Conference Paper Title*

*Note: Sub-titles are not captured in Xplore and should not be used

Khaled Amgad*, Mahmoud Mohamad†, Muhanad Atef‡ and Nader AbdAlGhani§
Computer Engineering Dept., Faculty of Engineering, Cairo University
Cairo, Egypt
*khaled.mohamed98@eng-st.cu.edu.eg, †mmmacmp@gmail.com, ‡muhanad.atef23@gmail.com, §nader_abdelghani@hotmail.com

***Abstract*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Scheduling is a crucial part of many real-time applications such as scheduling airlines and railways communications, product manufacturing processes, data packets routing in computer networks and pipe networks. Operating systems multi-processing environments are no different than those mediums mentioned, where processes compete over CPU utilization. This brought to existence the need for scheduling algorithms to justly assign processes to available CPUs in favour of optimizing performance measures, in particular, to maximize overall CPU utilization and throughput, and to minimize response time, waiting time and turnaround time. A schedular could be preemptive where it can temporarily interrupt a process without its cooperation and assign the CPU resources to a different process with the intention to assign them back to the former process, such operation is called context switching, or it could be non-preemptive (cooperative) by not being able to context switch between processes. Various preemptive and non-preemptive scheduling disciplines exist such as, first come, first serve (FCFS), shortest job first (SJF), shortest remaining time first (SRTF), round-robin (RR), multilevel queue and multilevel feedback queue. In FCFS, processes are non-preemptively executed according to their arrival time. In SJF, the CPU is assigned to the process with the smallest burst time. SRTF is a preemptive version of SJF where at each iteration, the process with the least remaining burst time takes control of the CPU. Round-robin preemptively assigns the CPU to each process in the ready queue for a static amount of time called quantum and executes them in an FCFS fashion. Multilevel queue algorithm partitions the ready queue into several queues to which processes are perpetually assigned and are executed according to another scheduling algorithm (e.g. RR). Processes cannot move from one queue to another. However, in a multilevel feedback queue algorithm, processes that don't terminate in one queue, due to having CPU burst time more than the time quantum assigned to their particular queue, are shifted to a lower priority queue. Due to the fact that long processes eventually sink to the lowest priority queue, they are carried out using FCFS to prevent starvation. Several papers discuss different methods to optimize the algorithms mentioned, each come with their set of advantages and drawbacks. One of many is [2], whose approach achieves better average response time but at the cost of hindering the scheduling process due to recalculating time quantum for each queue using a recurrent neural network. Our proposed approach implements the best of all worlds regarding [1] and [3] by avoiding each drawback and optimizing their advantages.

II. Related Work

Several papers proposed various types of approaches to improve the overall efficiency of the multilevel feedback queue scheduling algorithm. The chosen quantum time for each queue plays a major role. For instance, in [1], the highest priority queue has been given a relatively low quantum period, and as the priority of a queue decreases, its quantum gets multiplied by a constant. Hence, it is essential to choose a proper method to compute the time quantum value to minimize response time and maximize overall performance. In [2], an algorithm is presented for solving these drawbacks and minimizing the response time. In this algorithm, a Recurrent Neural Network (RNN) was used to determine both the number of queues and the optimized time quantum value for each queue. The RNN generates an effective model to compute the time quantum value. Their proposed intelligent version of the MLFQ offers good results, however, it suffers from a few drawbacks, the first being the direct proportionality of its network learning time and the amount of input data, and the second is the possibility of experiencing initial overhead at the first iterations of the algorithm. Our approach proposes an enhanced version of MLFQ using an optimized version of RR named shortest remaining burst round-robin (SRBRR) as in [3] which avoids the learning and overhead time in [2]. Regarding the various approaches to improve the MLFQ algorithm, those attempts dealt with starvation by assigning different quantum values to the ready queues depending on their priority. Our approach deals with starvation by boosting processes from lower priority queues to higher ones according to a specific policy.

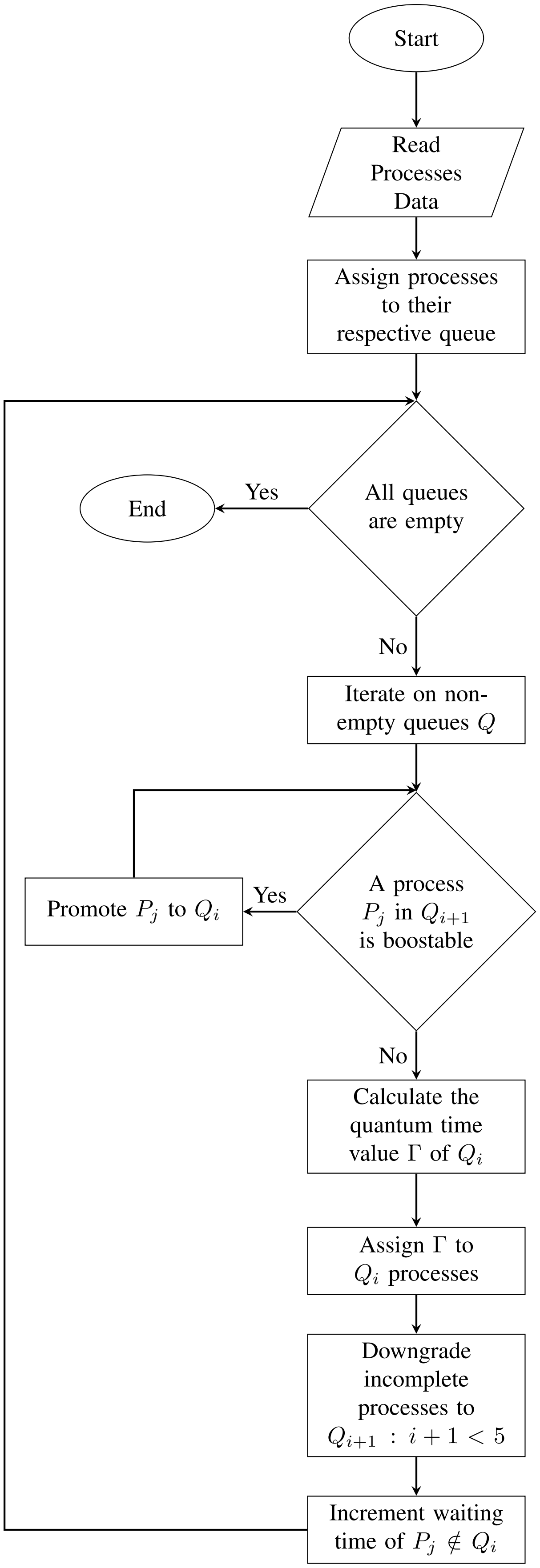
III. Proposed Approach

For a multilevel feedback queue scheduling algorithm, three parameters should be considered. The first being the chosen scheduling algorithm for each queue, especially the last queue as its scheduling algorithm must help treat starvation. The second is the criteria according to which a process priority should be increased based on its waiting time in its particular ready-queue, this technique is also known as Aging. The third is the criteria according to which a process priority should be decreased. In our proposed algorithm, there are 5 queues sorted in ascending order in line with their priority number, 0 being the highest priority and 4 being the lowest. Each queue uses a modified version of the round-robin scheduling algorithm stated in [3]. In [3], processes are sorted in ascending order according to their burst time and are assigned a time quantum that equals the median burst time of those processes. This dynamic quantum time RR algorithm provides better turnaround time and waiting time than the standard static quantum time RR algorithm whose quantum if set too short, leads to many context switches and if set too long, morphs the algorithm into an FCFS algorithm. The proposed alteration on the stated algorithm in [3] is that each queue quantum time isn't exactly the median burst time, but the burst time whose index is shifted from the median value towards the higher burst time values with a step count equal to its queue number, hence, the gradual increase of quantum time as priority decreases. For clarification, a queue with priority equal 2 has the following processes denoted by their burst time: 100, 300, 550, 600, 620, 700, 720, 900, 1200, the median value is 620, assuming we are using zero-based numbering, its index equals 4, and since we are in a queue whose priority equals 2, therefore the quantum slice value according to the proposed approach equals the burst value at index 6 (as in 4 + 2) which is 720. If the processes of a certain queue didn't terminate after assigning its queue quantum time, they are shifted downwards to the next lower priority queue. After introducing new processes into a queue, the quantum time slice is recalculated. Processes age whenever they are in a queue whose priority is one less than that currently getting scheduled and satisfies the following inequality:

$$1 - \frac{waiting\ time\ of\ P}{burst\ time\ of\ P} < 0.95 \qquad (1)$$

Those procedures are repeated for all the generated queues until all processes reach the lowest priority queue where they are rescheduled until their completion.

A. Pseudocode Flowchart of the Proposed Approach



Start
Read Processes Data
Assign processes to their respective queue
All queues are empty
Yes
End
No
Iterate on non-empty queues Q
A process P_j in Q_{i+1} is boostable
Yes
Promote P_j to Q_i
No
Calculate the quantum time value Γ of Q_i
Assign Γ to Q_i processes
Downgrade incomplete processes to $Q_{i+1}\ :\ i+1 < 5$
Increment waiting time of $P_j \notin Q_i$

B. Proposed Algorithm

Algorithm 1 Enhanced Multilevel FeedBack Queue Scheduling Algorithm

```
1: procedure EMLFQ(Number of processes $n$, priority of each process $\alpha_i$, burst time of each process $\beta_i$)
   Declaration and Initialisation:
2:     Queue $Q_i$ where $i \in \{0, 1, 2, 3, 4\}$, waiting time $\omega_i = 0$ of $P_i$, quantum time $\Gamma$, quantum time index $\gamma$
3:     for $i = 0$ to $n$ do
4:         Assign $P_i$ to $Q_{\alpha_i}$
5:     end for
   Loop Process:
6:     $i = 0$
7:     while $Q_i$ is NOT empty do
8:         for $j = i + 1$ to $5$ do
9:             if $1 - \omega_\theta/\beta_\theta < 0.95$ for $\theta \in Q_j$ then
10:                Promote $P_\theta$ to the current queue $Q_i$
11:            end if
12:        end for
   Calculating the quantum time value:
13:        $ascending\_sort(\beta_{Q_i})$
14:        $\gamma = Q_i.size()/2 + i$
15:        $\Gamma = \beta_\gamma$ where $\beta_\gamma \in Q_i$
   Assigning the Quantum Value to each Process:
16:        for $P_j$ in $Q_i$ do
17:            $\beta_j = \beta_j - \Gamma$
18:            if $\beta_j <= 0$ then
19:                Remove $P_j$ from $Q_i$
20:            else if $i < 4$ then
21:                Downgrade $P_j$ to $Q_{i+1}$
22:            end if
23:        end for
   Increment Waiting Time:
24:        for $j = 0$ to $n$ and $P_j \notin Q_i$ do
25:            $\omega_j = \omega_j + 1$
26:        end for
   Increment Queue Index
27:        if $i < 4$ then
28:            $i = i + 1$
29:        end if
30:    end while
31: end procedure
```

IV. Experimental Analysis

A. Assumptions

The proposed scheduling algorithm is software replicated using Python scripts which simulate scheduling independent CPU-bound processes with predetermined burst time, arrival time and the queue to which they belong on a single processor environment which guarantees that no more than a single process uses the CPU at any arbitrary moment.

B. Experimental Scheme

On one hand, the input arguments to the proposed algorithm implementation are the number of processes to be scheduled, burst time, arrival time and the queue to which they belong to. On the other hand, output parameters are the average waiting time, average turnaround time and throughput. The following equations are used to calculate the previously mentioned output parameters:

$$Average\ Waiting\ Time = \frac{Total\ Waiting\ Time}{Number\ of\ Processes} \quad (2)$$

$$Average\ Turnaround\ Time = \frac{Total\ Turnaround\ Time}{Number\ of\ Processes} \quad (3)$$

$$Throughput = \frac{Number\ of\ Executed\ Processes}{Total\ Execution\ Time} \quad (4)$$

C. Performance Metrics

As a means to have a concrete viable evaluation of either the proposed algorithm or any other scheduling algorithm, the output parameters are taken into consideration for analysis. Since the average waiting time indicates the average time that a process has to starve for, therefore the lower the average waiting time is the better. The same principle applies to the average turnaround time and the number of context switches, as the former implies the average time spent by the process since its arrival time to its completion and the latter costs time as the CPU is assigned back and forth between different processes. Contrarily to the prior metrics, the larger the throughput is the better as it indicates the number of processes that are completely executed per unit time.

D. Simulation

For the sake of showcasing the proposed algorithm, a number of processes, their predetermined burst time and arrival time are taken as input to a Python simulation script. Suppose that the input to the script is according to the following table:

TABLE I

Process	*Arrival Time*	*Burst Time*	*Queue*
1	0	60	1
2	0	50	1
3	0	40	2
4	0	30	2
5	0	10	3
6	0	210	3
7	0	200	3

According to the proposed algorithm, the time quanta calculated are as follows:

TABLE II

Queue	*Quantum Value*
1	55
2	40
3	615
4	0
5	0

All processes are sorted ascendingly according to their remaining time scheduled in their respective queue through assigning the time quantum calculated for each queue. The time spent scheduling a particular queue is the waiting time for the subsequent queues.

The scheduling process goes as follows:

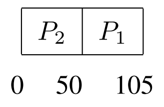


Fig. 1. Q_1 Gantt Chart

Considering that each process in Q_1 is assigned a quantum value of 50, as we reach the last process in Q_1, the total time consumed equals 105, which happens to be the time that all the other processes have to wait for in the subsequent queues hence the increase of their waiting time by 105 units of time.

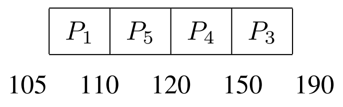


Fig. 2. Q_2 Gantt Chart

Even though P_5 is assigned to Q_3 as in Table IV, it was promoted to Q_2 due to satisfying (1).

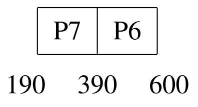


Fig. 3. Q_3 Gantt Chart

Whenever processes reach the lowest queue precompletion, they are scheduled using the RR scheduling algorithm with a relatively large quantum time value which is in most cases similar to using the FCFS algorithm because, as the time quantum value of the RR algorithm tends to infinity which could practically be a very large number relative to the available processes remaining time, the algorithm tends to morph into the FCFS algorithm. This procedure is repeated until all the processes are finished. Simulation results are shown in the table below:

TABLE III

Average Turnaround Time	*Average Waiting Time*	*Throughput*
1	0	60

E. Performance Comparisons

To assess the performance of the proposed algorithm implementation, a varying number of processes is taken in six different experiments as input to the Python simulation script. In each experiment, the output of the proposed algorithm implementation is compared to the output of another scheduling algorithm implementation such as standard MLFQ with static quantum RR and other variants of MLFQ and RR addressed in different papers.

1) Experiment 1: In this experiment, the proposed algorithm results are compared against both a static and a dynamic version of the RR algorithm addressed in [1].

TABLE IV
EXPERIMENT 1 INPUT

Process	*Arrival Time*	*Burst Time*	*Queue*
1	1	25	1
2	5	70	1
3	6	84	1
4	7	17	1
5	8	35	1

TABLE V
EXPERIMENT 1 RESULTS

	Average Turnaround Time	*Average Waiting Time*
Proposed Algorithm	115	68.8
Static RR	161.4	116.2
Dynamic RR	150.8	107.6

V. PREPARE YOUR PAPER BEFORE STYLING

Before you begin to format your paper, first write and save the content as a separate text file. Complete all content and organizational editing before formatting. Please note sections V-A–V-E below for more information on proofreading, spelling and grammar.

Keep your text and graphic files separate until after the text has been formatted and styled. Do not number text heads—LaTeX will do that for you.

A. Abbreviations and Acronyms

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, ac, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

B. Units

- Use either SI (MKS) or CGS as primary units. (SI units are encouraged.) English units may be used as secondary units (in parentheses). An exception would be the use of English units as identifiers in trade, such as "3.5-inch disk drive".
- Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity that you use in an equation.
- Do not mix complete spellings and abbreviations of units: "Wb/m^2" or "webers per square meter", not "webers/m^2". Spell out units when they appear in text: ". . . a few henries", not ". . . a few H".
- Use a zero before decimal points: "0.25", not ".25". Use "cm^3", not "cc".)

C. Equations

Number equations consecutively. To make your equations more compact, you may use the solidus (/), the exp function, or appropriate exponents. Italicize Roman symbols for quantities and variables, but not Greek symbols. Use a long dash rather than a hyphen for a minus sign. Punctuate equations with commas or periods when they are part of a sentence, as in:

$$a+b=\gamma \tag{5}$$

Be sure that the symbols in your equation have been defined before or immediately following the equation. Use "(5)", not "Eq. (5)" or "equation (5)", except at the beginning of a sentence: "Equation (5) is . . ."

D. LaTeX-Specific Advice

Please use "soft" (e.g., `\eqref{Eq}`) cross references instead of "hard" references (e.g., `(1)`). That will make it possible to combine sections, add equations, or change the order of figures or citations without having to go through the file line by line.

Please don't use the `{eqnarray}` equation environment. Use `{align}` or `{IEEEeqnarray}` instead. The `{eqnarray}` environment leaves unsightly spaces around relation symbols.

Please note that the `{subequations}` environment in LaTeX will increment the main equation counter even when there are no equation numbers displayed. If you forget that, you might write an article in which the equation numbers skip from (17) to (20), causing the copy editors to wonder if you've discovered a new method of counting.

BibTeX does not work by magic. It doesn't get the bibliographic data from thin air but from .bib files. If you use BibTeX to produce a bibliography you must send the .bib files.

LaTeX can't read your mind. If you assign the same label to a subsubsection and a table, you might find that Table I has been cross referenced as Table IV-B3.

LaTeX does not have precognitive abilities. If you put a `\label` command before the command that updates the counter it's supposed to be using, the label will pick up the last counter to be cross referenced instead. In particular, a `\label` command should not go before the caption of a figure or a table.

Do not use `\nonumber` inside the `{array}` environment. It will not stop equation numbers inside `{array}` (there won't be any anyway) and it might stop a wanted equation number in the surrounding equation.

E. Some Common Mistakes

- The word "data" is plural, not singular.
- The subscript for the permeability of vacuum μ_0, and other common scientific constants, is zero with subscript formatting, not a lowercase letter "o".
- In American English, commas, semicolons, periods, question and exclamation marks are located within quotation marks only when a complete thought or name is cited, such as a title or full quotation. When quotation marks are used, instead of a bold or italic typeface, to highlight a word or phrase, punctuation should appear outside of the quotation marks. A parenthetical phrase or statement at the end of a sentence is punctuated outside of the closing parenthesis (like this). (A parenthetical sentence is punctuated within the parentheses.)
- A graph within a graph is an "inset", not an "insert". The word alternatively is preferred to the word "alternately" (unless you really mean something that alternates).
- Do not use the word "essentially" to mean "approximately" or "effectively".
- In your paper title, if the words "that uses" can accurately replace the word "using", capitalize the "u"; if not, keep using lower-cased.
- Be aware of the different meanings of the homophones "affect" and "effect", "complement" and "compliment", "discreet" and "discrete", "principal" and "principle".
- Do not confuse "imply" and "infer".
- The prefix "non" is not a word; it should be joined to the word it modifies, usually without a hyphen.
- There is no period after the "et" in the Latin abbreviation "et al.".
- The abbreviation "i.e." means "that is", and the abbreviation "e.g." means "for example".

An excellent style manual for science writers is [**?**].

F. Authors and Affiliations

The class file is designed for, but not limited to, six authors. A minimum of one author is required for all articles. Author names should be listed starting from left to right and then moving down to the next line. This is the author sequence that will be used in future citations and by indexing services. Names should not be listed in columns nor group by affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization).

G. Identify the Headings

Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

Component heads identify the different components of your paper and are not topically subordinate to each other. Examples include Acknowledgments and References and, for these, the correct style to use is "Heading 5". Use "figure caption" for your Figure captions, and "table head" for your table title. Run-in heads, such as "Abstract", will require you to apply a style (in this case, italic) in addition to the style provided by the drop down menu to differentiate the head from the text.

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced.

H. Figures and Tables

a) Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation "Fig. 4", even at the beginning of a sentence.

TABLE VI
TABLE TYPE STYLES

Table Head	**Table Column Head**		
	Table column subhead	***Subhead***	***Subhead***
copy	More table copy[a]		

[a]Sample of a Table footnote.

Fig. 4. Example of a figure caption.

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity "Magnetization", or "Magnetization, M", not just "M". If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write "Magnetization (A/m)" or "Magnetization {A[m(1)]}", not just "A/m". Do not label axes with a ratio of quantities and units. For example, write "Temperature (K)", not "Temperature/K".

ACKNOWLEDGMENT

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks ...". Instead, try "R. B. G. thanks...". Put sponsor acknowledgments in the unnumbered footnote on the first page.

REFERENCES

Please number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first ..."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [**?**]. Papers that have been accepted for publication should be cited as "in press" [**?**]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [**?**].

REFERENCES

[1] H.S. Behera, Reena Kumari Naik, Suchilagna Parida, "Improved multilevel feedback queue scheduling using dynamic time quantum and its performance analysis", *International Journal of Computer Science and Information Technologies*, vol. 3, no. 2, 2012, pp. 3801-3807.

[2] MohammadReza EffatParvar, Karim Faez, Mehdi EffatParvar, Mehdi Zarei, Saeed Safari, "An intelligent MLFQ scheduling algorithm (IMLFQ) with fault tolerant mechanism", *Sixth International Conference on Intelligent Systems Design and Applications*, vol. 3, 2006, pp. 80–85.

[3] Rakesh Mohanty, H. S. Behera, Khusbu Patwari, Monisha Dash, "Design and performance evaluation of a new proposed shortest remaining burst round robin (SRBRR) scheduling algorithm", *International Symposium on Computer Engineering & Technology*, vol. 17, 2010.