COMPARISON BETWEEN MIXED INTEGER PROGRAMMING WITH HEURISTIC METHOD FOR JOB SHOP SCHEDULING WITH SEPARABLE SEQUENCE-DEPENDENT SETUPS

I Gede Agus Widyadana
Lecturer at the Department of Industrial Engineering, Faculty of Industrial Technology
Petra Christian University

ABSTRACT

The decisions to choose appropriate tools for solving industrial problems are not just tools that achieve optimal solution only but it should consider computation time too. One of industrial problems that still difficult to achieve both criteria is scheduling problem. This paper discuss comparison between mixed integer programming which result optimal solution and heuristic method to solve job shop scheduling problem with separable sequence-dependent setup. The problems are generated and the result shows that the heuristic methods still cannot satisfy optimal solution.

Keywords: job shop, sequence dependent setup, Mixed Integer Programming, heuristic.

1. INTRODUCTION

Today many researches are focused in scheduling problem because this problems are occur practically and still difficult to find some methods that are satisfy computation time and optimal solution.

The job shop scheduling problem can be described as the problem of assigning a set of jobs with predetermined machine operation sequences to a set of machines so as to minimize the time required to complete all jobs in the systems known as makespan. The sequence dependent setups time means that setup of particular job in the particular machine are depend on job that have been processed before.

The job shop scheduling problem with sequence dependent setups appears frequently in the real life situations, but mainly because its complexity, it has received little attention from researchers until recently.

Studies of scheduling problems involving sequence dependent setup have mostly focused on single and parallel machine cases. For the job shop scheduling problem with sequence dependent setups, Zhou and Egbelu presents a heuristic procedure for minimizing the makespan and Ovacik and Uzsoy describe a heuristic, based on the shifting bottleneck procedure, for minimizing the maximum lateness jobs.

In their paper, Choi and Korkmaz built a new heuristic method for job shop scheduling with separable sequence dependent setups and then compare their heuristic method with Zhou and Egbelu method and Earliest-Start-Time-First with shortest-processing-time tie breaking rule (EST/SPT) by generated some problems.

The result of their comparison shows that their heuristic method is the most powerful in general. But they do not compare their heuristic method with mixed integer programming formulation that guarantees best solution.

2. METHODS

There are two methods that are used to solve Job Shop Scheduling with Separable Sequence-Dependent Setups, and then they are compared according good solution and computation time.

2.1 Mixed Integer Programming Formulation

The job shop-scheduling problem we consider has the following usual assumptions. There are m machines and n jobs in the system. Each job j has a fixed machine (operation) sequence $sj = (s_1^j, s_2^j, \ldots, s_m^j)$, where s_i^j represents the i_{th} machine that the job j is proceed on. Each machine can process one job at a time and each job can be processed by only one machine at a time. Preemption of jobs is not allowed and a separable, sequence-dependent setup is necessary for each job before it can be processed on each machine.

The following notation is used throughout the paper. Unless stated otherwise, indices j and k will denoted jobs and except for the case when used as subscript of s, indices i and h will be used for denoting machines. Also, in this paper, we will distinguish an operation from a job. A job consists of series of operation and an operation (O_{ij}) is defined as the processing of job j on a particular machine i.

n = number of jobs in the system;
m = number of machines in the system;
A = set of ordered pairs of jobs, $\{(j,k) | j = 1,\ldots,n, k = 1,\ldots,n, j \neq k\}$;
s_j = machine operation sequence of job j, $s_j = (s_1^j,\ldots, s_m^j)$ $j = 1,\ldots,n$;
O_{ij} = operation corresponding to job j on machine i, $i=1..,m, j = 1..,n$;
p_{ij} = processing time of operation O_{ij}, $i=1..,m, j = 1..,n$;
S^i_{jk} = required setup time for job k on machine i, O_{ik}, when job j on machine i, O_{ij}, is processed immediately before O_{ik}, $j = 1,\ldots,n, k = 1,\ldots,n, j \neq k$;
t_i = completion time of latest scheduled operation on machine i, $i = 1,\ldots m$
q_j = lower bound estimate of the remaining time that job j has to stay in the system, $j = 1,\ldots,n$;
V_i = set of jobs available for processing on machine i at or after t_i, $i = 1,\ldots,m$;
V = union of Vi;
C_{max} = makespan of a schedule;
t_{ij} = scheduled completion time of operation O_{ij} $(\equiv CT(O_{ij}))$, $i=1..,m, j = 1..,n$;

With the above notations, Job Shop Scheduling with Separable Sequence-Dependent Setups can be formulated as mixed integer programming problem as follows. Without loss of generality, it is assumed that dummy job with zero processing time and zero sequence-dependent setup time is added on each machine, i.e. $S^i_{0j} = 0$, for $j = 1,\ldots n$, and $p_{i0} = 0$ for $i = 1,\ldots,m$. Let

$$x_{ijk} = \begin{cases} 1, & \text{if job k immediately follows job j on machine i,} \\ & \text{or equivalently, } O_{ik} \text{ immediately follows } O_{ij}, \\ 0, & \text{Otherwise.} \end{cases}$$

Then:

Minimize C_{max} (1)

Subject to

$t_{ó_h^j j} + p_{ó_h^j j} \le t_{ó_h^j j}$ $h = 1,\ldots,m\text{-}1,\ j = 1,\ldots,n;$ (2)

$t_{s_m^j j} + p_{s_m^j j} \le C_{\max}$ $j = 1,\ldots,n;$ (3)

$t_{ij} + p_{ij} + S_{jk}^{i} x_{ijk} \le t_{ik} + (1 - x_{ijk})M$ $i = 1,\ldots,m,\ j = 0,1,\ldots,n,$ (4)

$\sum_{k \ne j}^{n} x_{ijk} = 1$ $i = 1,\ldots,m,\ j = 0,1,\ldots,n;$ (5)

$\sum_{j \ne k}^{n} x_{ijk} = 1$ $i = 1,\ldots,m,\ k = 0,1,\ldots,n;$ (6)

$\sum_{\substack{(j,k) \in A \\ j \in Ui;\, k \notin Ui}} x_{ijk} \le |U_i| - 1$ $2 \le |U_i| \le (n+1) - 1;\ i = 1,..m,$ (7)

$x_{ijk} \in \{0,1\}, t_{ij} \ge 0.$ (8)

Constraint (2) and (3), which also appear in a regular job shop scheduling problem formulation, enforce the specified machine operation sequence of each job. Constraint (4) prevents operations within each machine from overlapping. With this constraint, job k immediately following job j on machine i can start only after both the processing of job j and the setup for job k are completed. This constraint becomes redundant if there is no adjacency between jobs j and k. Observe that job index k starts form 1 rather than 0. By ignoring the case for $k = 0$, constraint (4) associated with the last scheduled job an each machine and the dummy job is nullified.

Constraint (5) – (8) define the circular permutations of operation on each machine i. Constraint (5) picks exactly one job immediately following job j on machine i and constraint (6) picks exactly one job immediately preceding job k on machine i. Constraint (7), which is equivalent to the subtour elimination constraint in traveling salesman problem, guarantees the continuity of the adjacency between jobs in each machine, i.e. it prohibits the partial grouping of jobs in each machine. It should be mentioned here that constrained (7) is redundant with respect to constrain (4) in the sense that its removal from the formulation does not affect the solutions of above mixed integer program. However, the inclusion of constraint (7) in the formulation provides stronger mixed integer programming formulation for equivalent related to traveling salesman problem.

2.2 Heuristic Procedure

Three definitions for the description of the heuristic are:

1. $EST(O_{ij})$ is defined as earliest possible process start time of job j on machine i. $i = 1,\ldots,n,\ j = 1,\ldots,m.$ Let us assume without loss of generality that the h_{th} operation of job j, $Os_h^j j$ is performed on machine i, $i = s_h^j$. Then $EST(O_{ij})$ is computed as:

$$EST(O_{ij}) \equiv \begin{cases} max\{CT(O_{h-i}^{j}j), t_i + S_{fj}^{i}\} & \text{if } O\sigma_{h-1}^{j} \text{ is scheduled} \\ max\{EST(O_{h-1}^{j}j) + p_{h-1}^{j}j, t_i + S_{fj}^{i}\} & \text{if } O\sigma_{h-1}^{j} \text{ is not scheduled} \end{cases} \quad (9)$$

where *f*, finishing its operation O_{if} at t_i, is the last scheduled job on machine *i*. It should be note that $EST(O_{i,j})$ includes both setup time and idle time and that it is dynamically updated as t_i changes.

2. The tail of job j, denoted by q_j, represents a lower bound estimate of the remaining time that the job has to stay in the system. Ideally, this estimate would include lower bounds of the forced idle times and setup times. In this algorithm, q_j is estimated by a simple difference between the lower bound estimate of the completion time of job j, $EST(s_m^j j) + p s_m^j j$, and a projected completion time of operation O_{ij}. Thus if job j is available for processing on machine *i*, then its tail is computed as

$$q_j = EST(s_m^j j) + p s_m^j j - (EST(O_{ij}) + p_{ij}) \quad (10)$$

Initially, the tail of job j is computed as $q_j = \sum_{i=1}^{m} p_{ij}$.

3. Make use of in this heuristic is the minimum local makespan of a pair of operations, which is defined as the minimum makespan obtained by solving a two-job/*m* machine problem with release time, where the two jobs are those associated with the pair of operations under consideration. Instead of finding the minimum local makespan of a pair of operations, they use the lower bound on it. For two operations (jobs) on the same machine, the lower bound on the minimum local makespan is computed as, for j, $k \in V_i$ and $j \neq k$,

$$LB(O_{ij}, O_{ik}) \equiv \max \begin{Bmatrix} EST(O_{ij}) + p_{ij} + q_j, \\ EST(O_{ik}) + p_{ik} + q_k, \\ EST(O_{ij}) + p_{ij} + q_j + S_{jk}^{i} + p_{ik} + q_{k,} \end{Bmatrix} \quad (11)$$

then the lower bound on the completion time of job j at the state of system $(t_1,\ldots,t_m)$, $j \in V_i$, $i = 1,\ldots,m$, can be obtained as

$$LB(O_{ij}) = \min_{k \in Vi} \{LB(O_{ij}, O_{ik})\} \quad (12)$$

The lower bound in equation (3) represents an optimistic completion time of job j if operation O_{ij} is followed by operation O_{ik}. The lower bound in equation (4) is the best optimistic completion time of job j at a given state of the system $(t_1,\ldots,t_m)$.

In each iteration of algorithm below, they choose an operation from two candidate operations: one with the earliest start time and the other with minimum $LB(O_{ij})$ at a given state of the system, $(t_1,\ldots,t_m)$.

Algorithm SDJSS

Inputs: $m, n, s_j, p_{ij}, S_{jk}^{i}$

Outputs: $CT(O_{ij})$ $(=t_{ij})$

STEP 1. $t_1 = 0$, $EST(O_{ij}) = 0$, and $CT(O_{ij}) = 0$ for $i = 1,\ldots,m$, $j = 1,\ldots n$.

STEP 2. For all O_{ij} not yet scheduled, compute $EST(O_{ij})$ using equation (9).

For all $j \in V_i$, $I = 1,\ldots,m$, compute q_I using equation (10).

STEP 3. Let $Q_1 = \{(i,j) | EST(O_{ij}) \leq EST(O_{hk}),\ j \in V_i,\ k \in V_h,\ i, h = 1,\ldots,m\}$,
$R_1 = \{(i,j) \in Q_1 \mid p_{ij} \geq p_{hk}, (h,k) \in Q_1\}$.
Choose an operation Oi_1j_1 such that $(i_1,j_1) \in R_1$.

STEP 4. Compute $LB(O_{ij})$ using equation (12), for $j \in V_i$ and $i = 1,\ldots,m$.
Let $P = \{(i,j) \mid LB(O_{ij}) \leq LB(O_{hk}),\ j \in V_i,\ k \in V_h,\ i, h = 1,\ldots,m\}$,
$Q_2 = \{(i,j) \in P \mid EST(O_{ij}) \leq EST(O_{hk}), (h,k) \in P\}$,
$R_2 = \{(i,j) \in Q_2 \mid p_{ij} \geq p_{hk}, (h,k) \in Q_2\}$.
Choose an operation Oi_2j_2 such that $(i_2,j_2) \in R_2$.

STEP 5. If $i_1 = i_2$, then $i^* \leftarrow i_2$, $j^* \leftarrow j_2$; Else (if $i_1 \neq i_2$), $i^* \leftarrow i_1$, $j^* \leftarrow j_1$

STEP 6. $t_{i^*} \leftarrow EST(O_{i^* j^*}) + p_{i^* j^*}$;

$CT(O_{i^* j^*}) \leftarrow t_{i^*}$;

$V_{i^*} \leftarrow V_{i^*} - \{j^*\}, V_{\bar{i}^*} \leftarrow V_{\bar{i}^*} \cup \{j^*\}$, where $\bar{i}^*$ is the index of machine on which the job j^* is processed immediately after machine i^*.
If unscheduled operations remain, go to step 2.

3. COMPARISON

This paper only compare two problem due to time restrictions, both of the problems have 3 machines and 5 jobs with same operation.

3.1 First Problem

The first problem be generated with randomize processing time and setup times have a lower limit of 1 and upper limit of 5. Detail of the problem is:

Table 1. Processing Time

Job (j)	**1**	**2**	**3**	**4**	**5**
Machine seq. (s j)	3 → 2 → 1	1 → 3 → 2	3 → 2 → 1	1 → 2 → 3	3 → 1 → 2
Proc. Time (pij)	5 5 4	5 1 2	4 2 4	4 2 5	4 4 5

Setup Times (S^i_{jk})

Table 2. Machine 1

	k					
j	***0***	***1***	***2***	***3***	***4***	***5***
0	0	3	1	1	3	1
1	1	0	2	4	3	3
2	3	4	0	2	4	4
3	3	4	1	0	1	3
4	3	1	2	3	0	4
5	3	1	3	2	2	0

Table 3. Machine 2

j	k					
	0	***1***	***2***	***3***	***4***	***5***
0	0	4	2	4	4	1
1	3	0	1	3	4	4
2	3	2	0	3	3	1
3	4	3	4	0	4	4
4	4	2	3	1	0	3
5	3	2	4	1	1	0

Table 4. Machine 3

j	k					
	0	***1***	***2***	***3***	***4***	***5***
0	0	1	3	1	2	2
1	3	0	4	4	1	4
2	1	3	0	1	4	2
3	2	3	3	0	2	4
4	1	2	3	2	0	2
5	4	2	1	4	3	0

The optimal solution is calculated using softare LINGO and run in computer with Pentium II/400 specification. The result is:

Makespan : 33
No. of iterations : 575515
Elapsed time : 13 minutes, 39 seconds

Table 5. Schedule

Machine	Job Sequences
1	$4 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 1$
2	$1 \rightarrow 3 \rightarrow 4 \rightarrow 2 \rightarrow 5$
3	$1 \rightarrow 3 \rightarrow 2 \rightarrow 5 \rightarrow 4$

Machine	Job	Start setup time	Finish proc time
1	1	26	33
1	2	7	14
1	3	14	20
1	4	0	7
1	5	21	28
2	1	2	11
2	2	22	27
2	3	11	16

Machine	Job	Start setup time	Finish proc time
2	4	16	22
2	5	27	33
3	1	0	6
3	2	14	18
3	3	6	14
3	4	24	32
3	5	18	24

The makespan of heuristic methods are 34 with sequences:

Table 6. Job Sequences

Machine	Job Sequences
1	$2 \rightarrow 4 \rightarrow 1 \rightarrow 5 \rightarrow 3$
2	$1 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 5$
3	$1 \rightarrow 3 \rightarrow 2 \rightarrow 5 \rightarrow 4$

The first problem show that the result from mixed integer programming is better one unit time than heuristic method.

3.2 Second Problem

In the second problem, setup time is greater than processing time. This situation is common in real condition for this problem; so the second problem is generated randomly using spreadsheet with lower limit of 1 and an upper limit of 5 for processing time and lower limit of 11 and upper limit of 31 for setup time.

Table 7. The sequence dan processing time

Job (j)	**1**	**2**	**3**	**4**	**5**
Machine seq. (s j)	$3 \rightarrow 2 \rightarrow 1$	$1 \rightarrow 3 \rightarrow 2$	$3 \rightarrow 2 \rightarrow 1$	$1 \rightarrow 2 \rightarrow 3$	$3 \rightarrow 1 \rightarrow 2$
Proc. Time (pij)	4 4 2	2 3 3	2 5 1	2 3 4	2 3 2

Setup Times (S^i_{jk})

Table 8. Machine 1

	k					
j	***0***	***1***	***2***	***3***	***4***	***5***
0	0	26	16	26	19	24
1	18	0	13	25	28	20
2	20	30	0	24	27	22
3	25	24	27	0	24	30
4	21	22	29	15	0	30
5	29	14	24	26	14	0

Table 9. Machine 2

	k					
j	***0***	***1***	***2***	***3***	***4***	***5***
0	0	27	13	16	22	19
1	20	0	17	15	18	19
2	20	28	0	23	29	29
3	25	25	31	0	25	16
4	24	19	22	20	0	17
5	17	24	30	13	16	0

Table 10. Machine 3

j	k					
	0	***1***	***2***	***3***	***4***	***5***
0	0	14	18	22	23	30
1	11	0	30	23	19	12
2	24	21	0	12	20	27
3	12	18	23	0	11	22
4	19	12	25	17	0	17
5	27	21	15	26	19	0

The optimal solution from mixed integer programming model is:
Makespan : 115
No. of iterations : 896070
Elapsed time : 19 minutes, 54 seconds

Table 11. Schedule

Machine	Job Sequences
1	$2 \rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 1$
2	$3 \rightarrow 5 \rightarrow 4 \rightarrow 1 \rightarrow 2$
3	$3 \rightarrow 5 \rightarrow 2 \rightarrow 1 \rightarrow 4$

Machine	Job	Start setup time	Finish proc time
1	1	83	109
1	2	0	18
1	3	67	83
1	4	51	76
1	5	26	51
2	1	67	95
2	2	95	115
2	3	8	29

Machine	Job	Start setup time	Finish proc time
2	4	53	71
2	5	35	53
3	1	66	91
3	2	48	66
3	3	0	24
3	4	91	114
3	5	24	48

The makespan of heuristic methods are 125 with sequences:

Table 12. Job Sequences

Machine	Job Sequences
1	$4 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 5$
2	$1 \rightarrow 3 \rightarrow 2 \rightarrow 4 \rightarrow 5$
3	$1 \rightarrow 3 \rightarrow 2 \rightarrow 4 \rightarrow 5$

It can be seen from the result that makespan from heuristic method are 125, so mixed integer programming in this case are better 10 unit times or 8% than heuristic method.

4. CONCLUSION

Calculation result from two case show that mixed integer programming give better result than heuristic method which be build by Choi & Korkmaz although their method are better than other heuristic method that showed in their paper.

It can be seen that with same problem size and operation, but with wider deviation and greater setup time, performance of heuristic method become worst compare with mixed integer-programming method.

Although mixed integer programming method will give optimal result for job shop scheduling with separable sequence-dependent setups, but computation time increase fast when problem size are bigger, and there is possibility that software can't solve the problem because of limitation capacity.

The result of two cases is not enough to make satisfy conclusion, so it is better to make more cases by generate it with variation in problem size, operation route, set up and operation time.

Because performances of heuristic methods in this problem have big possibility far from optimal solution, so there is area for research to find another heuristic method, which can give better performance. Research in meta-heuristic method like genetic algorithm, simulated annealing and tabu search maybe can give better result.

REFERENCES

Murty Katta G., 1995. *Operations Research Deterministic Optimization Models*, Prentice Hall: Upper Saddle River, New Jersey.

Pinendo Michael, 1995. *Scheduling Theory, Adgorithms and System*, Prentice hall: Eaglewood Cliffs New Jersey.

LINGO, 1999. *User's guide*, Lindo Systems Inc.