International Journal of Computer Applications (0975 – 8887)
Volume 33– No.9, November 2011

Fuzzy Better Job First Scheduling Algorithm

D. Pandey
Dept. of Mathematics
C.C S. University Meerut, India

Vandana
Dept. of Computer Applications
IIMT Group of Inst. Meerut, India

M. K. Sharma
Dept. of Mathematics
RSS College Pilakhuwa, India

ABSTRACT

Time of arrival, size of CPU burst and priority are three major factors that are usually attached with a process which is submitted for execution. Several scheduling policies exist which use one or the other of these factors to place the submitted processes in appropriate order in the ready queue. Each of the existing schedulers has some positive and negative implications by way of assigning individual importance to any of the three factors. This work is an attempt to analyze the collective effect of time of arrival, size of CPU burst and priority of the process, through a logical combination of all the three. Fuzzy Better Job First (FBJF) scheduling algorithm logically integrates these three factors of a process and uses fuzzy ranking approach to determine the next most worthy job to be executed. The proposed policy thus enjoys advantages all the criteria to considerable extent.

General Terms

Arrival time, CPU burst, Priority, CPU Scheduling algorithm.

Keywords

Ready queue, Process, Average waiting time(AWT), HRRN, Membership function, Fuzzification, Defuzzification.

1. INTRODUCTION

It is well known that three major considerations are generally attached with a process when it is submitted to a scheduling system for the purpose of execution. These are its time of arrival, size of its CPU burst and the priority assigned with the process, if any. Each of these has its own importance. It depends on the design of the scheduler as to which of these considerations is given main importance. There are some basic algorithms that possess considerations for one or the other. FCFS looks for arrival time, SJF bothers about the size of burst and priority scheduling gives preference in accordance with the assigned priorities. In FCFS, processes are executed in the order of arrival, without any regard to its size and priority. Although an attempt to draw the advantage of size of CPU burst in FCFS has been done in [1], yet FCFS is particularly troublesome for time-sharing systems, where it is important that each user gets a share of CPU at regular intervals. It would be disastrous to allow one process to keep the CPU busy for an extended period of time. Shortest job first is an approach to reducing the bias inherent in FCFS, in favor of long processes. But this algorithm focuses only on smallest CPU burst, without any concern to the priority or its arrival order. FCFS and SJF algorithms select the processes in their order of arrival and burst size respectively, but both of these give no consideration to priority. Conventional operating systems employ numerical priorities for scheduling processes [2]. A priority scheduler simply grants the processor to the process with the highest priority. One of the problems with a pure priority scheduling scheme is that lower-priority processes may suffer starvation, if there is always a steady supply of higher-priority ready processes. Priority aging is a common technique that gradually increases the priorities of processes that have been waiting to execute for a long time [3]. Having observed several positive and negative implications of assigning individual importance to arrival order, burst length or priorities in different scheduling algorithms, it sounds proper to analyze their collective effect through a logical combination of all the three. This will help the scheduler in determining the next most worthy job to be executed. An effort to mix the functions of some basic scheduling algorithms has been done by Al-Husainy [4]. He introduced a new factor f that represented the mixed effect produced from the combination of three factors. His approach relies on the term *percentage ratio,* which involves imprecision in its calculation and judgment for appropriateness. Pandey and Vandana[5] have also worked in the same direction. Referring Fuzzy logic Aburus and Miho [6] suggested an improvement over HRRN. Kadhim and Al-Aubidy[7] combined the effect of priority and execution time to get new priority. This paper designs a new scheduler that uses fuzzy logic to combine the importance attached to the arrival order, CPU burst length, and priority of the process to select a better job to be executed. The approach is based on fuzzy ranking. AWTs have been computed on different randomly generated data-sets and the results have been compared with FCFS, SJF and Round Robin policies to justify the approach.
The contents of this paper are organized as follows: In Section 2, fuzzy ranking method for fuzzy better job first algorithm has been introduced. Section 3 defines the algorithm and its method of implementation. Results on different data-sets have been discussed in Section 4, while the last section highlights the conclusions.

2. FUZZY RANKING METHOD

Fuzzy ranking concept is used to order the processes for the purpose of execution on the processor by taking into consideration the combined effect of three attributes. This method is based on fuzzy logic that combines three input attributes: Arrival order, CPU burst and Priority of a process. Each of these attributes is classified into different linguistic categories; for example, priority of a process can be *High, Normal* and *Low*. Likewise CPU estimates can be *Short, Medium* and *Long* and arrivals can be *Early, Intermediate, Late etc*. Fuzzy sets representing their linguistic concepts are then defined for each of the attributes; *Arrival, CPU burst and Priority*. The least common multiple of the ranges of three attributes or a suitable fraction of it is chosen to be a common range for defining membership functions. Trapezoidal membership functions may prove to be a better representation of linguistic concepts. The linguistic terms are the fuzzy variables representing the state of the process corresponding to each attribute.
Every admitted process is assigned a fuzzy variable corresponding to each attribute in proportion to the common range. The fuzzy variables are then used to combine the three attributes of the process in the form of a new fuzzy set in accordance with a *fuzzy rule base*. This is performed by using fuzzy aggregation operator to obtain fuzzy ranking. The new fuzzy set is defuzzified to obtain a crisp output value for every process. Crisp output values are then sorted in an increasing order of values to provide crisp ranking to the processes for

execution on the processor. The entire fuzzy ranking mechanism can easily be implemented through a *Fuzzy Logic Controller.*

3. ALGORITHM AND IMPLEMENTATION

A general sequence of Fuzzy Better Job First (FBJF) algorithm with fuzzy ranking approach is listed below:

Step 1: Define linguistic categories such as *low, medium, high etc.* to be used for the time of arrival, size of CPU burst and priority of the processes. Linguistic category of each attribute is attached with every admitted process.

Step 2: Define membership function distributions representing the linguistic concept.

Step 3: Set-up a fuzzy rule base on the basis of the number of categories defined for each attribute in Step 1. If *l, m, n* denote the number of categories defined for time of arrival, size of burst, priority respectively then the fuzzy rule base will have ($l \times m \times n$) rules.

Step 4: Use standard fuzzy union to aggregate the membership functions of three attributes attached with each process in accordance with the rule base. This produces a single fuzzy set combining the three attributes of the process.

Step 5: Use centroid method to defuzzify the single fuzzy set obtained in Step 4. This gives a crisp output value for each process.

Step 6: Rank the processes in ascending order of output values. The ranks serve as the order of execution of the processes. Go to Step 7 for ranking the processes having equal output values.

Step 7: Processes having same output values shall have same rank. To order the processes having same rank, any of the following procedure may be adopted as per the suitability of requirement:

(a) Order the same rank processes in the order of their time of arrival. This scheduling algorithm is called **Fuzzy Better Job First (Arrival),** in short **FBJF (A)**.

(b) Order the same rank processes in the order of their CPU burst size. This scheduling algorithm is called **Fuzzy Better Job First (Burst),** in short **FBJF (B)**.

(c) Order the same rank processes in the order of their priorities. This scheduling algorithm is called **Fuzzy Better Job First (Priority),** in short **FBJF(P)**.

To understand FBJF algorithm, we observe its implementation through the following example of fifteen processes. Processes with randomly drawn time of arrival, size of CPU bursts and assigned priorities are presented in Table 1.

Table 1: Processes with attribute values

P- ID	CPU Burst	Arrival Time	Priority
1	1.48	0.19	2
2	5.16	1.12	5
3	9.71	1.84	2
4	2.28	3.28	3
5	7.11	3.45	4
6	1.10	4.45	5
7	0.41	5.64	1
8	3.74	6.49	3
9	2.07	7.62	2
10	1.78	9.18	4
11	2.15	10.02	4
12	0.81	11.65	3
13	4.24	12.50	2
14	9.84	13.17	4
15	8.78	14.45	1

Let us define the CPU-bursts of all processes into following linguistic categories:

Short burst : $0 \le x_i \le 3$;

Medium burst: $4 \le x_i \le 6$

Long burst: $7 \le x_i \le 10$.

The time of arrivals of all processes are defined into following linguistic categories:

Early arrival : $0 \le x_i \le 5$;

Intermediate arrival: $6 \le x_i \le 10$;

Late arrival: $11 \le x_i \le 15$.

The priorities of all processes are defined as following:

High priority : $1 \le x_i \le 2$;

Low priority: $3 \le x_i \le 5$.

Fuzzy sets for Arrival time, Burst size and Priority are defined by following trapezoidal membership functions:

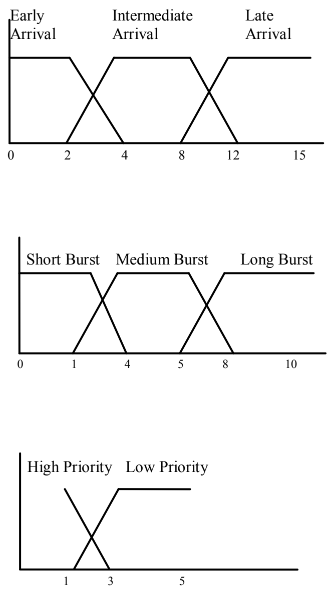


Fig. 1: Membership functions for three attributes

It may be pointed out here that to ignore the effect of any attribute, we must define the membership function corresponding to that attribute to be zero for the entire range.

In compliance to step 3, we set-up a fuzzy rule base, that consists of following 18 ($3 \times 3 \times 2$) rules of fuzzy logic.

IF *arrival* is *early*, *CPU burst* is *short* and *priority* is *high* THEN *output is* O_1.

IF *arrival* is *early*, *CPU burst* is *short* and *priority* is *low* THEN *output is* O_2.

IF *arrival* is *early*, *CPU burst* is *medium* and *priority* is *high* THEN *output is* O_3.

IF *arrival* is *early*, *CPU burst* is *medium* and *priority* is *low* THEN *output is* O_4.

IF *arrival* is *early*, *CPU burst* is *long* and *priority* is *high* THEN *output is* O_5.

IF *arrival* is *early*, *CPU burst* is *long* and *priority* is *low* THEN *output is* O_6.

IF *arrival* is *intermediate*, *CPU burst* is *short* and *priority* is *high* THEN *output is* O_7.

IF *arrival* is *intermediate*, *CPU burst* is *short* and *priority* is *low* THEN *output is* O_8.

IF *arrival* is *intermediate*, *CPU burst* is *medium* and *priority* is *high* THEN *output is* O_9.

IF *arrival* is *intermediate*, *CPU burst* is *medium* and *priority* is *low* THEN *output is* O_{10}.

IF *arrival* is *intermediate*, *CPU burst* is *long* and *priority* is *high* THEN *output is* O_{11}.

IF *arrival* is *intermediate*, *CPU burst* is *long* and *priority* is *low* THEN *output is* O_{12}.

IF *arrival* is *late*, *CPU burst* is *short* and *priority* is *high* THEN *output is* O_{13}.

IF *arrival* is *late*, *CPU burst* is *short* and *priority* is *low* THEN *output is* O_{14}.

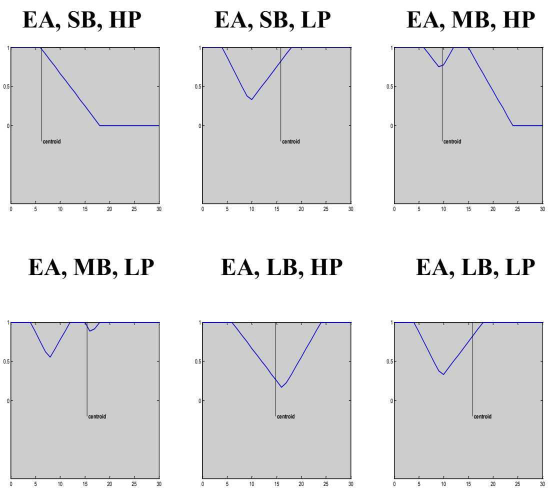
IF *arrival* is *late*, *CPU burst* is *medium* and *priority* is *high* THEN *output is* O_{15}.

IF *arrival* is *late*, *CPU burst* is *medium* and *priority* is *low* THEN *output is* O_{16}.

IF *arrival* is *late*, *CPU burst* is *long* and *priority* is *high* THEN *output is* O_{17}.

IF *arrival* is *late*, *CPU burst* is *long* and *priority* is *low* THEN *output is* O_{18}.

As explained in steps 4 and 5, the output values O_1 *to* O_{18} are obtained by defuzzyfying the single combined fuzzy set by centroid method. In this method the centroid of the single fuzzy set gives the output value. We present in Fig. 2, the output values O_1 *to* O_{18} determined by the centroid of the new membership function obtained using MATLAB.



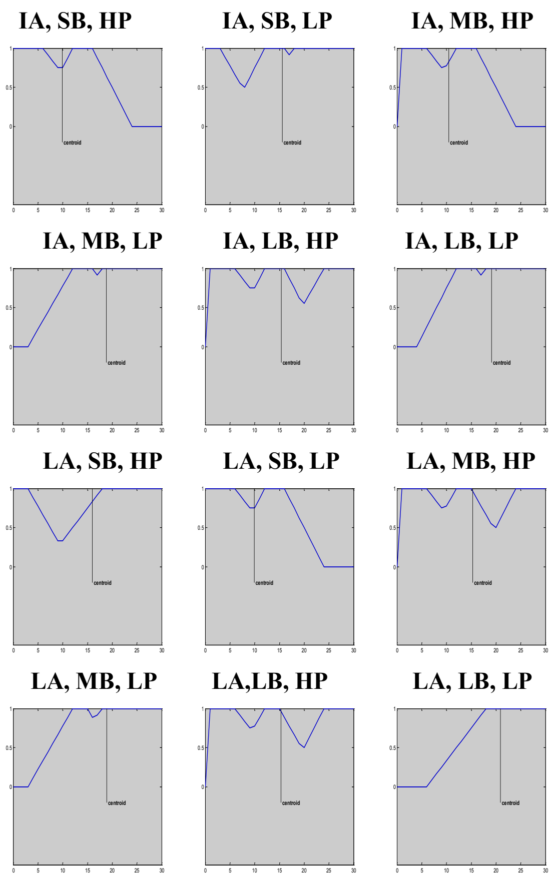


Fig 2: Output values by centroid method

Following Step 6, output values are used to rank the processes and the same rank cases are handled as suggested in Step 7. Table 2 lists the output value for each process and the order of execution of the processes according to all variants of FBJF, that is, FBJF(A), FBJF(B) and FBJF(P).

Table 2: Output values and Ranks

P-ID	Output Values	FBJF(A)	FBJF(B)	FBJF(P)
1	6.220	1	1	1
2	15.830	8	8	8
3	18.880	14	14	14
4	15.440	4	5	4
5	15.835	9	9	9
6	15.440	5	4	5
7	9.909	2	2	2
8	18.860	12	12	12
9	9.909	3	3	3
10	15.607	7	7	7
11	16.008	10	11	11
12	16.008	11	10	10
13	18.870	13	13	13
14	20.920	15	15	15
15	15.526	6	6	6

Average Waiting Time (AWT) under different scheduling policies is computed for this example, using the data given in Table 1. It can be observed from the following table that the performances of all variants of FBJF are better than both, the FCFS and Priority scheduling and are tolerably comparable to SJF.

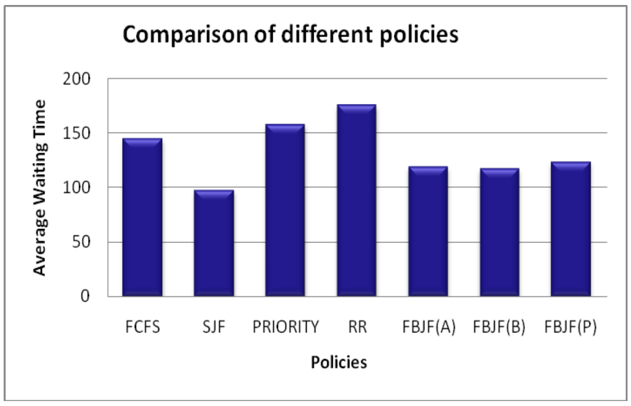
Table 3: Average waiting time under different policies

Policy	FCFS	SJF	Priority	FBJ (A)	FBJF (B)	FBJF (P)
AWT	26.04	15.22	26.74	20.16	19.99	20.07

4. RESULTS AND DISCUSSIONS

We worked on several randomly generated data sets to observe the performance of FBJF scheduling algorithm. Results presented in this paper correspond to three data-sets of 50 processes each. Different ranges of arrival-time, burst-size and numerical priorities have been considered. We have defined three linguistic categories for each attribute: Burst (*Short, Medium, Long*), Arrival time (*Early, Intermediate, Late*), Priority (*High, Normal, Low*). Same data-sets were used to evaluate the average waiting time for different scheduling policies, in order to compare the performance of the proposed scheduling algorithm against them. Values for average waiting time computed according to five different scheduling policies on three data-sets are presented through bar graph in Fig. 3, 4 and 5 below. It can be observed that the performance of each variant of the proposed algorithm on average waiting time is significantly better than FCFS, Priority scheduling and Round Robin (with quantum 5) policies and is tolerably comparable to SJF. However all variants of FBJF are competitive within themselves.

Fig. 3: Results of data-set 1



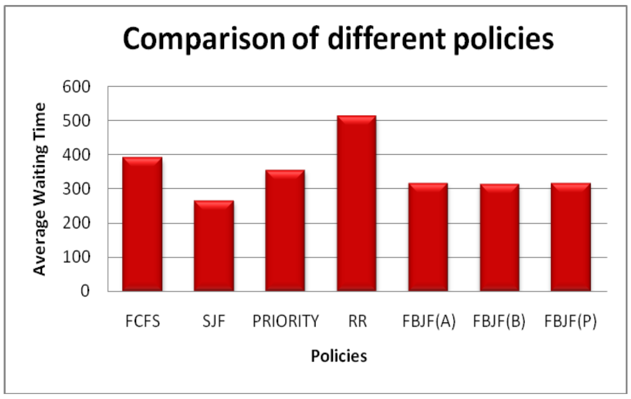


Fig. 4: Results of data-set 2

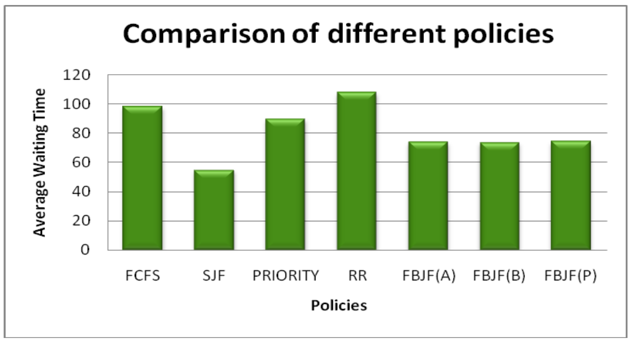


Fig. 5: Results of data-set 3

5. CONCLUSIONS

Fuzzy Better Job First scheduling algorithm proposed in this paper uses fuzzy set theoretic approach to combine three attributes (order of arrival, size of CPU burst and priority) associated with an admitted process, to help the scheduler to select the most worthy job to be processed next. Fuzzy ranking method has been applied to order the processes in the ready queue after integrating all the three factors. The philosophy of FBJF policy permits it to enjoy advantages of all included attributes to some extent. Despite taking the combined effect of three attributes into consideration, this policy is capable of providing an additional weightage to any particular attribute through its variants. Our algorithm can be used to function as purely SJF by choosing fuzzy sets corresponding to *Arrival* and *Priority* having membership functions to be zero for the entire range. Likewise FCFS and pure Priority scheduling can be performed through this algorithm. Further from the comparison of results of FBJF with other scheduling algorithms on average waiting time recorded in Section IV, it is evident that except for SJF (which is more a theoretical approach), its performance is better than other practical scheduling methods, such as FCFS, Priority and Round robin scheduling. Moreover, the rule base in FBJF operates on linguistic categories instead of individual data of the processes, therefore our algorithm will perform more efficiently than Al-Husainy[4] for the system having large number of processes.

6. REFERENCES

[1] D. Pandey, Vandana and M.K. Sharma, "CPU Scheduling: FCFS with Shorter Processes First", MR International Journal of Engineering and Technology, 1 (2), 2008, pp 11-17.

[2] H. M. Deital, "Operating Systems", Pearson, 2006.

[3] William Stallings, "Operating Systems- Internals and Design Principles", Pearson, 2006.

[4] A.F. Al-Husainy Mohammed, "Best-Job first CPU Scheduling Algorithm", Information Technology Journal 6(2), 2007, pp 288-293.

[5] D. Pandey and Vandana "Weighted Approach To Better Job First Scheduling", Journal of International Academy of Physical Sciences, 14(1), pp 101-112, 2010.

[6] Abdurazzag Ali Aburas, Vladimir Miho ,"Fuzzy Logic Based Algorithm for Uniprocessor Scheduling", Proceedings of the International Conference on Computer and Communication Engineering, Kuala Lumpur, Malaysia, pp 499-504, May 13-15, 2008.

[7] Shantha J. Kadhim and Kasim Al-Aubidy "Design and Evaluation of a Fuzzy-Based CPU Scheduling Algorithm", CCIS 70, pp 45-52, 2010.