VOL. 12, NO. 21, NOVEMBER 2017 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

LEAST MAKESPAN SCHEDULING ALGORITHM FOR THE HIERACHICAL COMBINED SYSTEMS WITH JOBSHOP CHARACTERISTICS

Y. -J. Lee
Department of Technology Education, Korea National University of Education, South Korea
E-Mail: lyj@knue.ac.kr

ABSTRACT

This paper aims to present the scheduling algorithm to minimize the makespan for the combined systems which has jobshop characteristics. Given the flow and hierarchy structure and the deterministic operation time of each job are given respectively, we derive the extended scheduling algorithm by modifying an existing jobshop algorithm. In addition, we present the algorithm application procedure to satisfying several different operation cases. Our proposed algorithm can be used to build the scheduling policy for the assembly industries with different flow structure and jobshop characteristics.

Keywords: scheduling algorithm, Jobshop characteristics, makespan, hierachical jobs.

1. INTRODUCTION

We face many scheduling problems under widely varying situations in our daily life [1, 2]. The scheduling problem arose originally in industrial and manufacturing firms. However, in various other application areas such as transportation system, computer operating system, and Internet of Things (IoT) environment [3, 4, 5], an efficient scheduling policy is required.

By investigating an efficient scheduling algorithm, we can obtain the sequencing order on each machine in the production field and process order on the CPU to minimize the corresponding cost. Hence, cost is closely related to the timespan from the job start to the job finish. Timespan is represented as the makespan in the production field, and turnaround time in the computer system. Therefore, in order to shorten the timespan, the objective of scheduling algorithm is to find out the optimal or near optimal schedule.

In terms of the flow characteristic of jobs, scheduling problems can be classified into flow-shop problem and jobshop problem [6, 7, 8]. In the former problem, each job is handled by the processor in the same order, whereas in latter problem, the handling order by the processor may vary according to the job.

In this study, we propose a novel scheduling algorithm for the system composed of individual jobs and hierarchical combined jobs (CPT). CPT is composed of several individual jobs and processed after preceding jobs are finished. In our system, each job including CPT has jobshop characteristics.

In our algorithm which extends the result of [9], the decision variable is the processing sequence of job operation on each processor which minimizes total makespan [10, 11, 12, 13]. Our solution can be used in the production system, IoT environment, and mobile network [14, 15] with hierarchical combined jobs. Some numerical examples show that our algorithm can obtain a relatively good solution in a short computation time.

The rest of the paper is organized as follows. Section 2 presents modelling and algorithm for the proposed system, and section 3 presents a numerical example, followed by concluding remarks in section 4.

2. MODELING AND SCHEDULING ALGORITHM FOR SYSTEM WITH THE COMBINED JOBS

2.1 System overview

In our proposed system, job structure is depicted in Figure-1. CPT 1 can be started after both job 1 and job 2 are finished. CPT 2 is processed after job 3 and CPT 1 are finished. CPT 3 is started after Job 4 and CPT 2 are finished. This combined job can be finished after CPT 3 and job 5 are completed.

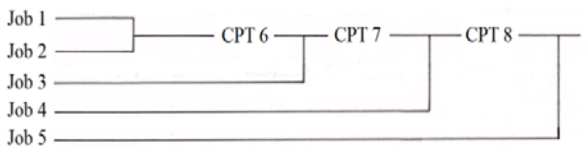

Figure-1. Job structure with CPTs and hierarchical jobs.

Each job requests different processors with its own sequence and needs different processing times. This structure means that each job has typical jobshop characteristic. However, this paper additionally considers the hierarchical combined jobs (CPT) that the typical jobshop problem does not.

There are several shops in the jobshop production field. Each shop is distinguished by its own task and has several processors with individual operation. In Figure-1, for example, Job 3 passes the processor 1 for 1.7 unit time in the shop 1. Job 1 passes the processor 2 and 3 for 2.0 and 1.8 units, respectively in the shop 2. Job 1 and Job 2 are assembled into CPT 1.

2.2 Modelling

Before building the model, we first consider the following assumptions.

- Job structure and flow structure including the deterministic processing time for each operation are given. Several jobs may be combined into one CPT which has jobshop characteristic.

- Each job operation time is non-preemptive, that is, operation time slice is not allowed. Each job may require several times operations on the same processor.
- Each job has jobshop characteristics, which is not undirected and several operations with the same linear precedence. There is no initial processor performing only the first operation of a job, nor terminal processor.

We introduce the following notations. PS_t represents a partial schedule containing t scheduled operations. S_t represents the set of schedulable operations at stage t, corresponding to a given PS_t. E_j and C_j show the earliest times at which the operation $j \in S_t$ could be started and finished, respectively. Then our model is described in (1).

$$\text{Minimize} \quad C^* = \{C_j\},\ j \in S_t \qquad (1)$$

Therefore, we have to find out the processor on which C^* can be obtained in (1).

We consider now previous priority rules for scheduling. SPT (Shortest Processing Time) rule selects the operation with the minimum processing time. FCFS (First Come First Served) rule selects the operation that entered S_t earliest. MWKR (Most Work Remaining) rule selects the operation associated with the job having the most work remaining to be processed. MOPNR (Most Operations Remaining) rule selects the operation that has the largest number of successor operations. LWKR (Least Work Remaining) rule selects the operation associated with the job having the least work remaining to be processed. RANDOM rule select the operation at random [2].

Until now, there is no single priority rule to satisfy all jobshop problems. However, for makespan minimization problem, MWKR rule is known to be most powerful among the above mentioned priority rules. That is, to assign the job with a heavy unprocessed workload first leads to minimize the makespan.

Additionally, in most of solution methods including branch and bound and simulation, the computation time increases sharply as problem size increases. We, therefore, select MWKR rule as first choice rule and SPT rule as the tie breaker of MWKR rule. This schedule routine is already presented in [2]. However, that routine does not consider CPT and priority for a variety of types of jobs. We modify and extend the previous schedule routine in order to apply to the system described in Section 2.1.

2.3 Algorithm

We determine the earliest start time (E_j) of CPT. We let its preceding jobs and level, and finish time to be nc_{ip}, v_i, and $f(nc_{ip})$, respectively. For example, in Figure-1, the level of CPT 2 is higher than that of CPT 1.
By using (2), we can compute E_j of CPT j.

$$E_j = \text{Max}\ [\ f(nc_{ip})],\ p \in i \text{ and } j \in S_t \qquad (2)$$

Solution algorithm for our proposed system is presented in Figure-2.

```
Algorithm Scheduling for Combined Jobs
Stage 1: for jobs with multi processors and jobs
             included in CPT, Call scheduling function;
Stage 2: for every CPT,
           from lower level to higher level on the CPT,
                 Compute $E_j$ by using equation (2);
                 Call scheduling function;
Stage 3: for jobs with single processor,
             Call scheduling function;
Stage 4: STOP

Scheduling Function
Step 1: Set $t = 0$ and begin with $PS_t$ as the null.
          for jobs with single processor $i$, if $K_i == 1$,
          we exclude job $i$. $K_i$ represents the number of
          operations for job $i$. Initially, $S_t$ includes all
          operations with no predecessors.
Step 2: Determine the processor $w^*$ satisfying equation (1).
Step 3: for each operation $j \in S_t$ that requires processor $w^*$
          and for which $E_j < E^*$, compute a priority index
          according to MWKR and SPT priority rules.
          for jobs with single processor, , if $K_i == 1$,
              we exclude job $i$.
          for other cases, if $j == X_i$, then we exclude job $i$.
              $X_i$ represents the last operation of job $i$.
Step 4: Find the operation with the smallest index and add
          this operation to $PS_t$ as early as possible, thus
          creating only one partial schedule, $PS_{t+1}$.
          In this case, if $j == X_i$ or $K_i == 1$,
                then we exclude job $i$.
          if all operations $j$, $\forall\ j \in S_t$ are assigned,
                then goto STEP 5.
          otherwise goto STEP 2.
Step 5: for job $i$ with $K_i == 1$, Add operation time of job $i$
          to the finish time of processor $w$ which is
          obtained from STEP 1 to STEP 4.
Step 6: for job $i$ with $j == X_i$, repeat STEP 5.
Step 7. return
```

Figure-2. Scheduling algorithm for the combined jobs.

In stage 1 of Figure-2, we first assign jobs with multi processors. In order to minimize the makespan, we select a job with most work remaining time. If we assign a job with single processor first, the next schedule is affected by that job. Since jobs with single processor does not demand any operation on other processor, to assign jobs with multiple processors and jobs with single processor simultaneously may delay the makespan. Thus, we should assign jobs with multi processors before jobs with single processor.

Secondly, we assign jobs included in CPT. Since each CPT has to be handled after its preceding jobs or CPTs were processed, it is reasonable that jobs included in CPT must be assigned before CPTs.

In stage 2, each CPT is composed of jobs or/and CPTs that have level showing the processing order. Therefore, CPT structure should be known. When we assign a certain CPT, the assign order is given from lower level to higher level gradually. The reason is why the CPT with lower level must be processed earlier than the CPT with higher level. The earliest start time (E_j) for the CPT is computed by using equation (2).

In stage 3, jobs with single processor are classified into jobs with jobshop characteristic and jobs with only one operation. It is obvious that assigning job with the jobshop characteristic prior to jobs with only one operation is to minimize the makespan.

3. NUMERICAL EXAMPLE

So far, we describe the modelling and solution algorithm for the system composed of hierarchical combined jobs. The job structure of example is given in Figure-1. Flow structure of jobs and CPTs are presented in Table-1. Processing time of jobs and CPTs is given by Table-2.

Table-1. Flow structure of jobs and CPTs

Job	operation (processor)		
	1	2	3
Job 1	3	2	1
Job 2	2	3	1
Job 3	2	1	3
Job 4	2	1	
Job 5	2		
CPT 6	5		
CPT 7	6	5	
CPT 8	5		

Table-2. Processing time of jobs and CPTs

Job	operation (processor)		
	1	2	3
Job 1	4	3	4
Job 2	4	4	2
Job 3	2	5	5
Job 4	2	3	
Job 5	6		
CPT 6	3		
CPT 7	2	4	
CPT 8	3		

We apply our scheduling algorithm given in Figure-1 to the example and then obtain Table-3.

Table-3. Application results of scheduling algorithm

Stage	Description	
1	We set an initial vector for processor start time, $b_k = (0,0,0)$. We find S_0 = {(1,1,3), (2,1,2), (3,1,2)}. A triplet (i, j, w) denotes that operation j of job i requires processor w. For example, (2,1,2) means that first operation of job 2 requires processor 2.	
	Earliest start time of (1,1,3), (2,1,2), and (3,1,2) are $E_{113} = E_{212} = E_{312} = 0$ $E^* = \text{Min}\ \{E_j\} = \{0, 0, 0\} = 0$	
	We select MWKR priority rule and compute the remaining times for job j, $j \in S_0$ $R_1 = 4 + 3 + 4 = 11$ $R_2 = 4 + 4 + 2 = 10$ $R_3 = 2 + 5 + 5 = 12$ We compute $R^* = \text{Max}\ \{R_2, R_3, R_4\} = R_3 = 12$ PS_1 = {(3,1,2)} starts at time 0.	
	We repeat the above procedure for other PS_t. Then we obtain the intermediate schedule in Table-4.	
2	For CPT 6 For CPT 7 For CPT 8	we set nc_{ip} = 1, 2 and $level_6 = 1$ nc_{ip} = 3, 6 and $level_7 = 2$ nc_{ip} = 4, 7 and $level_8 = 3$
	For level$_6$ = 1	S_0 = {(6,1,5)} $E_{615} = \text{Max}\ \{f_{6,1}, f_{6,2}\}$ $= \text{Max}\{14, 16\} = 16$
	For level$_7$ = 2	S_0 = {(7,1,6)} $E_{616} = \text{Max}\ \{f_{7,3}, f_{6,5}\}$ $= \text{Max}\{15, 19\}$ $= 19$ PS_{19} = {(7,1,6)} starts at time 19. S_0 = {(7,2,5)} $E_{725} = \text{Max}\ \{C_{716}, f_{73}\}$ $= \text{Max}\{21, 15\}$ $= 21$ PS_{21} = {(7,2,5)} starts at time 21.
	For level$_8$ = 3	S_0 = {(8,1,5)} $E_{615} = \text{Max}\ \{C_{725}, f_{8,4}\}$ $= \text{Max}\{25, 19\} = 25$
3	We assign job 5 with single processor to processor 2.	

Table-4 shows the intermediate schedule for the example described in Table-1 and Table-2. E_j and C_j represent the earliest start and finish times of job (j), respectively.

Table-4. Intermediate schedule for the example

Job (*j*)	E_j, C_j	E_j, C_j	E_j, C_j
1	0, 4 processor 3	6, 9 processor 2	10, 14 processor 1
2	2, 6 processor 2	6, 10 processor 3	14, 16 processor 1
3	0, 2 processor 2	2, 7 Processor 1	10, 15 processor 3

We obtain the final schedule for the example which is shown in Table-5.

Table-5. Final schedule for the example.

Job (*j*)	E_j, C_j	E_j, C_j	E_j, C_j
1	0, 4 processor 3	6, 9 processor 2	10, 14 processor 1
2	2, 6 processor 2	6, 10 processor 3	14, 16 processor 1
3	0, 2 processor 2	2, 7 Processor 1	10, 15 processor 3
4	9, 11 processor 2	16, 19 processor 1	
5	11, 17 processor 2		
CPT 6	16, 19 processor 5		
CPT 7	19, 21 processor 6	21, 25 processor 5	
CPT 8	25, 28 processor 5		

Figure-3 depicts the flow diagram of the final schedule for the example.

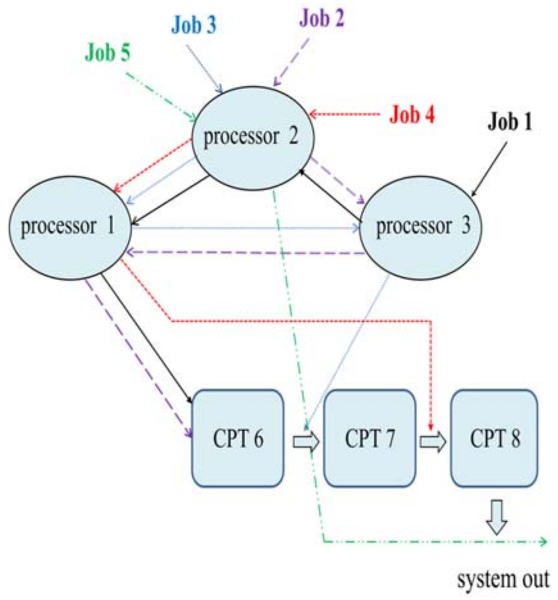

Figure-3. Flow diagram of the final schedule for example.

4. CONCLUSIONS

This paper proposes a scheduling algorithm for the system composed the hierarchical combined jobs which has jobshop characteristics. In order to minimize the makespan and handle the hierarchical jobs, we modified and extend the previous algorithm based on the most work remaining heuristics and shortest processing time heuristic.

Although we obtain good solutions through some numerical examples, our approach is heuristic, so we cannot guarantee the optimal solution. This shortcoming can be solved by combining heuristic approach and simulation.

In future studies, it is necessary to evaluate the effectiveness of the proposed algorithm by comparison with other algorithms and to consider the job with the stochastic operation time.

REFERENCES

[1] Pinedo M. 2015. Scheduling, Theory, Algorithms, and Systems. Springer.

[2] Baker K. 1985. Introduction to Sequencing and Scheduling. pp. 196-200.

[3] Lee Y. J. 2017. Integrated information and communication learning model for Raspberri Pi environment. ARPN Journal of Engineering and Applied Science. 12(17): 5088-5093.

[4] Lee Y. -J. 2017. Lower bound for mean object transfer latency in the narrowband IoT environment. International Journal of Applied Engineering Research. 12(12): 3365-3369.

[5] Lee Y. -J. 2015. Novel quality of service measure for web transactions in multiple user access environment. International Journal of Applied Engineering Research. 10(16): 37439-37444.

[6] Al-Hinai and Piya S. 2015. Jobshop Scheduling for skill-dependent Make-to-order system. 5th International Conference on Industrial Engineering and Operations.

[7] Zheng Y., Qu J. and Wang L. 2015. An improved cooperative PSO algorithm for job-shop scheduling problem. Lecture Notes in Computer Science. 8944: 265-277.

[8] Mahdavinejad R.A. 2011. Job shop-scheduling problems- Single process. Applied Mechanics and Materials. 44-47: 330-334.

[9] Lee Y. 1986. A Scheduling policy for the system composed of hierarchical jobs having jobshop

characteristics. Daejon and Jungkyong Junior College Review. 15: 165-181.

[10] Azadeh A., Ghaderi S.F., Dehghanbaghi M. and Dabbaghi A. 2010. Integration of simulation, design of experiment and goal programming for minimization of makespan and tardiness. International Journal of Advanced Manufacturing Technology. 46(5-8): 431-444.

[11] Masin M. and Ravit T. 2014. Liner programming-based algorithms for the minimum makespan high multiplicity jobshop problem. Journal of Scheduling. 17(4): 321-338.

[12] Udaiyakumar K. C. and Chandrasekaran M. 2014. Application of firefly algorithm in jobshop scheduling problem for minimization of makespan. Procedia Engineering. 97: 1798-1807.

[13] Farashahi H. G., Baharudin B., Shojaeipour S. and Jaberi M. 2011. Efficient genetic algorithm for flexible job-shop scheduling problem using minimize makespan. Communications in Computer and Information Science. 135: 385-392.

[14] Lee Y. -J. 2016. Location management agent for SCTP handover in mobile network. International Journal of Applied Engineering Research. 11(11): 7532-7536.

[15] Lee Y. -J. 2016. Some considerations for SCTP handover scheme in mobile network. International Journal of Applied Engineering Research. 11(11): 7526-7531.