Songklanakarin J. Sci. Technol.
40 (5), 1203-1218, Sep. - Oct. 2018

http://www.sjst.psu.ac.th

Original Article

A memetic algorithm to minimize the total sum of earliness tardiness and sequence dependent setup costs for flow shop scheduling problems with job distinct due windows

Anot Chaimanee* and Wisut Supithak

Department of Industrial Engineering, Faculty of Engineering, Kasetsart University, Chatuchak, Bangkok, 10900 Thailand

Received: 21 December 2016; Revised: 22 June 2017; Accepted: 16 July 2017

Abstract

The research considers the flow shop scheduling problem under the Just-In-Time (JIT) philosophy. There are *n* jobs waiting to be processed through *m* operations of a flow shop production system. The objective is to determine the job schedule such that the total cost consisting of setup, earliness, and tardiness costs, is minimized. To represent the problem, the Integer Linear Programming (ILP) mathematical model is created. A Memetic Algorithm (MA) is developed to determine the proper solution. The evolutionary procedure, worked as the global search, is applied to seek for the good job sequences. In order to conduct the local search, an optimal timing algorithm is developed and inserted in the procedure to determine the best schedule of each job sequence. From the numerical experiment of 360 problems, the proposed MA can provide optimal solutions for 355 problems. It is obvious that the MA can provide the good solution in a reasonable amount of time.

Keywords: flow shop scheduling, earliness tardiness, due window, optimal timing algorithm, Memetic Algorithm

1. Introduction

In the past several years, many researchers have conducted research in the production scheduling under the Just-In-Time (JIT) philosophy. The objective of the JIT is to produce and deliver products not before or after their committed due dates. Any jobs completed early must be held by manufacturer until their due dates and, hence, incur some costs as a result of product deterioration, storage, and insurance. On the other hand, those jobs completed after their due dates can cause many problems such as customer penalties, loss of sales, or potential loss of reputation. In accordance with Baker and Scudder (1990), an ideal schedule is the one in which all jobs are finished exactly on their due dates. The most obvious objective of scheduling problem under the JIT policy is to minimize the deviations of job completion times around their due dates. It can be seen as the minimization problem of total sum of earliness and tardiness penalties (E/T scheduling problem). The E/T scheduling problem can be divided into several categories according to the types of machine system, due date, and characteristics of weight penalties.

According to Pinedo (2002), the E/T scheduling problem of jobs having different due dates in a single machine production system is NP hard. Lee and Kim (1995) studied the job scheduling problem on single machine with common due date. The objective was to minimize the total generally weighted of earliness and tardiness penalties. The similar problem but with distinct due dates was discussed by Lee and Choi (1995). Szwarc and Mukhopadhyay (1995) proposed an optimal timing to find an optimal timing job starting position for the E/T problem with the predetermined job sequence. Sarper (1995) proposed the minimization problem of the sum of absolute deviations of job completion times around a common due date for the two machines flow shop. Moslehi *et*

*Corresponding author
Email address: anotchaimanee@hotmail.com

al. (2009) studied similar problem but distinct due dates. The author addressed the case of minimizing the sum of maximum earliness and tardiness. Yoon and Ventura (2002) proposed a procedure for minimizing the mean weighted absolute deviation of job completion times around their due dates when jobs are scheduled in a lot-streaming flow shop. Sufen *et al*. (2005) discussed the E/T scheduling problem of *n* jobs *m* machines flow shop with uncertainty of job processing times. Chandra *et al*. (2009) considered the E/T problem under a common due date in the flow shop production system. The similar problem but with distinct due date was discussed by Schaller and Valente (2013) and M'Hallah (2014). According to the survey research on E/T scheduling problems with job due window conducted by Janiak *et al*. (2015), both common and distinct due window problems are NP-hard. While Yeung *et al.* (2004) discussed the E/T scheduling problem with common due window, the cases of problem with distinct due windows were studied by Behnamian *et al*. (2009), Koulamas (1996), and Wan and Yen (2002). Nonetheless, as being known so far, there is no research conducted relevant to the E/T scheduling problem with flow shop machine system and job distinct due windows.

In recent-years, a growing number of literatures suggest the application of Genetic Algorithm (GA) as one of those powerful metaheuristic being used to solve combinatorial optimization problems (Cheng *el al*., 1995, Reeves, 1995; Sevaux & Dauzere-Peres, 2003). According to Sevaux and Dauzere-Peres (2003), the main difference between the GA and other metaheuristics such as Tabu Search (TS) or Simulated Annealing (SA) is that not only GA maintains the population of solution rather than unique current solution but it allows the exploration of a larger solution space as well. Because of those benefits discussed previously and the simplicity to represent each job as a gene of a chromosome in the solution representation, the GA has been applied by many researchers to seek for the good solution in the job sequencing problems. Some of them are Lee and Choi (1995), Lee and Kim (1995), Murata *et al*. (1996), Reeves (1995), and Sufen *et al*. (2005). The Memetic Algorithm (MA) can be considered as the extension model of general GA. According to Tavakkoli-Moghaddam *et al*. (2009), unlike traditional GA, the Memetic Algorithm is population-based search approach combining evolutionary procedure with local refinement strategies such as local neighborhood search.

This paper considers the E/T scheduling problem of jobs having distinct due windows in *m*-operation flow shop production system. The mathematical model is introduced to represent the problem. In order to determine the starting time of each job when the job sequence is known, the optimal timing algorithm for the E/T flow shop scheduling system is created. The Memetic Algorithm based on evolutionary procedure with the insertion of optimal timing algorithm is presented to determine the good solution to the problem.

2. Problem Characteristics

There are *n* jobs waiting to be process on *m* machines flow shop production system. Each job has its own earliness penalty, tardiness penalty, earliest due date, and latest due date. Any jobs completed before their earliest due dates incur earliness penalties. On the other hand, those jobs completed after their latest due dates incur tardiness penalties. All jobs are assumed to be available for the production at the beginning of planning horizon. The objective is to determine production schedule such that the sum of earliness and tardiness costs of all jobs are minimized. The following notations are used throughout the paper.

n = number of jobs

m = number of machines

i, j = index of jobs ; $i, j = 0,1,2,\dots,n$

$[i],[j]$ = index of job positions in given sequence ; $i, j = 1,2,\dots,n$

k = index of machines ; $k = 1,2,\dots,m$

g = index of sub-schedules ; $g = 1,2,\dots,G$

r = index of job clusters ; $r = 1,2,\dots,R$

$C_{i,k}$ = completion time of job i on machine k

W_i = length of due window of job i ; $W_i = t_i - e_i$

e_i = earliest due date of job i

t_i = latest due date of job i

E_i = earliness of job i ; $E_i = \max(e_i - C_{i,m}, 0)$

T_i = tardiness of job i ; $T_i = \max(C_{i,m} - t_i, 0)$

OT_{i_1} = the time period of job i when that job completed after its earliest due date ; $OT_{i_1} = \max(C_{i,m} - e_i, 0)$

OT_{i_2} = the time period of job i when that job completed before its latest due date ; $OT_{i_2} = \max(t_i - C_{i,m}, 0)$

α_i = earliness cost of job i

β_i = tardiness cost of job i

$\gamma_{i,j}$ = setup cost of job j when it immediately processed after job i. Here, $\gamma_{0,j}$ refers to the setup cost of job j when it is the first job being process on a machine.

$P_{i,k}$ = processing time of job i on machine k

TC = total cost

J_i = job i of all n jobs

J_0 = a dummy job having processing time on all machines equal to zero

$J_{[i]}$ = job in i^{th} position of the given sequence

J_g^{first} = the first job in sub-schedule g

J_g^{last} = the last job in sub-schedule g

σ_r = cluster r

$J_{i,r}$ = job i in cluster r ; $i = F, F+1,\dots, E, E+1,\dots, W, W+1,\dots,T, T+1,\dots,L$

$J_{F,r}$ = the first job in cluster r

$J_{E,r}$ = the last early job in cluster r

$J_{W,r}$ = the last job is completed in due window and condition $t_W - C_{W,r} > 0$ is hold

$J_{T,r}$ = the first tardy job in cluster r

$J_{L,r}$ = the last job in cluster r

$$x_{i,j} = \begin{cases} 1 \text{ if job } i \text{ precedes job } j \\ 0 \text{ otherwise} \end{cases}$$

y_1 = the shift distance being calculated from early jobs in cluster r; $y_1 = \min_{F \le i \le E} \{e_i - C_{i,r}\}$

y_2 = the shift distance being calculated from jobs completed in due window with the condition $t_i - C_{i,r} > 0$ for $i = E+1,\dots,W$ are hold; $y_2 = \min_{E+1 \le i \le W} \{t_i - C_{i,r}\}$

S_{G+1}^{first} = starting time of the job in sub-schedule G+1; $S_{G+1}^{first} = \infty$

C_G^{last} = completion time of the last job in sub-schedule G; $S_{G+1}^{first} - C_G^{last} = \infty$

R_g = the last cluster on sub-schedule g

M = large number

The following example demonstrates the problem characteristic.

Example 1:

Given that five jobs are waiting to be processed through three operations of a flow shop production system. The data of each job are presented in the Table 1.

Table 1. Data for Example 1.

J_i	Setup cost in dollars/setup ($\gamma_{i,j}$)					Earliness cost in dollars/day (α_i)	Tardiness cost in dollars/day (β_i)	Processing times (days)			Due windows	
	j=1	j=2	j=3	j=4	j=5			$P_{i,1}$	$P_{i,2}$	$P_{i,3}$	e_i	t_i
i=0	2	2	3	5	2	0	0	0	0	0	0	0
i=1	0	1	3	4	4	2	4	10	8	5	29	31
i=2	2	0	1	3	3	3	6	9	7	8	43	46
i=3	1	2	0	2	2	1	3	7	5	10	32	34
i=4	3	1	2	0	5	2	4	10	9	10	59	61
i=5	1	2	1	3	0	2	5	6	9	10	55	58

The Gantt chart in the Figure 1 represents one possible production schedule of the problem (this schedule may not be the optimal). Here, the research assumes that all jobs are processed with the same sequence on all machines.

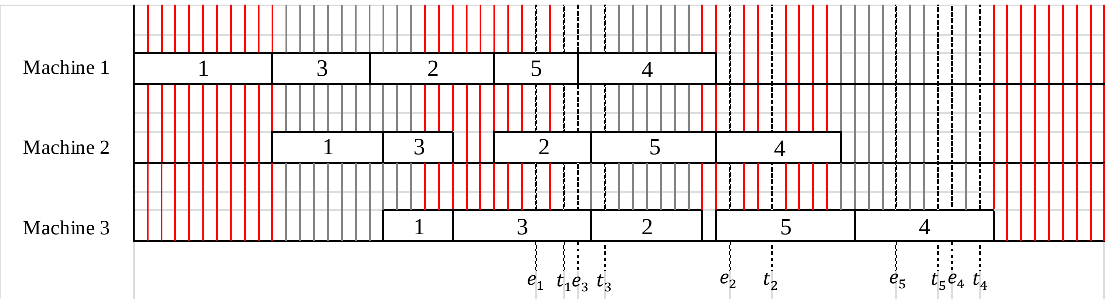

Figure 1. One possible production schedule.

The cost associated with schedule in Figure 1 can be calculated as follows:

Setup Cost: The total setup cost incurred from the sequence can be calculated as $\gamma_{0,1}+\gamma_{1,3}+\gamma_{3,2}+\gamma_{2,5}+\gamma_{5,4}$ = 2+3+2+3+3 = \$13.

Earliness cost: Job 1, 2, and 5 are early for 6, 2, and 3 days, respectively. The earliness cost can be calculated as $\alpha_1 E_1+\alpha_2 E_2+\alpha_5 E_5$ = 2(6)+3(2)+2(3) = \$24

Tardiness cost: Only job 4 is tardy. The tardy cost is $\beta_4 T_4$ = 4(1) = \$4.

Note that job 3 has no penalty since it is completed within the due window.

The total cost; TC = 13+24+4 = \$41.

2.1 Integer linear programming

The mathematical model based on Integer Linear Programming (ILP) was developed and can be shown as follows:

Objective Function

$$minimize\ TC=\sum_{i=1}^{n}\left(\alpha_i E_i+\beta_i T_i\right)+\sum_{i=0}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\gamma_{i,j}x_{i,j}$$

Constraints

$$\sum_{j=1}^{n} x_{0,j} = 1 \tag{1}$$

$$\sum_{\substack{i=0 \\ i\neq j}}^{n} x_{i,j} = 1 \qquad ;j = 1, 2,...,n \tag{2}$$

$$\sum_{\substack{j=1 \\ j\neq i}}^{n} x_{i,j} \leq 1 \qquad ;i = 1, 2,..., n \tag{3}$$

$$C_{i,1} \geq P_{i,1} \qquad ;i = 1, 2,...,n \tag{4}$$

$$C_{j,k} - C_{i,k} + M\left(1 - x_{i,j}\right) \geq P_{j,k} \qquad ;i,j(i \neq j) = 1,...,n;\ k = 1,...,m \tag{5}$$

$$C_{i,k} - C_{i,k-1} \geq P_{i,k} \qquad ;i = 1, 2,...,n;\ k = 2,...,m \tag{6}$$

$$C_{i,m} + E_i - OT_{i_1} = e_i \qquad ;i = 1, 2,...,n \tag{7}$$

$$C_{i,m} - T_i + OT_{i_2} = t_i \qquad ;i = 1, 2,...,n \tag{8}$$

The objective function represents the sum of total cost including setup, earliness and tardiness costs of all jobs. Constraint (1) identifies the first job of the sequence. Constraint (2) ensures that each job can have at most one immediately preceding job. According to constraint (3), there is at most one job can be immediately processed after job *i*. Constraint (4) guarantees that the first job being process on the first machine cannot start before the time of zero. Constraint (5) ensures that any two adjacent jobs on the same machine are processed continuously without overlapping. Constraint (6) requires that any operations of a job cannot be overlapped. Constraints (7) and (8) are to determine earliness and tardiness of job *i*.

In order to determine the problem solution, the Memetic Algorithm is proposed. The evolutionary procedure is applied to search for the good job sequence. In order to determine the optimal schedule for each job sequence, the optimal timing algorithm is constructed and inserted into the evolutionary procedure.

3. Problem Properties

Define the initial schedule as the schedule in which all jobs in the given sequence are started on each machine as soon as possible. Note that in the optimal schedule, each job cannot be processed before its starting times in the initial schedule.

Property 1: If there is no idle time between any two consecutive jobs $J_{[i]}$ and $J_{[i+1]}$ in the initial schedule, the optimal schedule can have idle time between $J_{[i]}$ and $J_{[i+1]}$ only when $t_{[i+1]} - e_{[i]} > P_{[i+1],m}$.

Explanation: Given that, in an initial schedule, there is no idle time between jobs $J_{[i]}$ and $J_{[i+1]}$ on the last machine. In the optimal schedule, if job $J_{[i]}$ is not early $\left(C_{[i],m} \geq e_{[i]}\right)$ and job $J_{[i+1]}$ is not tardy $\left(C_{[i+1],m} \leq t_{[i+1]}\right)$, since $C_{[i+1],m} > C_{[i],m}$, the term $t_{[i+1]} - e_{[i]}$ must be greater than $C_{[i+1],m} - C_{[i],m}$, which can be written as shown in (9).

$$t_{[i+1]} - e_{[i]} \geq C_{[i+1],m} - C_{[i],m} \tag{9}$$

When there is idle time between jobs $J_{[i]}$ and $J_{[i+1]}$, the amount of $C_{[i+1],m} - C_{[i],m}$ is greater than $P_{[i+1],m}$. The condition can be written as follows.

$$C_{[i+1],m} - C_{[i],m} > P_{[i+1],m} \tag{10}$$

The inequality (11) is constructed from inequalities (9) and (10)

$$t_{[i+1]} - e_{[i]} > P_{[i+1],m} \tag{11}$$

From the initial schedule, given that two consecutive jobs are placed in the same cluster with the following conditions.

$$t_{[i+1]} - e_{[i]} \leq P_{[i+1],m} \tag{12}$$

$$S_{[i+1],m} - C_{[i],m} = 0 \tag{13}$$

Property 2: In the optimal schedule, jobs belong to the same cluster must be processed without interruption.

Explanation: Inequality (12) and equation (13) identify the property.

Property 3: In the optimal schedule, for each cluster, those early jobs must be processed before those tardy jobs.

Explanation: Given that, on the last machine, jobs $J_{[i]}$ and $J_{[i+1]}$ are grouped in the same cluster, the following equation can be created by adding the term $-S_{[i],m} - P_{[i],m}$ on both sides of the inequality (12).

$$t_{[i+1]} - S_{[i],m} - P_{[i],m} - P_{[i+1],m} \leq e_{[i]} - S_{[i],m} - P_{[i],m} \tag{14}$$

The inequality (15), reduced from inequality (14), confirms that the early job must be processed before the tardy job.

$$t_{[i+1]} - C_{[i+1],m} \leq e_{[i]} - C_{[i],m} \tag{15}$$

Property 4: In the optimal schedule, for each cluster, those early jobs must be processed before those on-time jobs.

Explanation: Since $e_{[i+1]} < t_{[i+1]}$, the term $e_{[i+1]} - C_{[i+1],m}$ is less than $t_{[i+1]} - C_{[i+1],m}$. Applying this relationship to the inequality (15), the result can be shown as follows which concludes to the property.

$$e_{[i+1]} - C_{[i+1],m} < t_{[i+1]} - C_{[i+1],m} \leq e_{[i]} - C_{[i],m} \tag{16}$$

Property 5: In the optimal schedule, for each cluster, those on-time jobs must be processed before those tardy jobs.

Explanation: Since $e_{[i]} < t_{[i]}$, the term $e_{[i]} - C_{[i],m}$ is less than $t_{[i]} - C_{[i],m}$. Applying this relationship to the inequality (15), the result can be shown as follows which concludes to the property.

$$t_{[i+1]} - C_{[i+1],m} \leq e_{[i]} - C_{[i],m} < t_{[i]} - C_{[i],m} \tag{17}$$

Property 6: In the optimal schedule, if two consecutive jobs of the same cluster are early, then $E_{[i]} > E_{[i+1]}$.

Explanation: If two consecutive early jobs are grouped in the same cluster, the condition $t_{[i+1]} - e_{[i],} \leq P_{[i+1],m}$ must be hold. The result is shown in the following inequality (18).

$$e_{[i+1]} - e_{[i]} < P_{[i+1],m} \tag{18}$$

The following inequality is created by adding the term $-S_{[i],m} - P_{[i],m}$ on both sides of the inequality (18).

$$e_{[i+1]} - S_{[i],m} - P_{[i],m} - P_{[i+1],m} < e_{[i]} - S_{[i],m} - P_{[i],m} \tag{19}$$

The inequality (19) can be reduced to the inequality (20). Since both jobs are early, the two terms on both sides are positive and therefore the earliness of $J_{[i]}$ is greater than the earliness of $J_{[i+1]}$.

$$e_{[i+1]} - C_{[i+1],m} < e_{[i]} - C_{[i],m} \tag{20}$$

Property 7: In the optimal schedule, if two consecutive jobs of the same cluster are tardy, then $T_{[i+1]} > T_{[i]}$.

Explanation: If two consecutive tardy jobs are grouped in the same cluster, the condition $t_{[i+1]} - e_{[i],} \leq P_{[i+1],m}$ must be hold, which results in the following inequality (21).

$$t_{[i+1]} - t_{[i]} < P_{[i+1],m} \tag{21}$$

The following inequality is created by adding the term $-S_{[i],m} - P_{[i],m}$ on both sides of the inequality (21).

$$t_{[i+1]} - S_{[i],m} - P_{[i],m} - P_{[i+1],m} < t_{[i]} - S_{[i],m} - P_{[i],m} \tag{22}$$

The inequality (22) can be reduced to the inequality (23). Here, both jobs are late; the two terms on both sides are negative. Since, the tardiness is the absolute value of lateness, the tardiness of $J_{[i+1]}$ is greater than $J_{[i]}$.

$$t_{[i+1]} - C_{[i+1],m} < t_{[i]} - C_{[i],m} \tag{23}$$

4. Optimal Timing Algorithm

The function of Optimal Timing Algorithm (OPT) is to determine the optimal starting time of each job for a given sequence. The concept of the algorithm starts from constructing the initial schedule and, then, dividing jobs into clusters before grouping several clusters into sub-schedule. The final step of the algorithm is to shift each job cluster to the right side until the sum of earliness and tardiness costs of all jobs is lowest. The concept of OPT procedure introduced in this study can be explained as follows.

On the last machine, any two jobs are assigned to the same cluster when the conditions (12) and (13) are hold. Jobs in the same cluster should be processed continuously without idle time. Since earliness and tardiness of each job are determined by comparing the job completion time on the last machine with its due window, only starting, processing, and completion times of job on the last machine are considered. Applying the clustering method to the initial schedule in the Figure 1, there are three clusters as shown in the Figure 2.

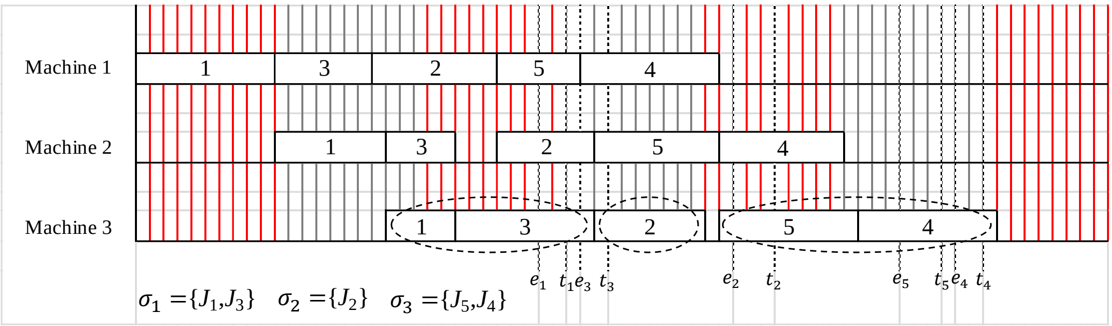

Figure 2. Results obtained from clustering method.

After all clusters are determined, they must be grouped into sub-schedule. The clusters σ_r and σ_{r+1} will be in the same sub-schedule if the equation (24) is hold.

$$C_{L,r} = S_{F,r+1} \tag{24}$$

For any cluster $\sigma_r = \{J_{F,r}, J_{F+1,r}, \ldots, J_{E,r}, J_{E+1,r}, \ldots, J_{W,r}, J_{W+1,r}, \ldots, J_{L,r}\}$, the equation (25) is applied to determine whether or not the cluster should be shifted to the right.

$$\Delta(r) = \sum_{i=F}^{E} \alpha_{i,r} - \sum_{i=W+1}^{L} \beta_{i,r} \tag{25}$$

If $\Delta(r) \leq 0$, the cluster σ_r should not be moved because it will only increase the total cost. On the other hand, if $\Delta(r) > 0$, the cluster σ_r should be shifted to the right side. Therefore, the total cost can be reduced by the product of $\Delta(r)$ and shift distance $E(r)$. Here, the shift distance can be calculated using the equation (26)

$$E(r) = \min\{y_1, y_2, S_{g+1}^{first} - C_g^{last}\} \tag{26}$$

If cluster σ_r does not have any jobs of condition $e_i \leq C_{i,r} < t_i$, then

$$y_2 = \infty \tag{27}$$

If there are no early jobs in the cluster σ_r, this cluster should not be shifted to the right. In this case, the following conditions are hold.

$$\Delta(r) = -\sum_{i=W+1}^{L} \beta_{i,r} \tag{28}$$

$$y_1 = \infty \tag{29}$$

Suppose that all jobs belong to the cluster σ_r are completed within their due windows and ended before their latest due dates ($t_i - C_{i,r} > 0$), then

$$\Delta(r) = 0 \tag{30}$$

$$y_1 = \infty \tag{31}$$

If all jobs in cluster σ_r are tardy jobs, then

$$\Delta(r) = -\sum_{i=W+1}^{L} \beta_{i,r} \tag{32}$$

$$y_1 = \infty \tag{33}$$

$$y_2 = \infty \tag{34}$$

The optimal timing procedure constructed in the research can be summarized as follows:

Step 0: Re-index each job according to its position in given sequence. Then, determine the initial schedule by starting each job on each machine as soon as possible.

Step 1: Divide n jobs on last machine into clusters, if inequality (12) and equation (13) are valid, jobs $J_{[i]}$ and $J_{[i+1]}$ are grouped in the same cluster.

Step 2: Group each cluster into sub-schedule according to equation (24).

Step 3: Set g=0.

Step 4: Set g=g+1, if g>G, go to step 9. Otherwise, calculate $\Delta(r)$ and $E(r)$ for each cluster in sub-schedule g and go to step 5.

Step 5: Determine the minimal h such that $\sum_{r=1}^{h} \Delta(r) \leq 0$.

Step 5.1: if h exists and $h=R_g$, go to step 4. Otherwise, consider step 5.2.

Step 5.2: if h exists and $h \neq R_g$, go to step 6. Otherwise, consider step 5.3.

Step 5.3: if h does not exist, then go to step 7.

Step 6: The first h clusters must not be moved. Remove the first h clusters from consideration, go to step 5 to evaluate the remaining clusters.

Step 7: Consider $(1 \leq r \leq R_g)$, determine the smallest $E(r)$ and, then, shift clusters in the sub-schedule g to the right at the distance equals to the smallest $E(r)$. Go to step 8.

Step 8: If $C_g^{last} < S_{g+1}^{first}$, update $\Delta(r)$ and $E(r)$, go to step 5. Else, if $C_g^{last} = S_{g+1}^{first}$, combine sub-schedule g with sub-schedule g+1 and go to step 4.

Step 9: Stop.

So far, only the operations of jobs on the last machine have been considered. For each job, the starting time of the remaining operations (machines m-1 to 1) can be determined using the following equations.

$$C_{i,k} = \min\left(S_{i+1,k}, S_{i,k+1}\right) \quad ;i = 1,2\ldots,n-1\ ;\ k = 1,2\ldots,m-1 \tag{35}$$

$$C_{n,k} = S_{n,k+1} \quad ;k = 1,2,\ldots,m-1 \tag{36}$$

The calculation of the optimal timing algorithm for the problem discussed in the Example 1 is demonstrated in the Example 2.

Example 2:

In the Example 1, the given job sequence is $J_1 \rightarrow J_3 \rightarrow J_2 \rightarrow J_5 \rightarrow J_4$. The optimal starting time of each job in this sequence can be determined using the optimal timing algorithm explained previously as follows:

Step 0: The initial schedule is shown in Figure 1. Note that by re-indexing jobs according to their positions, jobs J_1, J_3, J_2, J_5 and J_4 can be called $J_{[1]}, J_{[2]}, J_{[3]}, J_{[4]}$, and $J_{[5]}$, respectively.

Step 1: Apply inequality (12) and equation (13) to divide five jobs into clusters. The job members of each cluster are shown as follows:

Cluster 1: $J_{[1]}, J_{[2]}$

Cluster 2: $J_{[3]}$

Cluster 3: $J_{[4]}, J_{[5]}$

Step 2: Group each cluster into sub-schedule according to equation (24)

Sub-schedule 1: Cluster 1, Cluster 2

Sub-schedule 2: Cluster 3

Step 3: Set *g*=0.

Step 4: Set *g*=0+1, consider the first sub-schedule.

$$\Delta(1) = \alpha_{[1]} = 2, E(1) = \min\{6,1,1\} = 1$$

$$\Delta(2) = \alpha_{[3]} = 3, E(2) = \min\{2,\infty,1\} = 1$$

Step 5: $\Delta(1) > 0, \Delta(1) + \Delta(2) > 0$.

Step 5.3: The minimal *h* does not exist, go to step 7.

Step 7: Shift cluster 1 and 2 by $\min\{E(1), E(2)\} = 1$, go to step 8

Step 8: $C_1^{last} = S_2^{first} = 42$, combine sub-schedule 1 with sub-schedule 2. The second sub-schedule is now composed of clusters 1, 2, and 3. Go to step 4.

Step 4: Set g = 1+1 = 2, consider the second sub-schedule.

$$\Delta(1) = \alpha_{[1]} - \beta_{[2]} = 2 - 3 = -1, E(1) = \min\{5,\infty,\infty\} = 5$$

$$\Delta(2) = \alpha_{[3]} = 3, E(2) = \min\{1,\infty,\infty\} = 1$$

$$\Delta(3) = \alpha_{[4]} - \beta_{[5]} = 2 - 4 = -2, E(3) = \min\{3,\infty,\infty\} = 3$$

Step 5: $\Delta(1) < 0$.

Step 5.2: The minimal *h*=1, go to step 6

Step 6: The first cluster does not move. Delete the first cluster from the consideration and go to step 5.

Step 5: $\Delta(2) > 0$, $\Delta(2) + \Delta(3) > 0$.

Step 5.3: The minimal *h* does not exist, go to step 7.

Step 7: Shift cluster 2 and 3 by $\min\{E(2), E(3)\} = 1$, go to step 8

Step 8: $C_2^{last} < S_3^{first}$ $(63 < \infty)$, update $\Delta(2)$, $\Delta(3)$, $E(2)$, $E(3)$. Go to step 5.

$$\Delta(2) = 0, E(2) = \min\{\infty,3,\infty\} = 3$$

$$\Delta(3) = \alpha_{[4]} - \beta_{[5]} = 2 - 4 = -2, E(3) = \min\{2,\infty,\infty\} = 2$$

Step 5: $\Delta(2) = 0$.

Step 5.2: The minimal h=2, go to step 6.

Step 6: The second cluster does not move. Delete the second cluster from the consideration and go to step 5.

Step 5: $\Delta(3) < 0$.

Step 5.1: The minimal h=R_g=3, go to step 4.

Step 4: Set g=2+1=3, g>G (3>2), go to step 9.

Step 9: Stop.

The result obtained from the procedures is shown in the Figure 3 below.

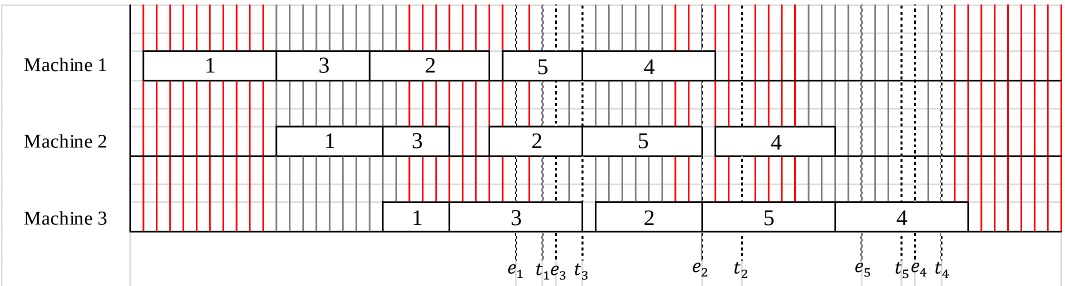

Figure 3. Results from the optimal timing algorithm.

From the Figure 3, jobs 1 and 5 are early while job 4 is tardy. The total cost associated with final schedule can be calculated as (setup cost = \$13) + (earliness cost = \$14) + (tardiness cost = \$8) = \$35.

5. Memetic Algorithm

The Memetic Algorithm is proposed to determine the good solution to the problem in a reasonable amount of time. In this research, the evolutionary procedure is applied to determine the good job sequence, which can be considered as the global search. For the local search, the optimal timing algorithm presented in the previous section is inserted in the evolutionary procedure. The function of OPT is to determine the best starting position of each job for the given job sequence. The Memetic procedure is illustrated in the Figure 4.

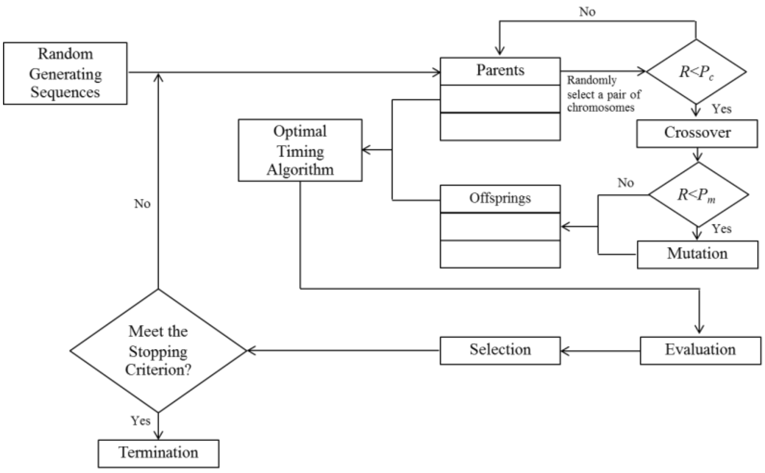

Figure 4. MA procedure.

5.1 Representation and initialization

In the representation, a chromosome can be considered as a sequence of jobs. Each gene is an integer number represented a job in the sequence. For the illustration, a chromosome of [1 3 2 5 4] represents the production sequence of $J_1 \rightarrow J_3 \rightarrow J_2 \rightarrow J_5 \rightarrow J_4$. Note that all machines have the same production sequence and the optimal timing algorithm can be applied to each production sequence in order to generate the best schedule. The chromosomes in the initial population are generated until the number of chromosomes equals to the initial population size.

5.2 Crossover procedure (Uniform order based crossover)

According to Lee and Choi (1995), the uniform order based crossover is considered to be best fit for job sequencing problems. Therefore, this method is selected as the crossover operator in the research.

5.3 Mutation procedure (Swapping mutation)

Each offspring created from the crossover operator is evaluated to see if the mutation should occur. The swapping mutation is used here. The method is to, first, randomly select two genes from a chromosome and, then, exchange their positions.

5.4 Evaluation

The purpose of evaluation is to determine quality and fitness value of each chromosome. The total cost of each chromosome (TC_i), represented the chromosome quality, can be calculated according to the objective function mentioned in the session 2.1. The chromosome fitness value (f_i) is determined using the equation 37. This value can be considered as the probability that each chromosome will be selected as a member of the next generation. Here, those good chromosomes with low total costs have greater chances to be selected.

$$f_i = \frac{1}{TC_i} \tag{37}$$

5.5 Selection

Similar to the work of Cheng *et al*. (1995), two selection operations, elitist and roulette wheel, are applied to perform the reproduction step of evolutionary procedures. The elitist is implemented to preserve the best chromosome in the enlarge population (parents + off-springs) of current generation for the population of next generation. The roulette wheel is, then, applied to select the remaining chromosomes to be the members of the next generation in such a way that a fitter chromosome has greater chance to be selected.

6. Computational Results

This section is to evaluate the performance of Memetic Algorithm discussed previously. The solution obtained from the Memetic Algorithm is compared with the optimal solution yielded from Integer Linear Programming (small size problems) and the Branch and Bound method (small and large size problems). All three approaches are coded with the MATLAB R2014b and compiled with the Intel(R) Core(TM) i7 CPU processor 3.07 GHz RAM 7.88 GB. In the Branch and Bound (B&B), the partial job schedules are created by applying the optimal timing algorithm to partial job sequences. The remaining partial job sequences after applying the concept of Branch and Bound to the Example 1, using the solution obtained from the MA as an upper bound, are demonstrated in the Figure 5. The deviation percentage value *(%Dev*) applied to evaluate the MA performance can be calculated as:

$$(\%Dev) = \frac{(TC_{MA} - TC_{Opt}) \times 100}{TC_{Opt}} \tag{38}$$

where TC_{MA} is the total cost obtained from the MA.

TC_{Opt} is the optimal total cost obtained from the ILP (small size problems) and the B&B (small and large size problems).

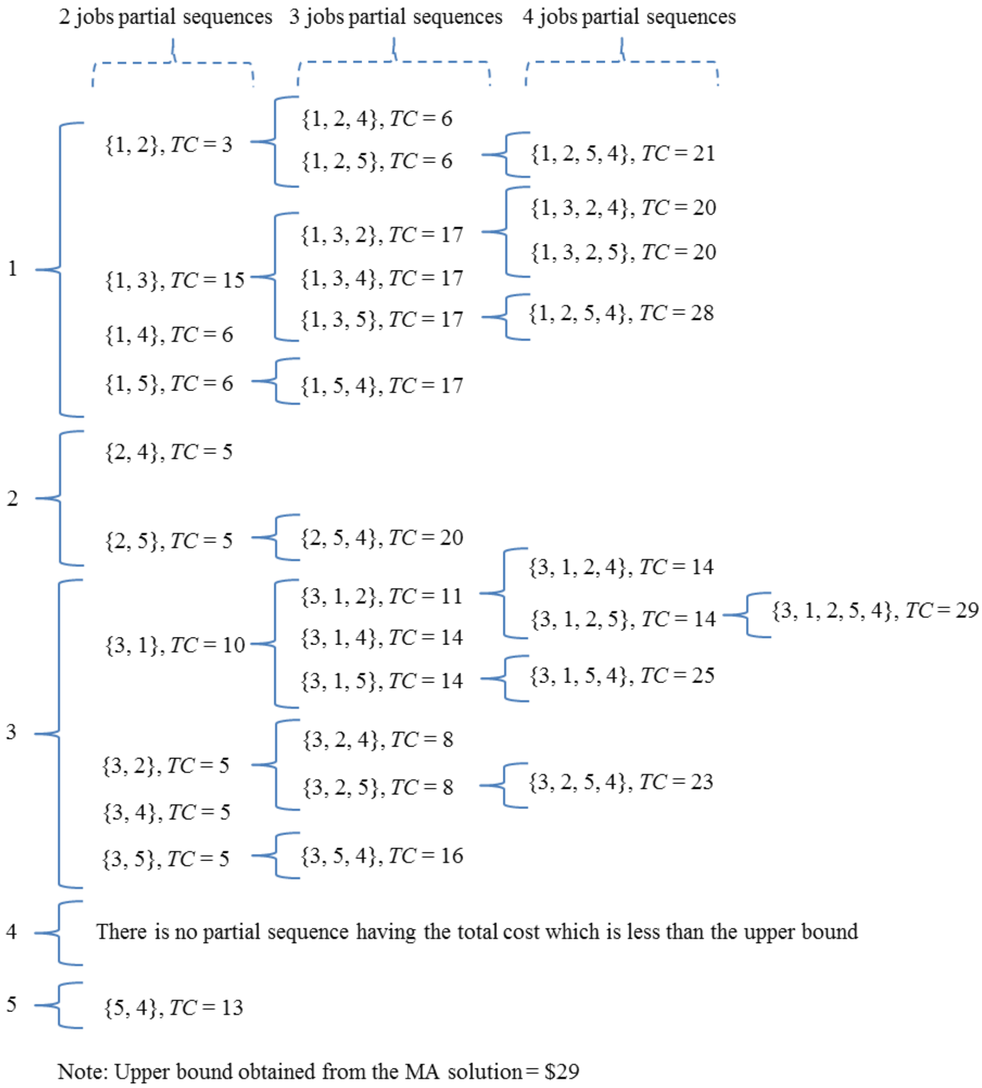

Figure 5. Branch and Bound structure for Example 1.

Table 2 presents the details of problem setup. To determine the proper probabilities of crossover (p_c) and mutation (p_m), forty trial problems were evaluated. The experiment suggests that the combination of p_c=0.8 and p_m=0.1 should be selected. In comparison to the other, this combination yields the best result for thirty four out of forty problems. Three stopping criteria are applied in the MA. The first criterion stops the search when total cost of the best chromosome in a generation equals to zero. The second criterion terminates the MA when the total cost reduction percentage is smaller than 0.01 after 200 consecutive generations. The last criterion finishes the search when the number of generations reaches 1,000.

In the experiment, the influences of number of jobs (3 levels), number of machines (2 levels), and ratio of tardiness to earliness penalties (4 levels) on the MA performance are evaluated. Note that there are totally twenty four treatments with fifteen replications in each treatment. The summary results from twenty four treatment combinations are shown in Table 3.

Table 2. Details of problem setup.

Characteristics	Values
Number of jobs (n)	5, 10, 12
Number of machines (m)	3, 5
Processing time ($P_{i,k}$)	Discrete uniform [1,10]
Due dates	Discrete uniform [$\min_{\forall i}(\sum_{k}^{m} P_{i,k}),(1.5\sum_{i=1}^{n}\sum_{k=1}^{m} P_{i,k})/m)$]
Deviation of Due dates	Discrete uniform [1,3]
Earliest due dates (e_i)	Due dates - Deviation of Due dates
Latest due dates (t_i)	Due dates + Deviation of Due dates
Earliness cost (α_i)	Discrete uniform [1,5]
Tardiness cost (β_i)	$0.5\alpha_i$, $1.0\alpha_i$, $1.5\alpha_i$, $2.0\alpha_i$
Setup cost ($\gamma_{i,j}$)	Discrete uniform [0,5]

Table 3. Average deviation percentage and average computational time of each treatment (15 replications of each treatment).

Treatments $(n,m,\beta/\alpha)$	Number of optimal solution found	Average Deviation Percentage Value from 15 problems	Average Computational Time (sec.)		
			MA	Branch and Bound	ILP
(5,3,0.5)	15	0.00	10.51	0.04	9.39
(5,3,1.0)	15	0.00	9.56	0.04	10.08
(5,3,1.5)	15	0.00	10.59	0.03	8.55
(5,3,2.0)	15	0.00	9.76	0.03	10.58
(5,5,0.5)	15	0.00	9.78	0.05	21.31
(5,5,1.0)	15	0.00	10.13	0.05	23.36
(5,5,1.5)	15	0.00	9.69	0.05	23.04
(5,5,2.0)	15	0.00	9.49	0.06	21.17
(10,3,0.5)	15	0.00	26.97	65.34	N/A
(10,3,1.0)	15	0.00	22.54	391.08	N/A
(10,3,1.5)	15	0.00	22.42	429.82	N/A
(10,3,2.0)	15	0.00	21.29	126.33	N/A
(10,5,0.5)	14	0.02	22.01	783.62	N/A
(10,5,1.0)	14	0.02	20.66	952.54	N/A
(10,5,1.5)	15	0.00	20.38	1320.30	N/A
(10,5,2.0)	15	0.00	19.62	2780.94	N/A
(12,3,0.5)	14	0.48	39.54	1379.84	N/A
(12,3,1.0)	15	0.00	38.13	1471.08	N/A
(12,3,1.5)	15	0.00	30.63	5503.93	N/A
(12,3,2.0)	15	0.00	32.07	4902.84	N/A
(12,5,0.5)	15	0.00	27.40	9684.52	N/A
(12,5,1.0)	15	0.00	26.89	10973.95	N/A
(12,5,1.5)	15	0.00	29.06	10565.61	N/A
(12,5,2.0)	13	0.41	27.40	19352.21	N/A

*Note that the deviation percentage value demonstrates the percentage of difference between the solution obtained from the MA and the optimal solutions yielded from ILP and B&B.

From the study result, the MA yields optimal solution for 355 out of 360 problems. The treatments of (10, 5, 0.5), (10, 5, 1.0) ,and (12, 3, 0.5) provide the optimal solution for 14 out of 15 replications and the treatment of (12, 5, 2.0) found the optimal solution for 13 out of 15 replications. On average, the maximum deviation percentage, occurring in the treatment of (12, 3, 0.5), is 0.48 percent. The maximum computational time of the MA and the Branch and Bound method are 39.54 and 19,352.21 seconds, respectively. It is obvious that when the problem size is getting larger, the computational time of Branch and Bound increases dramatically. This result emphasizes that the MA heuristic is appropriate to be applied to those medium and large size problems.

7. Conclusions

The research addresses the flow shop scheduling problem with jobs having different due windows under the just in time philosophy. The objective is to minimize total cost composing of setup, earliness, and tardiness costs. The mathematical model is developed to represent the problem. The Memetic Algorithm with the insertion of optimal timing algorithm has been created to determine the good solution in a reasonable amount of time. According to the method proposed, the function of evolutionary procedure is to search for the good production sequences. The optimal timing algorithm is, then, applied to determine the optimal schedule of each production sequence. For performance evaluation, the solutions obtained from MA heuristic is compared with the optimal solutions yielded from the Branch and Bound method. From the study result of 360 problems, the MA heuristic provides the optimal solutions for 355 problems. On average, the maximum computational time of the MA and the Branch and Bound are 39.54 and 19,352.21 seconds, respectively. This result emphasizes the benefit of applying MA heuristic to solve the problem of medium and large sizes.

Acknowledgements

This research received the funding of Kasetsart University Scholarship for Doctoral Student, which is granted by the Kasetsart University.

References

Baker, K. R., & Scudder, G. D. (1990). Sequencing with earliness and tardiness penalties: A Review. *Operations Research*, *38*(1), 22-37.

Behnamian, J., Zandieh, M., & Fatemi Ghomi, S. M. T. (2009). Due window scheduling with sequence-dependent setup on parallel machines using three hybrid meta-heuristic algorithms. *The international Journal of Advanced Manufacturing and Technology*, *44*, 795–808.

Chandra, P., Mehta, P., & Tirupati, D. (2009). Permutation flow shop scheduling with earliness and tardiness penalties. *International Journal of Production Research, 47*(20), 5591–5610.

Cheng, R., Gen, M., & Tozawa, T. (1995). Minmax earliness/tardiness scheduling in identical parallel machine system using genetic algorithms. *Computers and Industrial Engineering*, *30*(1-4), 513–517.

Janiak, A., Janiak, W. A., Tomasz, K., & Tomasz, K. (2015). A survey on scheduling problems with due windows. *European Journal of Operational Research, 242*, 347–357.

Koulamas, C. (1996). Single-machine scheduling with time windows and earliness/tardiness penalties. *European Journal of Operational Research, 91*, 190–202.

Lee, C. Y., & Choi, J. Y. (1995). A genetic algorithm for job sequencing problems with distinct due dates and general early-tardy penalty weights. *Computers and Operations Researches, 22*(8), 857-869.

Lee, C. Y., & Kim, S. J. (1995). Parallel genetic algorithms for the earliness tardiness job sequencing problem with general penalty weights. *Computers and Industrial Engineering, 28*(2), 231-243.

M'Hallah, R. (2014). An iterated local search variable neighborhood descent hybrid heuristic for the total earliness tardiness permutation flow shop. *International Journal of Production Research, 52*(13), 3802–3819.

M'Hallah, R. (2014). Minimizing total earliness and tardiness on a permutation flow shop using VNS and MIP. *Computers and Industrial Engineering, 75,* 142–156.

Moslehi, G., Mirzaee, M., Vasei, M., Modarres, M., & Azaron, A. (2009). Two-machine flow shop scheduling to minimize the sum of maximum earliness and tardiness. *International Journal of Production Economics, 122*, 763–773.

Murata, T., Ishibuchi, H., & Tanaka, H. (1996). Genetic algorithms for flow shop scheduling problem. *Computers and Industrial Engineering, 30*(4), 1061-1071.

Pinedo, M. (2002). *Scheduling theory, algorithms, and systems*. New Jersey, NJ: Prentice Hall.

Reeves, C. R. (1995). Genetic algorithms for flow shop sequencing. *Computers and Operations Research, 22*(1), 5–13.

Sarper, H. (1995). Minimizing the sum of absolute deviations about a common due date for the two-machine flow shop problem. *Applied Mathematical Modelling, 19*, 698-710.

Schaller, J., & Valente, M. S. J. (2013). A comparison of metaheuristic procedures to schedule jobs in a permutation flow shop to minimise total earliness and tardiness. *International Journal of Production Research, 51*(3), 772–779.

Sevaux, M., & Dauzere-Peres, S. (2003). Genetic algorithms to minimize the weighted number of late jobs on a single machine. *European Journal of Operational Research, 151*, 296–306.

Sufen, L., Yunlong, Z., & Xiaoying, L. (2005). Earliness/tardiness flow-shop scheduling under uncertainty. *Proceeding of the 17th IEEE International Conference on Tool with Artificial Intelligence*, 499-506.

Szwarc, W., & Mukhopadhyay, S. K. (1995). Optimal timing schedules in earliness-tardiness single machine sequencing. *Naval Research Logistics, 42*, 1109-1114.

Tavakkoli-Moghaddam, R., Safaei, N., & Sassani, F. (2009). A memetic algorithm for the flexible flow line scheduling problem with processor blocking. *Computers and Operations Research, 36*, 402–414.

Wan, G., & Yen, B. P. C. (2002). Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties. *European Journal of Operational Research, 142*, 271–281.

Yeung, W. K., Oguz, C., & Cheng, T. C. E. (2004). Two-stage flow shop earliness and tardiness machine scheduling involving a common due window. *International Journal of Production Economics, 90*, 421-434.

Yoon, S. H., & Ventura, J. A. (2002). An application for class of single-machine weighted earliness and tardiness problems. *European Journal of Operational Research, 52*, 167-178.