexibility that does not exist in current scheduling disciplines. Detecting a bad state that leads to bad state and procedure to avoid it is an example of the helpfulness of this learning capability [14]. Neural networks are the principle AI system to use here because they can be deployed in dissimilar situations when preparation them. Input and output are continuous numbers, which are also usually process data accessible. It also allows a variety of search for “brute force” by testing. Individual can essentially add character still if it is unclear how these characters can be used to compute process priorities[16]. The network can “figure out” how to do this, if possible. In addition, in case of noise at the entrance, neural networks disgrace elegantly [3][14][15].

III. THE PROPOSED WORK

MLFQ uses a number of queues with numerous time slices. Static time slice to the queue is control of MLFQ algorithm. In this paper, we have projected a novel variation of MLFQ algorithm. We suggest algorithm in which the cut off time is assigned to each queue for the MLFQ programming in such a way that changes with each round to perform dynamically and the neural network is used to regulate this time division again to optimize the response time. The on the total performance of algorithm is to be observed to be enhanced by MLFQ using dynamic time slice and neural network over MLFQ using static time slice for each queue. a lot of context switches possible if time slice is very small and the algorithm becomes FCFS if it extremely big. So we try to solve this difficulty by building option of lively time slice suitably, where the time slice are accustomed in all round according to active priority.

3.1 DTQ allocation with Neural approach

Turnaround instance (tat) is the time hole between the direct of submission of process and the immediate of its conclusion. Neural network is used to optimize turnaround time. Number of the queues and quantum of each queue concern the turnaround time openly. In this paper, we suggest the algorithm for solving these inconveniences and optimizing the turnaround time. In this algorithm Neural system has been utilized to discover the minimized quantum of each queue. Following method can be functional using neural network:

1. Execute the programs with diverse particular time slices which gives lowest turnaround-time.
2. construct the facts base of standing and lively type of the programs from the execution traces obtained in step 1 and train them with the neural algorithm
3. If a fresh program comes, categorize it and run the program with this predicted Time slice.
4. If the novel program occurrence is not in the knowledgebase, go to step 1.

Fig 1: Architecture for Intelligent DTQ allocation.
Dynamic quantum is considered by calculating $B_{sp}$ i.e. burst span, calculated by average of initial burst value of, mid of the queue burst value and last burst value of queue. Also average of priorities and highest priority of queue is measured to estimate dynamic time slice. Once the dynamic quantum is assigned to the queue, it is fed to the neural system used to adjust this quantum again to achieve a minimized response time. As shown in Figure 1, the neural network updates the weights and then changes the quantum of the input queues and specifies a novel quantum for the queues. We can discover the property of this transformation on the standard answer time, the novel quantities of how much will be specified to the NN function.

The quantum of queues is fed back to the inputs in a recursive approach, earnings just the original quantum of an individual queue is fed to the input and the other queues obtain the previous amounts as inputs. After replacing the original quantum of a particular queue in NN purpose, using pre-assumed default processes used to attain the main turnaround time, the novel turnaround time caused by this transform is establish. Here, when a alteration is practical in the quantum of a particular queue, the numeral of queues can be malformed. It is probable that dropping the quantum caused supplementary processes are moved to the lower queues or a new queue is further to the number of required queues. By characterizing or recognizing programs it may be likely to make out their previous implementation history and expect their resource needs. Here Figure 2 shows how to reduce the TaT of programs by using NN procedure. We determine certain stationary and lively description of a program like input type, input size, text, process size and uninitialized data info size, taken as features which NN is used to optimize turnaround time. We call dynamic Time segment as the CPU burst time that minimizes turnaround time.

![Design of the Network](image)

Fig 2: Design of the Network.

### 3.2 Parameter Selection

Turnaround time decline rate gradually increases with the key dimension of the program. We believe here, processes of type computation bound and I/O bound. So here we need to revision the execution performance of numerous programs with its description that can be used to see coming the dynamic time slice. We can receive agent programs similar to matrix multiplication, categorization arrays, recursive, random numeral generator programs etc. And we exercise labelled data about these processes for training.

### 3.3 Proposed Algorithm

no = whole number of processes  
Priority = numeral of priorities  
$Pr_{high_{st}}$ = highest precedence in queue  
$AVG_{priority}$ = Average of priorities  
$B_{sp}$ = Burst span  
Input: Number of processes ($P_1, P_2, \ldots, P_n$)  
Burst time of processes ($B_{s1}, B_{s2}, \ldots, B_{sn}$)  
Priority of processes ($P_{ri1}, P_{ri2}, \ldots, P_{rim}$).  
$TQ$ = Time Quantum  
$B_i$ = initial burst value of queue,  
$B_m$ = mid burst value of queue  
$B_l$ = last burst value of queue  
Output: $TaT_{Av}$ = Average turnaround time

1. Process arrives in organized queue with coming time and predicted burst time.  
2. Sort the processes into queues according to uphill direct of burst time
3. Assign the priority to process in ascending order.
4. Modernize the about i/p type, i/p size, text, program size and uninitialized data info size.  5. For all queue x=1 to n repeat the following:
   5. 1 compute time quantum for each queue Qx as follows. TQ (Qx)= (Bsp * n) / (AVGpriority * Prighest)  Where Bsp= ( (Bi-initial+Bm-mid+Bl-last)/3)  compute Average Turnaround time TaTAv

   For each job in the that particular queue
   Step a. Do steps b, c and d WHILE queue REQUEST becomes free.
   Step b. choose the initial process from the ready queue and assign the CPU to it for a time gap of up to 1 point quantum.
   Step c. If the remaining CPU burst time of the currently running process is less than equal half of the time quantum then allocate CPU again to the currently running process for remaining CPU burst time. After completion of execution, removed it from the ready queue and go to step a.
   Step d. Remove the currently running process from the ready queue REQUEST and put it at the tail of the ready queue.

5. 2 Using RNN locate the best possible importance of queue according to other queue quantum and the average turnaround time that is found in the earlier stages.

3.4 Implementation and Experimental Setup

The investigational method is separated into four stages. In the initial stage, we make the data set from the program's run traces and create the information support with the standing and vibrant features of the programs. In the second phase, we design and replicate MLFQ scheduler to study actions of MLFQ scheduling in terms of average turnaround time by using dynamic time slice, in the third phase we use NN to discover more appropriate time slice to get least amount average turnaround time by as long as it additional information about the process to choose upon time quantum and in the fourth phase we execute scheduler by neural approach as time quantum will be given to the scheduler suggested by RNN.

The experimentation is carried with the ready queue with 10 processors $P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9$ and $P_{10}$ with burst time’s 23, 75, 93, 48, 2, 5, 12, 20, 26 and 34 in two dissimilar cases such as Zero Arrival Time and Non Zero Arrival time. The following tables 1.1 and 1.2 presents the information regarding process and their burst with zero arrival time and non-zero arrival time and their priorities.

<table>
<thead>
<tr>
<th>S.NO</th>
<th>Process Name</th>
<th>Burst Time</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>P1</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>P2</td>
<td>75</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>P3</td>
<td>93</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>P4</td>
<td>48</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>P5</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>P6</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>P7</td>
<td>12</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>P8</td>
<td>20</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>P9</td>
<td>26</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>P10</td>
<td>34</td>
<td>10</td>
</tr>
</tbody>
</table>

Table: 1.2. Process with Non-Zero Arrival Time

<table>
<thead>
<tr>
<th>S. NO</th>
<th>Process Name</th>
<th>Burst Time</th>
<th>Priority</th>
<th>Arrival Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>P1</td>
<td>23</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>P2</td>
<td>75</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>P3</td>
<td>93</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>P4</td>
<td>48</td>
<td>6</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>P5</td>
<td>2</td>
<td>4</td>
<td>21</td>
</tr>
<tr>
<td>6</td>
<td>P6</td>
<td>5</td>
<td>5</td>
<td>24</td>
</tr>
<tr>
<td>7</td>
<td>P7</td>
<td>12</td>
<td>7</td>
<td>26</td>
</tr>
<tr>
<td>8</td>
<td>P8</td>
<td>20</td>
<td>8</td>
<td>28</td>
</tr>
<tr>
<td>9</td>
<td>P9</td>
<td>26</td>
<td>9</td>
<td>30</td>
</tr>
<tr>
<td>10</td>
<td>P10</td>
<td>34</td>
<td>10</td>
<td>32</td>
</tr>
</tbody>
</table>
Prediction of Turnaround time (TAT):
Here i am given the burst time of the process and arrival time of processes as an input to LM NN shown in the following fig. 1.1

Fig. 1.1 Process Input Data

Here i am passing the turnaround time of each process as output to the LM NN Model shown in the following fig. 1.2

Fig. 1.2 Process Outputs
In the following fig. 1.3 shows the execution process

Fig. 1.3 Neural Network executions

The following fig. 1.4 Show the overall response of process

Fig. 1.4 Overall Performances

The following fig. 1.5 shows the sample input to the LM NN model
The following fig. 1.6 show the expected turnaround time for the sample input data.

IV. CONCLUSION

When scheduling a commodity operating system, code author has to predict the best algorithm to enforce the desired task that should be accomplished. The best algorithm has to finish the desired process with intended throughput or execution time with the combination of several mechanisms. For example, Windows Vista/XP/NT uses a mixture of predetermine priority pre-emptive scheduling, a multilevel feedback queue, first come first serve, and round robin. There is no universal “best” scheduling algorithm, and numerous operating systems use comprehensive or combinations of the arrangement algorithms.
Neural networks are the ideal AI practice to use here because they can be deployed in dissimilar situations by education them. The input and output are incessant numbers, which are also frequently the process data available. It also allows a variety of search for "brute force" by conducting tests. One can just include characteristics, although it is not yet clear how this kind can be used to calculate process priorities. The network can "figure out" how to do this, if possible. In addition, in case of sound at the entrance, neural networks are kindly despoiled. But the drawback of neural networks is that it is very complicated to get any significance from their organization, because their information is sub-symbolically encoded. When a system is qualified for a new attribute, it will not give much information about how the characteristic understandably relates to other characteristics. But, on the other hand, there has been some research with the purpose of extracting rules from neural networks. This could be useful for creating more minimized rules-based schedulers after experimenting with the neural network scheduler. When features are introduced by subsystems that are "equally important" for classification. If this is the case, neural networks will work enhanced than algorithms that look at their input principles one at a time in a hierarchical way.

V. FUTURE SCOPE

Our potential effort will comprise extending our practice to evaluate performance of scheduler for other kind of processes. So for that we need to study various parameters of the process affecting turnaround time or response time whichever is required to minimize according to application. This might be useful for creating extra optimized rule-based planners following carrying out tests with the Recurrent Neural Network programmer.

REFERENCES

[15] Paolo Di Francesco ,“Design and implementation of a MLFQ scheduler for the Bacula backup software” 2012 , Malarden University , Sweden.(thesis)
[16] Remzi H , Andrea C ,“Operating Systems”, 2011, Four Easy Pieces (V0.4)
[28] Paolo Di Francesco ,“Design and implementation of a MLFQ scheduler for the Bacula backup software” 2012 , Malarden University , Sweden.(thesis)
[29] Remzi H , Andrea C ,“Operating Systems”, 2011, Four Easy Pieces (V0.4)

