

[Egyptian Informatics Journal (2016) 17, 125](http://dx.doi.org/10.1016/j.eij.2015.09.003)–[137](http://dx.doi.org/10.1016/j.eij.2015.09.003)

Cairo University

Egyptian Informatics Journal

[www.elsevier.com/locate/eij](http://www.elsevier.com/locate/eij) [www.sciencedirect.com](http://www.sciencedirect.com/science/journal/11108665)

# FULL-LENGTH ARTICLE

Designing of vague logic based multilevel feedback queue scheduler



Supriya Raheja [a](#_bookmark0),[\*](#_bookmark3), Reena Dadhich [b](#_bookmark1), Smita Rajpal [c](#_bookmark2)

a *Department of Computer Science & Engineering, School of Engineering & Technology, NCU (Formerly ITM University),*

*Gurgaon, Haryana, India*

b *Department of Computer Science & Informatics, University of Kota, Kota, Rajasthan, India*

c *Alpha Global IT, Toronto, Canada*

Received 28 January 2015; revised 2 August 2015; accepted 5 September 2015

Available online 19 October 2015

Abstract Multilevel feedback queue scheduler suffers from major issues of scheduling such as star- vation for long tasks, fixed number of queues, and static length of time quantum in each queue. These factors directly affect the performance of the scheduler. At many times impreciseness exists in attributes of tasks which make the performance even worse. In this paper, our intent is to improve the performance by providing a solution to these issues. We design a multilevel feedback queue scheduler using a vague set which we call as VMLFQ scheduler. VMLFQ scheduler intelli- gently handles the impreciseness and defines the optimum number of queues as well as the optimal size of time quantum for each queue. It also resolves the problem of starvation. This paper simu- lates and analyzes the performance of VMLFQ scheduler with the other multilevel feedback queue techniques using MatLab.

© 2015 Production and hosting by Elsevier B.V. on behalf of Faculty of Computers and Information,

KEYWORDS

Vague based Multilevel Feedback Queue (VMLFQ) Scheduler;

Scheduler;

Multilevel Feedback Queue (MLFQ) Scheduling; Vague set theory;

Vague inference system

Cairo University. This is an open access article under the CC BY-NC-ND license ([http://creativecommons.](http://creativecommons/)

org/licenses/by-nc-nd/4.0/).

1. Introduction

A scheduler is the key module of any contemporary operating system that manages the concurrent execution of active tasks by sharing the CPU time among these tasks. To achieve these goals it runs a scheduling algorithm which selects the next task

\* Corresponding author.

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

**Production and hosting by Elsevier**

to run as well as divide the CPU time. In a productive system, scheduler should be fair and efficient [[1,2]](#_bookmark53). Efficiency and fair- ness can be considered in terms of different parameters such as average waiting time, average turnaround time, average response time, and starvation. These goals vary with the sys- tem being used. Keeping these goals, operating system’s designers prefer to use Multilevel Feedback Queue (MLFQ) scheduling algorithm for scheduler over other scheduling algorithms.

The fundamental problems with the MLFQ scheduling are threefold: first is how to assign the parameters to the scheduler, such as how to decide the optimum number of queues, how much length of time quantum for each queue and how the pri- ority is assigned to each task, so that starvation will not occur

<http://dx.doi.org/10.1016/j.eij.2015.09.003>

1110-8665 © 2015 Production and hosting by Elsevier B.V. on behalf of Faculty of Computers and Information, Cairo University. This is an open access article under the CC BY-NC-ND license ([http://creativecommons.org/licenses/by-nc-nd/4.0/).](http://creativecommons.org/licenses/by-nc-nd/4.0/))

Q0 1

1-fv (X)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  | |  | |  | |  | |
|  | |  | |  | |  | |
|  | | | |  | | | |
|  | | | |  | | | |

# Q1

Q2

Figure 1 Variation in the size of time quantum within different queues.

[[3]](#_bookmark42). Most of the MLFQ schedulers allow variable length of time quantum to the queues. Often higher priority queues are assigned short time quantum for interactive tasks and the lower priority queues are assigned long time quantum as they

1-fv (*x*) tv (*x*)

0

*x* X

Figure 3 Vague member functions.

contain background tasks. [Fig. 1](#_bookmark4) illustrates the variation in size of time quantum within each queue. Each black box rep- resents the one time quantum.

Second, it tries to optimize the average turnaround time. Third, MLFQ desires that the system should be more respon- sive, thus to minimize the response time [[4]](#_bookmark42). However, the algo- rithms such as Round Robin minimize the response time but unfortunately increase the turnaround time [[5]](#_bookmark42). Moreover, task’s attributes can be having imprecise data which further affect these issues and make performance even worse. Hence, the focus of an operating system designer is to build a sched- uler that achieves all the desired goals of scheduling and at the same time handles the impreciseness.

In this paper we introduce a vague logic based new multi- level feedback queue CPU scheduler and call it as VMLFQ scheduler. VMLFQ scheduler considers all the above men- tioned problems with MLFQ and provides solutions to all. VMLFQ scheduler dynamically calculates the length of time quantum for each queue which makes the scheduler flexible. Hence, it can take decisions at run time. With all these, it also improves the performance of a system in terms of average waiting time, average turnaround time, average normalized turnaround time and average response time.

This paper is organized as follows. Section [2](#_bookmark5) gives brief explanation of the task scheduling algorithm and the Mul- tilevel Feedback Queue Scheduling. This section also discusses the related work of MLFQ. Section [3](#_bookmark15) provides the reader with the background information on vague set theory. Section [4](#_bookmark17) describes VMLFQ scheduler in detail. Section [5](#_bookmark34) discusses the simulation with the help of sample task sets and results. Finally, Section [6](#_bookmark47) concludes the work.

1. Related work

Scheduling algorithm is the technique that a scheduler uses to decide the next task to run. The performance of operating sys- tem mainly depends on the scheduling algorithm used by scheduler. There have been a number of scheduling algorithms proposed in the literature such as First Come First Serve, Priority, Shortest Job First, Round Robin, Multilevel Queue, and Multilevel FeedBack Queue scheduling algorithm [[6–9]](#_bookmark42). However, out of all, multitasking systems prefer Multilevel Feedback Queue Scheduling algorithm [[10,11]](#_bookmark42). As our focus is on Multilevel Feedback Queue, all these algorithms are out of scope of this paper.

Highest Priority

T5

2ms

T3

2ms

T1

2ms

Q0

T7

4ms

T6

4ms

T4

4ms

T2

4ms

Q1

Lowest Priority

T9

T8

Q2

Figure 2 Pictorial representation of multilevel feedback queue example.

Ready Queue

Task

admits

VMLFQ

Scheduler VIS-MLFQ

Scheduling Algorithm

Qn

Q2

Q1

Figure 4 VMLFQ scheduler.



Vague Logic Module

Vague Inference Engine

Task’s Attributes

Data Base

+

Rule Base

Ready Queue

Grade Function Module

Vague False Function

Vague Truth Function

Figure 5 Vague inference system for VMLFQ scheduler.

MLFQ scheduling is one of the widely known scheduling approaches for interactive systems. In 1962, Corbato et al. had first accounted the MLFQ scheduler in a system called as Compatible Time Sharing System (CTSS). Just like Mul- tilevel Queue, the Multilevel Feedback Queue also contains mul- tiple queues and each queue has a different priority. The highest priority task from highest priority queue is scheduled first with CPU. Here, the priority is the key factor for task to run since MLFQ allows moving the task from one priority queue to another priority queue. Therefore, the fundamental principle of MLFQ scheduling depends on how to set priorities of each task. Instead of assigning a static priority to task, MLFQ changes the priority according to its observed behavior [[4]](#_bookmark42).

Suppose, if a task repeatedly requests for input device and releases the CPU iteratively, MLFQ scheduler will mark this task as high priority task; rather if a task requests the CPU for long duration, MLFQ will mark it as low priority task. Additionally, if a task waits for CPU for long duration in lower priority queue, it will move to the higher priority queue. In such a manner, MLFQ tries to memorize about a task, and hence uses the past activities of the task to predict its behavior. Let us consider an example of MLFQ where ready queue is divided into three queues Q0, Q1 and Q2 as shown in [Fig 2](#_bookmark7). Here Q0 has higher priority than Q1, and Q2 has lowest prior- ity. Each queue has its own scheduling algorithm. The sched- uler follows RR scheduling approach for queues Q0 and Q1

|  |  |  |
| --- | --- | --- |
| Table 1 Sample task set 1. | | |
| Task name | Arrival time (ms) | Burst time (ms) |
| T1 | 0 | 40 |
| T2 | 0 | 30 |
| T3 | 0 | 50 |
| T4 | 2 | 70 |
| T5 | 4 | 25 |
| T6 | 6 | 60 |
| T7 | 7 | 45 |
|  |  |  |

Figure 6 VMLFQ scheduling example.



1

2 3

1 3

3

Q1

Q2

Q3

CPU Time

whereas for Q2, it follows FCFS approach. When the task enters in the system, firstly it is added at the end of Q0 and then system allots a fixed single time quantum. This scheduling algorithm provides the facility to move the tasks from one queue to another queue. If the task consumes more CPU time, the task is moved to the lower priority queue Q1 and allotted double time quantum.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| T1 | T2 | T3 | T4 | T5 | T6 | T7 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Q1 |  | | | | | | | |
|  | 0 | 10 | 20 | 30 | 40 | 50 | 60 | 70 |
| Q2 |  |  |  |  |  |  |  |  |
|  | 70 | 90 | 110 | 130 | 150 | 165 | 185 | 205 |
| Q3 |  |  |  |  |  |  |  |  |
|  | 205 | 215 | 235 | 275 | 305 | 320 |  |  |

Figure 7 Gantt chart for sample task set 1 using MLFQ.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| T1 | T2 | T3 | T4 | T5 | T6 | T7 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T1 | T3 | T4 | T6 | T7 |

Q1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| T1 | T2 | T3 | T4 | T5 | T6 | T7 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 |  | 10 | 20 | 30 | 40 | 50 | 60 | 70 |
|  |  |  |  |  |  |  |  |  |
| 70 |  | 90 | 110 | 130 | 150 | 165 | 185 | 205 |
|  |  |  |  |  |  |  |  |  |
| 205 |  | 215 | 235 | 265 | 295 | 310 |  |  |
|  | T4 |  |  |  |  |  |  |  |

Q2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| T1 | T2 | T3 | T4 | T5 | T6 | T7 |

Q3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T1 | T3 | T4 | T6 | T7 |

Q4

310 320

Figure 8 Gantt chart for sample task set 1 using PMLFQ.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| T1 | T2 | T3 | T4 | T5 | T6 | T7 |

0

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Q1 |  | | | | | | | | |
|  | 0 | 10 | | 20 | 30 | 40 | 50 60 | | 7 |
| Q2 |  |  | |  |  |  |  | |  |
|  | 70 | 85 | | 101.5 | 118 | 134.5 | 151 167.5 18 | | |
| Q3 |  |  | |  |  |  |  | |  |
| 184 187.5 201 | | | | | 219.5 | 241.5 | 263.5 | 285.5 | |
| Q4 | | |  | |  |  |  |  | |
| 285.5 287 298.5 | | | | | 320 |  |  |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| T5 | T2 | T1 | T7 | T3 | T6 | T4 |

4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| T2 | T1 | T7 | T3 | T6 | T4 |

|  |  |  |
| --- | --- | --- |
| T3 | T6 | T4 |

Figure 9 Gantt chart for sample task set 1 using VMLFQ.

Q0 is using time quantum 2 ms and Q1 is using 4 ms as illus- trated in [Fig 2](#_bookmark7). In this example, the interactive and the I/O bound tasks (T1, T3 and T5) will complete their execution in Q0 but the CPU bound tasks (T2, T4, T6, T7, T8 and T9) will move to the lower level queues Q1 and Q2. The task that waits for a longer time can be moved to the higher priority queues.

Since MLFQ is mainly preferable algorithm for interactive tasks, Torrey and Coleman have implemented a MLFQ sched- uler to compare the response of interactive tasks. Proposed algorithm has shown some improvements in terms of turn- around time, as the MLFQ scheduling performance depends on the number of queues and the length of time quantum is assigned to each queue. With this aim, Parvar and Safari have utilized the recurrent neural network to optimize the number of queues and the size of time quantum of each queue of MLFQ scheduler [[12]](#_bookmark43). Hoganson has pointed the performance of MLFQ scheduler in terms of task starvation. MLFQ scheduling algorithm is efficient and effective for small CPU bound tasks or interactive tasks but in the lower priority queues, the CPU-intensive tasks may get starved and reduce the performance of MLFQ scheduler. He has presented an

This research work is grounded to handle the uncertainty using vague set theory. We call this as VMLFQ scheduler. We compare VMLFQ with MLFQ and PMLFQ scheduling. We claim that the proposed work results in better performance in terms of average waiting time, average response time, aver- age normalized turnaround time and average normalized turn- around time. In the next section we will discuss preliminaries of vague set theory which is the core part of our work.

1. Vague set theory

In this section we concisely discuss one of the generalized forms of fuzzy set theory called as vague set theory. Fuzzy set theory was specifically designed by Prof. Zadeh to mathe- matically represent impreciseness and uncertainty [[16–18]](#_bookmark48). It is a formalized tool for dealing with imprecision. Let *X* =

{*x*1, *x*2, .. ., *xn*} be the universe of discourse, where an element of *X* is denoted by *x*.

Definition 1 (*Fuzzy Set*). A fuzzy set *F* in *X* is defined by its membership function l : *X* ? [0, 1] where l (*x*) is the degree

approach that extenuates the MLFQ starvation problem [[13]](#_bookmark44). *F F*

Rao and Shet have further implemented the MLFQ sched- uler for multi-task real time systems [[14]](#_bookmark45). They have proposed a new multilevel feedback queue (NMLFQ) scheduling algo- rithm which is implemented in C++. Bhunia has also given a solution for the MLFQ scheduler for the tasks which get starved in the lower priority queues for waiting CPU. The pro- posed solution considers five queues. In this approach, the number of queues is fixed and also the time slice increases from upper to lower queues. The tasks from the lower queue may also be shifted to higher priority queues on the basis of remain- ing CPU burst time which results in the reduction of number of starved tasks up to some level [[15]](#_bookmark46). We are calling this algo- rithm as PMLFQ scheduling algorithm. In this book Arpaci- Dusseau and Arpaci-Dusseau have discussed numerous issues related to MLFQ scheduling [[4]](#_bookmark42). There is no development in the literature which shows the handling of uncertainty in MLFQ scheduling.

of membership of element *x* in a fuzzy set *F* [[19–22]](#_bookmark49).

In a fuzzy set, each element is assigned a single membership value in the interval [0,1]. This single membership value does not separate the evidence in favor and evidence against. Gau and Buehrer have pointed this single membership value and introduced vague set theory over fuzzy set theory which considers both evidences individually [[23]](#_bookmark50).

Definition 2 (*Vague Set*). A vague set *V* in *X* is defined by a truth membership function *tv*(*x*) ∈ [0, 1]and a false member- ship function *fv*(*x*) ∈ [0, 1], where *tv*(*x*) is a lower bound on the grade of membership of *x* derived from the evidence for *x*, *fv*(*x*) is a lower bound on the grade of membership of *x* derived from the evidence against *x*, and *tv*(*x*) + *fv*(*x*) 6 1. The grade of membership of *x* in the vague set is bounded to a subinterval [*tv*(*x*), 1 — *fv*(*x*)] of [0, 1] as shown in [Fig. 3](#_bookmark6) [[23,24]](#_bookmark50).

300 350

MLFQ PMLFQ VMLFQ

MLFQ PMLFQ VMLFQ

250

300

200

Waiting Time

150

100

250

200

Turnaround Time

150

100

50

1 2 3 4 5 6 7

Number of Tasks

50

1 2 3 4 5 6 7

Number of Tasks

## (a) (b)

60

MLFQ PMLFQ VMLFQ

50

40

Response Time

7

6.5

MLFQ PMLFQ VMLFQ

Normalized Turnaround Time

6

5.5

30 5

4.5

20

4

10

3.5

0

1 2 3 4 5 6 7

Number of Tasks

3

1 2 3 4 5 6 7

Number of Tasks

## (c) (d)

250

AWT ATT ANT ART

200

150

Performance

100

50

0

1 2 3

Sample Task Set 1

## (e)

Figure 10 (a) Waiting time (task set 1), (b) turnaround time (task set 1), (c) response time (task set 1), (d) normalized turnaround time (task set 1) and (e) overall performance result (task set 1).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Q1 |  | | | | | | | | |
|  | 0 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 |
| Q2 |  |  |  |  |  |  |  |  |  |
|  | 64 | 80 | 96 | 112 | 128 | 144 | 155 | 171 | 187 |
| Q3 |  |  |  |  |  |  |  |  |  |
|  | 187 | 253 | 259 | 263 | 296 | 345 | 363 | 406 |  |
| Figure 11 Gantt chart for MLFQ (sample task set 2). | | | | | | | | | |

1. VMLFQ scheduler

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| T1 | T2 | T3 | T4 | T5 | T7 | T8 |

|  |  |  |
| --- | --- | --- |
| Table 2 Sample task 2. | | |
| Task name | Arrival time (ms) | Burst time (ms) |
| T1 | 0 | 90 |
| T2 | 0 | 30 |
| T3 | 0 | 28 |
| T4 | 0 | 57 |
| T5 | 0 | 73 |
| T6 | 0 | 19 |
| T7 | 0 | 42 |
| T8 | 0 | 67 |
|  |  |  |

VMLFQ scheduler supports a finite number of queues *n* and a finite number of active tasks *N*. Each task requires arrival time *A* and burst time *B*. Let *Q* = {*Q*1, *Q*2, .. ., *Qn*} be the set of *n*

queues and *T* = {*T* , *T* , .. ., *T* } be the set of *N* tasks, then

1 2 *N*

Definition 3 (*Vague Value*). The interval [*tv*(*x*), 1 — *fv*(*x*)] is called the ‘vague value’ of *x* in *V* as given in [Fig 3](#_bookmark6). The vague set *V* is written as *V* 6 *x*, [*tv*(*x*), *fv*(*x*)] > : *x* ∈ *X*.

Consider a universe *X* for priority of the tasks. A vague value for priority of a task can be [0.5, 0.7]. Here, 0.5 and .3 represent the truth part and false part of vague value respec- tively, whereas remaining 0.2 stands for hesitated part of vague value [[25]](#_bookmark51).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 |

{*Ai*|*i* = 1, .. ., *N*} represents the arrival time and {*Bi*| *i* = 1, .. ., *N*} represents the burst time for *i*th task respectively. VMLFQ scheduler dynamically divides the ready queue into finite number of queues *Q*1, .. ., *Qn* ∈ *Q*. VMLFQ scheduler has two components as shown in [Fig. 4](#_bookmark8):

* Vague Inference System (VIS-MLFQ)
* Scheduling algorithm
  1. *VIS-VMLFQ*

Our VIS-MLFQ has the ability to learn the current behavior of the active tasks and based on the ability, it converts the inputs

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Q1 |  | | | | | | | | |
|  | 0 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 |
| Q2 |  |  |  |  |  |  |  |  |  |
|  | 64 | 80 | 96 | 112 | 128 | 144 | 155 | 171 | 187 |
| Q3 |  |  |  |  |  |  |  |  |  |
|  | 187 | 211 | 217 | 221 | 245 | 267 | 287 | 311 |  |
| Q4 |  |  |  |  |  |  |  |  |  |
|  | 311 | 353 | 362 | 387 | 406 |  |  |  |  |

Figure 12 Gantt chart for PMLFQ (sample task set 2).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| T1 | T2 | T3 | T4 | T5 | T7 | T8 |

|  |  |  |  |
| --- | --- | --- | --- |
| T1 | T4 | T5 | T8 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Q1 | 0 | 7.9 | 15.8 | 23.7 | 31.6 | 39.5 | 47.4 | 55.3 | 63.2 |
| Q2 |  |  |  |  |  |  |  |  |  |
|  | 63.2 | 74.3 | 90 | 105.7 | 121.4 | 137.1 | 152.8 | 168.5 | 184.2 |
| Q3 |  |  |  |  |  |  |  |  |  |
|  | 184.2 | 188.6 | 195 | 213.4 | 235.9 | 258.4 | 280.9 | 303.4 |  |
| Q4 |  |  |  |  |  |  |  |  |  |

303.4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| T3 | T2 | T7 | T4 | T8 | T5 | T1 |

|  |  |  |  |
| --- | --- | --- | --- |
| T4 | T8 | T5 | T1 |

Q5

T1

T5

387.6

314.3 335.2 361.4 387.6

388.3 406

Figure 13 Gantt chart for VMLFQ (sample task set 2).

into the desired output. Here, our aim is to generate an opti- mum value for time quantum. Hence, our VIS takes the burst time and number of tasks as inputs and maps these inputs into an optimum size of time quantum. [Fig. 5](#_bookmark9) illustrates the flow of data from one module to another module in VIS-VMLFQ. It has 4 modules as given below:

* Vague Logic Module
* Grade Function Module
* Vague Inference Engine
* Data Base and Rule Base
  + 1. *Vague Logic Module*

VLM converts the inputs into the respective vague values to handle the dependability of task’s parameters in the universe of discourse [0, 1]. These vague values are defined with true- membership function (*tQ*) and false-membership function (*fQ*)*.* VLM takes two parameters, burst time (*B*) and number of active tasks (*N*) as inputs from the Data Base. Based on these inputs *tQ* and *fQ* are defined as given in Eq. [(1)](#_bookmark23) and Eq. [(2)](#_bookmark24) respectively.

work, VLM handles the uncertainty and impreciseness by con- sidering the in-between dependability of different attributes of tasks during its decision making. It results in the formation of these two membership functions *tQ* and *fQ*, on which decisions of our scheduler depend.

* + 1. *Grade Function Module*

GFM defines the degree of accuracy of vague values. It receives the vague functions as input and by adding these two functions it returns the accuracy among vague value as given in Eq. [(3)](#_bookmark22). Similar to vague functions, the degree of accu- racy *SQ* should also be less than 1.

*SQ* = *tQ* + *fQ*; *SQ* ≤ 1 (3)

* + 1. *Data Base and Rule Base*

Data Base acts as the container for the ready tasks [[26–28]](#_bookmark52). It retrieves all the necessary information about tasks from the ready queue and stores within it, for example burst time *B*, arrival time *A*, number of ready tasks *N*, static time quantum *QS*. It also maintains the information about the maximum and minimum burst time, average burst time currently present in

*Bavg*

*t* =

*Q*

(1)

ready queue using Eqs. [(4)–(6)](#_bookmark25).

*Bavg*

+ *Bmax* + *N*

*Bmax* = max {*B*1; *B*2 .. .... *BN*} (4)

*Bavg*

*f* =

(2)

*B* = min {*B* ; *B* *B*

} (5)

*Q Bavg* + *Bmin* + *N*

Though *tQ* and *fQ* functions are independent to each other a

*min*

1 2 *N*

X*N Bi*

relation is drawn between these two vague functions i.e. *tQ* + *fQ* 6 1 and *tQ* 6 *fQ* 6 1. Since the system is not aware of the

*Bavg* =

*i*=1

*N* (6)

actual values of parameters and scheduler does not consider the dependability of parameters to schedule the tasks. In our

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| T6 | T3 | T2 | T7 | T4 | T8 | T5 | T1 |

It always fetches the current values from the ready queue and returns to VLM and Vague Inference Engine.

350 450

MLFQ PMLFQ VMLFQ

MLFQ PMLFQ VMLFQ

300

250

Waiting Time

400

350

Turnaround Time

300

200 250

150

100

50

1 2 3 4 5 6 7 8

Number of Tasks

200

150

100

50

1 2 3 4 5 6 7 8

Number of Tasks

## (a) (b)

60 10

MLFQ PMLFQ VMLFQ

MLFQ PMLFQ VMLFQ

9

50

Normalized Turnaround Time

8

40

Response Time

7

30 6

20

10

0

1 2 3 4 5 6 7 8

Number of Tasks

5

4

3

2

1 2 3 4 5 6 7 8

Number of Tasks

## (c) (d)

300

AWT ATT ANT ART

250

200

150

Performance

100

50

0

1 2 3

Sample Task Set 2

## (e)

Figure 14 (a) Waiting time (task set 2), (b) turnaround time (task set 2), (c) response time (task set 2), (d) normalized turnaround time (task set 2) and (e) overall performance result (task set 2).

* + 1. *Vague Inference Engine (VIE)*

VIE returns the optimum value of time quantum *QD*. It fetches the value of *QS* from the Data Base and the degree of accuracy from the GFM. Finally, on the basis of rules given in Eq. [(7)](#_bookmark29), it returns the size of time quantum *QD*.

if (6*i*; *Ai* == 0) then

*QD* = *SQ* × *QS* (7)

else

*QD* = *QS*

* 1. *VMLFQ Algorithm*

300

250

**Average Waiting Time**

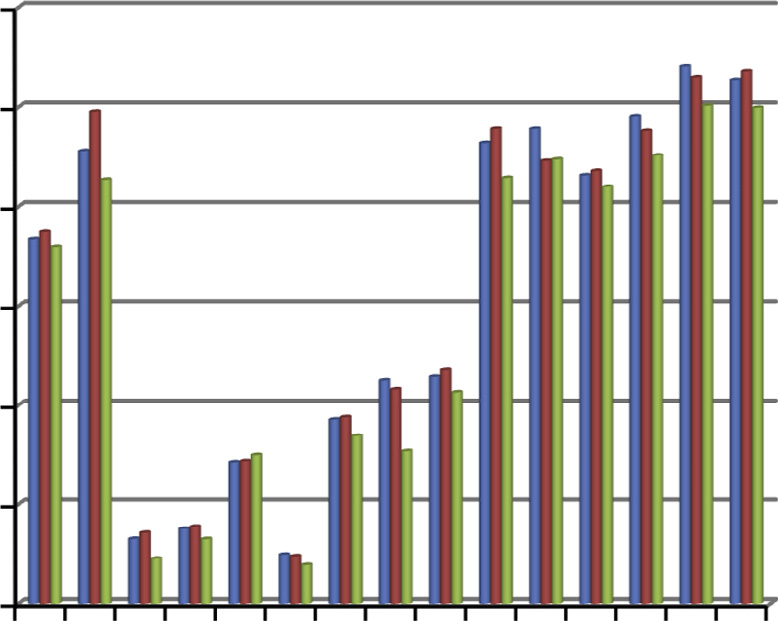
200

150

100

50

0



1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

### Samples

#### MLFQ PMLFQ VMLFQ

Figure 15 Average waiting time.

400

*Begin*

Initialize the variables

*QS* = static time quantum assigned by the system

*N* = number of tasks

*Do Loop:* 1, .. ., *N*

Initialize the variables

*B* = burst time of task. A = arrival time of task.

*//* initial value of remaining burst time.

*RBT = B*

*End Loop*

*Do Loop:* 1, .. ., *N*

Assign all tasks to Q1.

Calculate *QD* for Q1 // *Using VIS-MLFQ End Loop*

*Do Loop:* 1, .. ., *N*

Calculate the response ratio (RR) using Eq. [(8)](#_bookmark31).

*RR* =

*Waiting Time* + *Burst Time*

*Burst Time*

(8)

*End Loop*

Sort the Queue in descending order of RR. Schedule the tasks for QD time.

*if* (*RBT* < *QD*)

Task does not need more CPU time

*else*

Task moves to lower level queue *Qi* where 2 6 *i* 6 *n End if*

Calculate the dynamic quantum for lower level queues *Q*2 to

*Qn* using Eq. [(9)](#_bookmark33).

*QD* = *SQ*(*QD* + *N*)

(9)

Go to begin *until N* = = 0

*End*

350

**Average Turnaround Time**

300

250

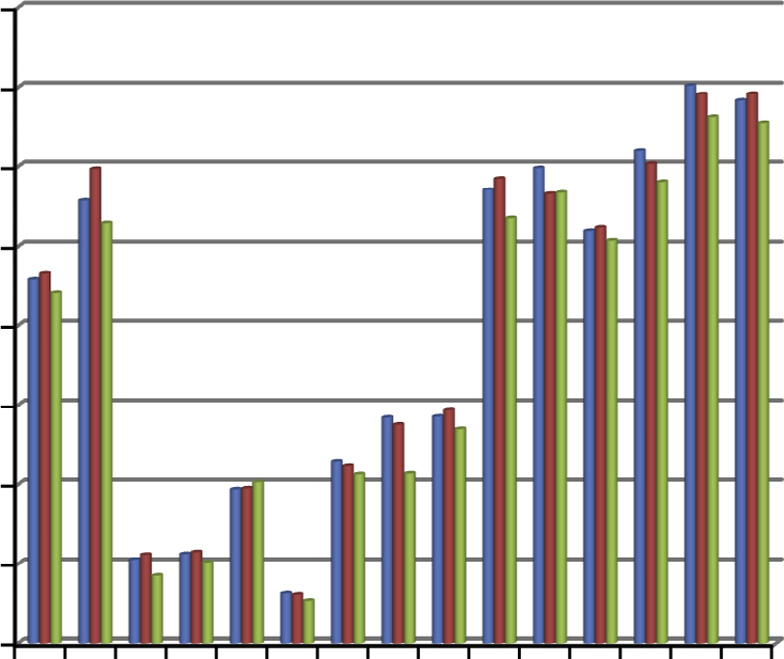
200

150

100

50

0



1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

### Samples

#### MLFQ PMLFQ VMLFQ

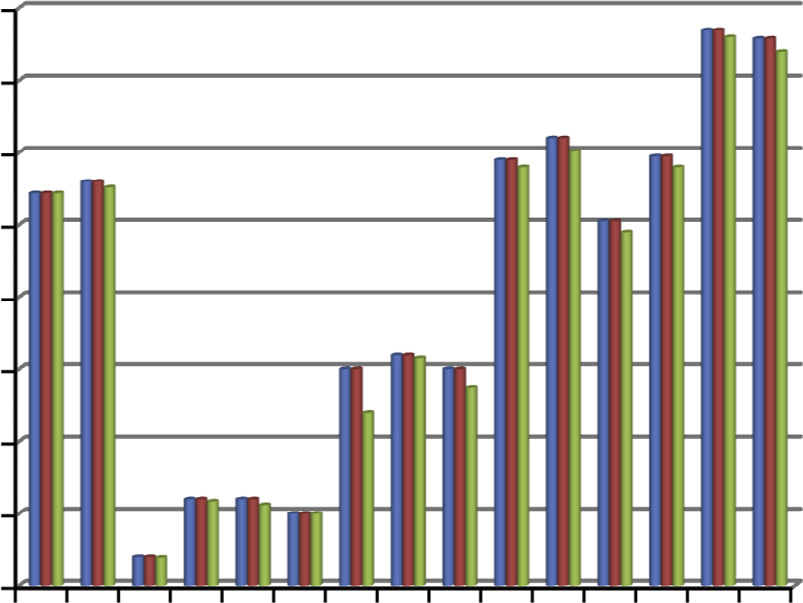
Figure 16 Average turnaround time.

* 1. *Working of VMLFQ scheduler*

Initially, it assigns all active tasks to highest priority queue Q1. VMLFQ scheduler sorts Q1 in the descending order of response ratio calculated using Eq. [(8)](#_bookmark31). It schedules the task with highest response ratio first with CPU for the length of

optimum time quantum which is calculated by the component VIS-MLFQ.

The length of optimum time quantum depends on the num- ber of tasks, their burst time, and also on the system assigned time quantum. After completing the assigned time quantum, the task will either move to next level queue Q2 or leave the system. If the task has completed its execution it will leave the system otherwise resumes its execution in the next level queue. The tasks of lower level queues will be scheduled only when the higher level queues becomes empty. As the task waits for a long time, the priority of the task will automatically move up on the basis of response ratio. Considering the response ratio for scheduling the tasks prevents the starvation problem in lower level queues [[29]](#_bookmark54). The value of time quantum for other queues *Q*2, .. ., *Qn* depends on the value of time quantum for previous queue, number of tasks in current queue and the remaining burst time of tasks.

40

35

**Average Response Time**

30

25

20

15

10

5

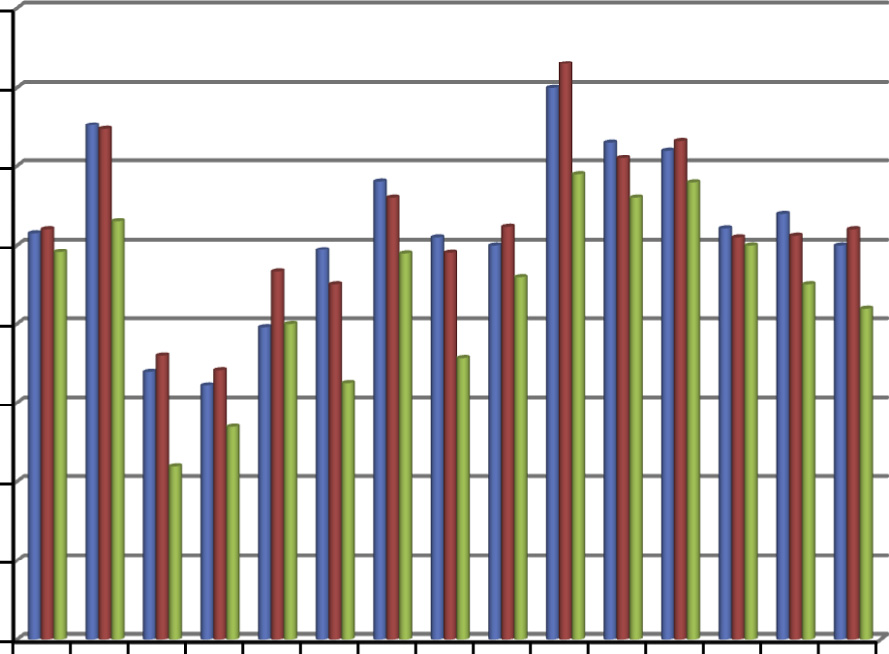
0

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

### Samples

#### MLFQ PMLFQ VMLFQ

Figure 17 Average response time.

8

7

6

5

4

3

2

1

Let’s look at an example, where task 2 is I/O, task 1 is CPU bound and task 3 is highly CPU bound. These three tasks are initially assigned to higher priority queue Q1 (see [Fig. 6](#_bookmark10)). As task 2 is I/O bound, it needed less CPU time and completed its execution within the assigned time quantum (calculated through VIS) but task 1 and task 3 being CPU bound need more CPU time than the assigned time quantum. Therefore, these tasks have shifted to lower priority queue Q2. A new time quantum is assigned to the tasks in Q2. Now we can see that in Q2, task 1 has completed its execution within the time quan- tum but task 3 needs more CPU time so it is again shifted to Q3. In this specific example our ready queue is divided into three queues but there can be more number of queues depend- ing on the requirement of application. Next section discusses the two components of VMLFQ scheduler in detail.

1. Simulation results

MatLab is used to design the VMLFQ scheduler. We are pre- senting some of the snippets for different modules defined for the VIS-MLFQ. Below is the snippet for VLM which fetches the data from the Data Base and returns the vague value.

function [tQ, fQ] = vague\_value(*b*, *n*) bavg=mean(*b*);

bmax=max(*b*); bmin=min(*b*);

tQ = (bavg)/(bavg + bmax + *n*); fQ = (bavg)/(bavg + bmin + *n*);

end

The following snippet is for GFM that computes the grade value after extracting the membership functions tQ and fQ from VLM.

0

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

function [SQ] = grade\_value(tQ, fQ) SQ = tQ + fQ;

end

 MLFQ  PMLFQ  VMLFQ

Figure 18 Average normalized turnaround time.

The given snippet tells us how VIE calculates an optimum value for time quantum.

ART ANT ATT

**Performance Improvement in %**

 MLFQ vs VMLFQ  PMLFQ vs VMLFQ

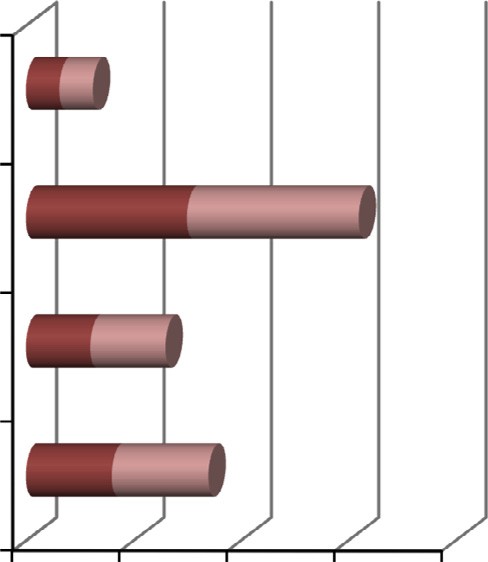
function[QD] = OTQ(SQ, Qs) if(sum= =0)

QD = SQ \* Qs;

else

QD = Qs;

end end



3.12 3.12

15

16

6 7

8 9

AWT

0 10 20 30 40

In this section, we are considering two sample task sets to eval- uate the work. These task sets show how the scheduling sequence of tasks using VMLFQ scheduler is different from

Figure 19 Improvement in performance of VMLFQ scheduler with MLFQ and PMLFQ scheduling.

other MLFQ scheduling techniques. Each sample task set is scheduled with MLFQ, PMLFQ and VMLFQ scheduling

algorithms. Finally, these algorithms are compared in terms of performance metrics discussed in next section.

* 1. *Performance criteria*

We discuss the performance metrics which we consider for evaluating the performance of VMLFQ scheduler.

Definition 4 (*Waiting Time*). Total time a task waits in ready queue from its submission to completion. Average waiting time can be calculated as given in Eq. [(10)](#_bookmark38). Here *W*1, *W*2, .. ., *WN* are the waiting times for tasks (*T*1, .. ., *TN*) ∈ *T* respec- tively and *N* is the total number of active tasks.

but it assumes fixed number of queues i.e. 5. Although, it has 5 queues but the size of time quantum in each queue increases in the same way as in MLFQ like 10 ms, 20 ms, 30 ms and so on. [Fig. 8](#_bookmark13) shows the sequence of scheduling in each queue. As all tasks have finished their execution before reaching Q5, we have therefore represented scheduling up to Q4 only.

Finally, we have applied the VMLFQ scheduler over the same task set *T*. VMLFQ scheduler intelligently divides the ready queue into four sub queues (*Q*1, .. ., *Q*4) ∈ *Q* at run time. Initially, all tasks are assigned to Q1 as discussed in Section [4](#_bookmark17). The optimum size of time quantum QD for each queue is cal- culated using VMLFQ scheduling algorithm and it returns the size for Q1, Q2, Q3 and Q4 as 10 ms, 16.5 ms, 22 ms and 30 ms respectively. The scheduling sequence for VMLFQ scheduler is

*AWT* = *W*1 + *W*2 + ... *WN*

*N*

(10)

shown in [Fig. 9](#_bookmark14). Based on the above Gantt charts, waiting time, turnaround time, response time and normalized turn- around time are computed using Eq. [(5)](#_bookmark26), Eq. [(6)](#_bookmark27), Eq. [(7)](#_bookmark29) and Eq. [(8)](#_bookmark31) respectively. The results are illustrated in [Fig. 10](#_bookmark16)(a),

Definition 5 (*Response Time*). Time elapsed between the task submission to its first useful output. Average response time can be calculated as given in Eq. [(11)](#_bookmark39).

*R*1 + *R*2 + .. . *RN*

(b), (c), (d) respectively.

[Fig. 10](#_bookmark16)(e) illustrates the overall performance of all three scheduling algorithms. Here, ‘‘1” represents MLFQ schedul- ing, ‘‘2” represents PMLFQ scheduling and ‘‘3” represents VMLFQ scheduling algorithm. From the above graph we

*ART* =

*N* (11)

can perceive the reduction in the performance metrics with the VMLFQ scheduling. Reductions itself show the improve-

Here *R*1, *R*2, .. ., *RN* are the response times for 6*Ti* ∈ *T,*

*i* = 1, .. ., *N* respectively.

Definition 6 (*Turnaround Time*). Total time of the task from the submission to its complete execution. Turnaround time can be computed by adding the waiting time and the burst time of the task. Average turnaround time can be calculated as given in Eq. [(12)](#_bookmark40). Here *TT*1, *TT*2, .. ., *TT*N are the turnaround times for tasks (*T*1, .. ., *TN*) ∈ *T* respectively.

*TT*1 + *TT*2 + .. . *TTN*

ment in the performance by VMLFQ.

*5.3. Sample task set 2*

Let us suppose another Sample task set 2 with eight tasks (*T*1, .. ., *T*8) ∈ *T* with different burst times but same arrival times as given in [Table 2](#_bookmark18). Now, consider the static time quan- tum *QS* as 8 ms. Again apply all three algorithms one by one over task set 2 to analyze the performances in more detail. Firstly, apply the MLFQ scheduling algorithm. The assigned

*ATT* =

*N* (12)

static time quantum QS for first queue is 8 ms. For second and third queues the value of time quantum used was 16 ms

Definition 7 (*Normalized Turnaround Time*). It indicates the relative delay experienced by a task. It is the ratio of turn- around time to the burst time. Average normalized turnaround time can be calculated as given in Eq. [(13)](#_bookmark41).

*NT*1 + *NT*2 + ... *NTN*

and 24 ms respectively. [Fig. 11](#_bookmark19) represents the scheduling sequence using Gantt chart for MLFQ scheduling.

Then, we have applied PMLFQ scheduling over the same task set. The time quantum used by PMLFQ scheduling for each queue is 8 ms, 16 ms, 24 ms and 32 ms respectively. The scheduling sequence for PMLQ is shown in [Fig. 12](#_bookmark20).

*ANT* =

*N* (13)

Finally, we have applied the proposed VMLFQ approach over the task set 2. In this case it divides the ready queue into

Here *NT*1, *NT*2, .. ., *NT*N are the normalized turnaround

times for tasks (*T*1, .. ., *TN*) ∈ *T* respectively.

* 1. *Sample task set 1*

Consider a sample task set *T* = {*T*1, *T*2, .. ., *T*7} with different arrival times and burst times as given in [Table 1](#_bookmark11). We have assumed static time quantum *QS* as 10 ms. Firstly, we have applied MLFQ scheduling algorithm over the task set *T*. MLFQ scheduling algorithm takes the size of time quantum for Q1 and Q2 as 10 ms and 20 ms respectively (discussed in Section [2](#_bookmark5)), whereas, Q3 does not need any time quantum as it follows FCFS algorithm. The sequence of scheduling for each queue is shown in [Fig. 7](#_bookmark12).

After that, we have applied the second scheduling algo- rithm i.e. PMLFQ. It behaves just like a MLFQ algorithm

five sub queues (*Q*1, .. ., *Q*5) ∈ *Q*. The VIS returns the value of QD as 7.9 ms i.e. used by scheduler as time quantum for Q1 whereas time quantum for Q2, Q3, Q4 and Q5 is 15.7 ms,

22.5 ms, 26.2 ms and 29 ms respectively. The scheduling sequence for VMLFQ is given in [Fig. 13](#_bookmark21).

Similarly for task set 2, waiting time, turnaround time, response time and normalized turnaround time are computed for each task and illustrated in [Fig. 14](#_bookmark28)(a), (b), (c) and (d) respectively.

Average waiting time *AWT*, average response time *ART*, average turnaround time *ATT* and average normalized turn- around time *ANT* are computed using Eqs. [(5), (6), (7) and (8)](#_bookmark26) respectively and are shown in [Fig. 14](#_bookmark28)(e). From the graph shown in [Fig. 14](#_bookmark28)(e), we can verify the improvement in the overall per- formance of VMLFQ as compared to MLFQ and PMLFQ. Randomly multiple sets of tasks were generated and simulated

using MatLab. All task sets were scheduled using these three algorithms. The performance comparison of these algorithms is illustrated in [Fig. 15](#_bookmark30), [Fig. 16](#_bookmark32), [Fig. 17](#_bookmark35) and [Fig. 18](#_bookmark36) for average waiting time, average response time, average turnaround time, and average normalized turnaround time respectively.

In the above graphs, red line represents the VMLFQ scheduling algorithm and in each case the outputs of VMLFQ algorithm is on lower side of the graph which shows the reduc- tion in the values of performance metrics. The reduction itself proves the improvement in the performance of scheduler as well as in the performance of system.

VMLFQ scheduler performs 3.12% better in average response time, 15% in average normalized turnaround time, 6% in average turnaround time and 8% in average waiting time as compared to MLFQ scheduling whereas it performs 3.12% better in average response time, 16% in average nor- malized turnaround time, 7% in average turnaround time and 9% in average waiting time as compared to PMLFQ scheduling as shown in [Fig. 19](#_bookmark37).

VMLFQ scheduler performs better mainly for four reasons: The proposed VMLFQ scheduler responds effectively to dynamic environment where number of queues are assigned at run time as well as time quantum to each queue is also pro- vided at run time. It handles the uncertainty and impreciseness of tasks. In addition, VMLFQ scheduler improves the starva- tion problem at the lower priority queues as we are considering the response ratio. Last but not the least, it improves the per- formance of the system in terms of average waiting time, aver- age response time, average turnaround time and average normalized turnaround time.

1. Conclusions

The purpose of our work is to present a novel approach to mul- tilevel feedback queue CPU scheduling. This paper discussed the problems and issues associated with the multilevel feedback queue (MLFQ) scheduling. To resolve the issues we presented a MLFQ scheduler that follows a vague logic approach to take the decisions while meeting the performance requirements. VMLFQ scheduling extends the concept of MLFQ scheduling using vague logic to consider the impreciseness associated with the scheduling parameters. The scheduler takes decisions based on the burst time and the number of tasks. The VMLFQ based scheduler provides the solution to each of the problems and issues. It has resolved the problem of starvation for lower level tasks. VMLFQ scheduler also followed the dynamic approach for assigning number of queues rather than fixed number of queues. Additionally, we compared the performance of VMLFQ scheduling algorithm with the MLFQ scheduling algo- rithm and PMLFQ scheduling algorithm via average response time, average waiting time, average turnaround time and aver- age normalized turnaround time. Based on the results shown in Section [5](#_bookmark34), we concluded that the VMLFQ scheduling algo- rithm has better performance over the traditional MLFQ scheduling and PMLFQ scheduling algorithms.

References

1. Tanenbaum AS, Woodhull AS. Operating systems design and implementation. 3rd ed.; 2006.
2. [Silberchatz A, Galvin PB, Gagne G. Operating systems concepts.](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0010) [8th ed. John Wiley and Sons; 2012](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0010).
3. [Torrey LA, Coleman J, Miller BP. A comparison of interactivity](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0015) [in the Linux 2.6 scheduler and an MLFQ scheduler. Softw Pract](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0015) [Experience 2007;37(4):347–64](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0015).
4. Arpaci-Dusseau RH, Arpaci-Dusseau AC. Operating systems: three easy pieces, scheduling: multilevel feedback queue. Version 0.80; 2014.
5. [Raheja S, Dhadich R, Rajpal S. An optimum time quantum using](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0025) [linguistic synthesis for round Robin CPU scheduling algorithm.](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0025) [Int J Soft Comput 2012;3(1):57–66](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0025).
6. [Nie B, Du J, Xu G, Liu H, Yu R, Wen Q. A new operating system](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0030) [scheduling algorithm. Advanced research on electronic commerce,](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0030) [web application, and communication, vol. 143. Springer; 2011. p.](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0030) [92–6](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0030).
7. [Dhamdhere DM. Operating systems a concept based approach.](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0035) [2nd ed. Tata McGraw-Hill; 2008](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0035).
8. [Milenkovic M. Operating system concepts and design. Interna-](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0040) [tional Edition. McGraw Hill; 1992](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0040).
9. [Haldar S, Aravind AA. Operating systems. Pearson Education](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0045) [India; 2009](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0045).
10. Abawajy JH. Job scheduling policy for high throughput comput- ing environments. In: Ninth IEEE international conferences on parallel and distributed systems, Ottawa, Ontario, Canada; 2002.
11. Feitelson DG, Rudolph L, Schwiegelshohn U. Parallel job scheduling, a status report. In: 10th international conference on job scheduling strategies for parallel processing, Springer-Verlag; 2004. p. 1–16.
12. [Parvar MRE, Parvar ME, Safari S. A starvation free IMLFQ](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0060) [scheduling algorithm based on neural network. Int J Comput](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0060) [Intell Res 2008;4(1):27–36](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0060).
13. [Hoganson KE. Reducing MLFQ scheduling starvation with](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0065) [feedback and exponential averaging. Consort Comput Sci Coll](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0065) [2009;25:196–202](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0065).
14. [Rao MV, Shet KC. Analysis of new multi-level feedback queue](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0070) [scheduler for real time kernel. Int J Comput Cognit 2010;8(3):5–16](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0070).
15. [Bhunia A. Enhancing the performance of feedback scheduling. Int](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0075) [J Comput Appl 2012;18(4):11–6](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0075).
16. [Klir GJ, St Clair UH, Yuan B. Fuzzy set theory: foundations and](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0080) [applications. Englewood Cliffs, NJ: Prentice Hall; 1997, ISBN: 0-](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0080) [13-341058-7](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0080).
17. [Zimmerman J. Fuzzy set theory and its applications. Norwell,](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0085) [Massachusetts, U.S.A.: Kluwer Academic Publishers; 2001](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0085).
18. [Atanassov K. Intuitionistic fuzzy sets. Fuzzy Sets Syst](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0090) [1998;20:87–96](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0090).
19. [Atanassov K. Intuitionistic fuzzy sets: theory and applications.](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0095) [New York: Physica-Verlag, Springer; 2000](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0095).
20. [Zadeh LA. Fuzzy sets. Inform Control 1965;8:338–56](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0100).
21. [Zadeh LA. The role of fuzzy logic in the management of](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0105) [uncertainty in expert systems. Fuzzy Sets Syst 1983;11:197–227](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0105).
22. [Zadeh LA. Outline of a new approach to the analysis of complex](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0110) [systems and decision processes. IEEE Trans Syst, Man Cybern](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0110) [1973;3:28–44](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0110).
23. [Gau WL, Buehrer DJ. Vague sets. IEEE Trans Syst, Man Cybern](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0115) [1993;23:610–4](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0115).
24. [Bustince H, Burillo P. Vague sets are intuitionistic fuzzy set.](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0120) [Fuzzy Sets Syst 1996;79:403–5](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0120).
25. [Lu A, Ng W. Vague sets or intuitionistic fuzzy sets for handling](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0125) [vague data: which one is better? Lecture notes in computer](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0125) [science, vol. 3716. Springer; 2005. p. 401–16, Conceptual Model-](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0125) [ing-ER2005](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0125).
26. [Raheja S, Dadhich R, Rajpal S. Many valued logics for modeling](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0130) [vagueness. Int J Comput Appl 2013;61(7):35–9](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0130).
27. [Raheja S, Dadhich R, Rajpal S. Designing of 2-stage CPU](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0135) [scheduler using vague logic. Adv Fuzzy Syst 2014:1–10](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0135).
28. [Raheja S, Dadhich R, Rajpal S. 2-Layered architecture of vague](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0140) [logic based multilevel queue scheduler. Appl Comput Intell Soft](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0140) [Comput 2014:1–12](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0140).
29. [Moallemi A, Asgharilarimi M. A fuzzy scheduling algorithm](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0145) [based on highest response ratio next algorithm. Computer](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0145) [sciences and software engineering. Springer; 2008. p. 75–80](http://refhub.elsevier.com/S1110-8665(15)00048-1/h0145).