FUZZY BETTER JOB SCHEDULING ALGORITHM

FACULTY:

Prof.ARINDAM DEY

TEAM MEMBERS:

ASHISH-19BCN7196

LOKESH-19BCN7242

USHA-19BCN7175

K.R.S.PRANATHI-19BCE7656

ABSTRACT

Non-deterministic polynomial-hard issues such as fuzzy job-shop scheduling problems (Fuzzy JSSPs) are a type of combinatorial optimization problem. Several academics have expanded the theoretical models of Fuzzy JSSPs and introduced techniques to solve them in recent decades. The classification of Fuzzy JSSPs, the constraints and objectives examined in Fuzzy JSSPs, and the approaches used to solve Fuzzy JSSPs are all covered in this study. The focus of the paper is on a review of meta-heuristic algorithms as state-of-the-art algorithms for Fuzzy JSSPs. Pre-processing, initialization techniques, and improvement algorithms are all investigated in three steps. Finally, this poll yielded possible research ideas for future studies.

INTRODUCTION

In the industry, economics, and management, responding quickly to a job-shop scheduling problem (JSSP) is critical. This problem belongs to the non-deterministic polynomial-hard problems class of combinatorial optimization difficulties [34]. In actuality, the assumption that JSSP duration periods in real-world problems have precise values is frequently violated. Because JSSPs contain human-centered variables, incorporating fuzzy processing time owing to man-made factors and fuzzy due date, which allows for a certain level of lateness, may be more acceptable. As a result, a novel type of JSSP called Fuzzy JSSP [9,33] has gained popularity. Since the mid-1960s, a large number of academics have improved theoretical models of crisp JSSPs, created new methods to solve them, and published thousands of publications in this field. The domain Fuzzy JSSP, on the other hand, was initially presented in 1995. This do-main presently has less than 60 publications available. The variety of methods for solving Fuzzy JSSPs that have been examined has also been unsatisfactory, demonstrating that research on Fuzzy JSSPs is still in its early stages. For the first time, this aspect motivates the current paper to gather and review the extant literature on Fuzzy JSSPs. This work has three goals: (1) gather available material and background on the Fuzzy JSSP domain; (2) identify potential gaps in this domain; and (3) encourage researchers to fill identified gaps in future studies. The following is the structure of this document. The overview and classification of Fuzzy JSSPs are presented in Section 2. The variation in constraints and objectives addressed in Fuzzy JSSPs is discussed in Section 3. The exact and heuristic techniques used for Fuzzy JSSPs are described in Section 4. Pre-processing, initialization methods, and improvement algorithms are all covered in Section 5 of meta-heuristic algorithms. The benchmark datasets for Fuzzy JSSPs are explained in Section 6. Finally, Section 7 summarizes our study and highlights the most important aspects for future research.

GENERAL TERMS

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

KEYWORDS

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

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 a 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.*

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´m´n*) rules.

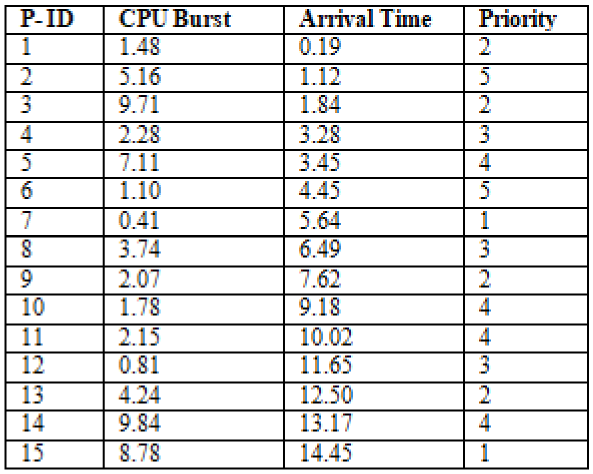
Step 4: Use standard fuzzy union to aggregate the membership functions of three attributes attached with each process in accordance with the rule base. This produces a single fuzzy set combining the three attributes of the process.

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 the same output values shall have the 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 the 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.

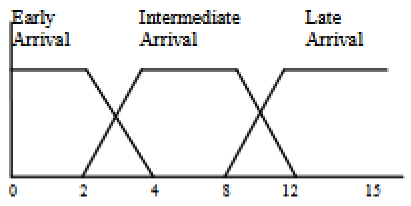


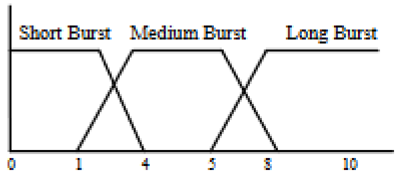
P-ID	CPU Burst	Arrival Time	Priority
1	1.48	0.19	2
2	5.16	1.12	5
3	9.71	1.84	2
4	2.28	3.28	3
5	7.11	3.45	4
6	1.10	4.45	5
7	0.41	5.64	1
8	3.74	6.49	3
9	2.07	7.62	2
10	1.78	9.18	4
11	2.15	10.02	4
12	0.81	11.65	3
13	4.24	12.50	2
14	9.84	13.17	4
15	8.78	14.45	1

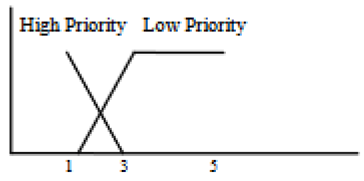
Table 1: Processes with attribute values

Let us define the CPU-bursts of all processes into following linguistic categories:
Shortburst: 0<=xi<=3;
Medium burst: 4<=xi<=6;
Longburst: 7<=xi<=10.
The time of arrivals of all processes are defined into following linguistic categories:
Early arrival : 0<=xi<=5;
Intermediate arrival: 6<=xi<=10;
Late arrival: 11<=xi<=15.
The priorities of all processes are defined as following:
High priority : 1<=xi<=2;
Low priority: 3<=xi<=5;

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







1. 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.

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

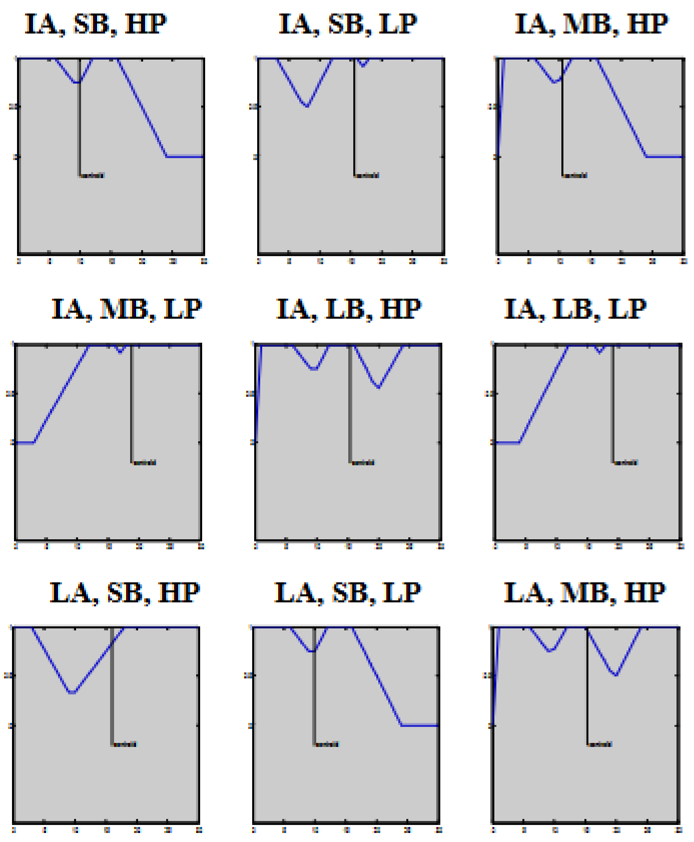
IF *arrival* is *early*, *CPU burst* is *short* and *priority* is *high* THEN *output is O*1 .
IF *arrival* is *early*, *CPU burst* is *short* and *priority* is *low* THEN *output is O*2 .
IF *arrival* is *early*, *CPU burst* is *medium* and *priority* is *high* THEN *output is O*3 .
IF *arrival* is *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 6O .
IF arrival is intermediate, CPU burst is short and priority is high THEN output is O7 .
IF arrival is intermediate, CPU burst is short and priority is low THEN output is O8 .
IF arrival is intermediate, CPU burst is medium and priority is high THEN output is O9 .
IF arrival is intermediate, CPU burst is medium and priority is low THEN output is O10 .
IF arrival is intermediate, CPU burst is long and priority is high THEN output is O11 .
IF arrival is intermediate, CPU burst is long and priority is low THEN output is O12 .
IF arrival is late, CPU burst is short and priority is high THEN output is O13 .
IF arrival is late, CPU burst is short and priority is low THEN output is O14 .
IF arrival is late, CPU burst is medium and priority is high THEN output is O15 .
IF arrival is late, CPU burst is medium and priority is low THEN output is O16 .
IF arrival is late, CPU burst is long and priority is high THEN output is O17 .
IF arrival is late, CPU burst is long and priority is low THEN output is O18 .

3. As explained in steps 4 and 5 , the output values O1 to O18 are obtained by defuzzifier the single combined fuzzy set by the centroid method.

4. In this method the centroid of the single fuzzy set gives the output value.

5. We present in Fig. 2, the output values O1 to O18 determined by the centroid of the new membership function obtained using MATLAB.



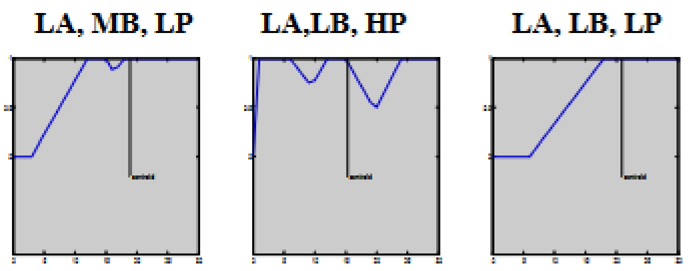


Fig 2: Output values by centroid method

6. 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) andFBJF(P).

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

Table 2: Output values and Ranks

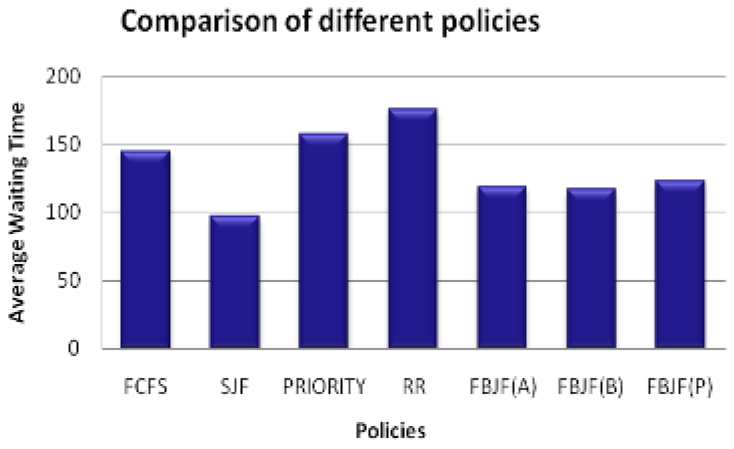
Average Waiting Time (AWT) under different scheduling policies is computed for this example, using the data given in Table 1. It can be observed from the following table that the performances of all variants of FBJF are better than both the FCFS and Priority scheduling and are tolerably comparable to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

Policy	FCFS	SJF	Priority	FBJ (A)	FBJF (B)	FBJF (P)
AWT	26.04	15.2 2	26.74	20.16	19.99	20.07

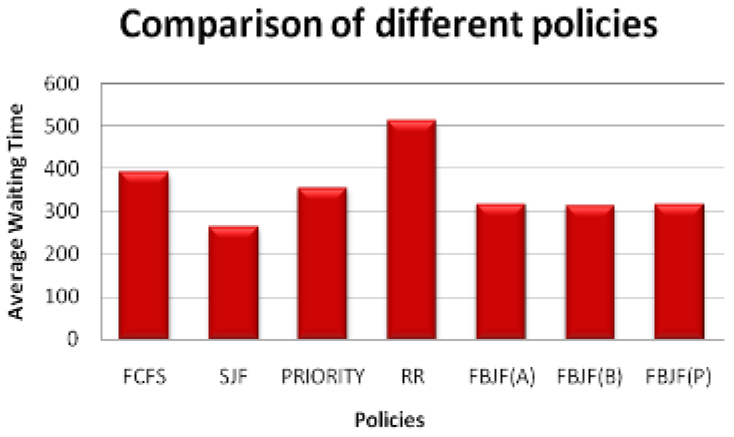
Table 3: Average waiting time under different policies

RESULTS AND DISCUSSIONS

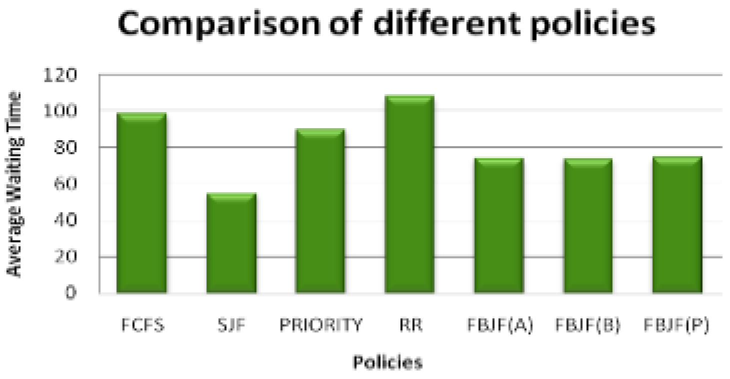
To test the performance of the FBJF scheduling method, we used multiple randomly generated data sets. The findings in this research are based on three data sets, each with 50 processes. Various arrival-time, burst-size, and numerical priority ranges were investigated. Each feature has three language categories: burst short, medium, and long), arrival time (early, intermediate, late), and priority (High, Normal, Low). The average waiting time for different scheduling strategies was evaluated using the same data sets in order to compare the performance of the proposed scheduling algorithm to them. A bar graph is used to display the average waiting time determined using five different scheduling policies on three data sets. It can be seen that each variant of the suggested algorithm performs differently in terms of average waiting time.



RESULTS FOR DATASET 1



RESULTS FOR DATASET 2



RESULTS FOR DATASET 3

CONCLUSIONS

Better Job, Fuzzy The first scheduling algorithm suggested in this study uses a fuzzy set theoretic approach to integrate three attributes connected with an admitted process (order of arrival, amount of CPU burst, and priority) to assist the scheduler in selecting the most meritorious job to be handled next. After combining all three parameters, the fuzzy ranking algorithm was used to order the processes in the ready queue. The philosophy of FBJF policy allows it to benefit to some extent from all of the included features. Despite considering the overall influence of three attributes, this policy's versions can give any particular attribute more weight. By selecting fuzzy sets appropriate to Arrival, our technique can be employed as a pure SJF.