Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling (CI-Sched 2007)

A Genetic Algorithm with Dominance Properties for Single Machine Scheduling Problems

Shih-Shin Chen, Pei-Chann Chang, Shih-Min Hsiung, Chin-Yuan Fan.

Abstract- **This paper considers a single machine scheduling problem in which n jobs are to be processed and a machine setup time is required when the machine switches jobs from one to the other. All jobs have a common due date that has been predetermined using the median of the set of sequenced jobs. The objective is to find an optimal sequence of the set of n jobs to minimize the sum of the job's setups and the cost of tardy or early jobs related to the common due date. Dominance properties are developed according to the sequence swapping of two neighborhood jobs. These dominance properties are further embedded in the Simple Genetic Algorithm to improve the efficiency and effectiveness of the global searching procedure. Analytical results in benchmark problems are presented and computational algorithms are developed.**

I. Introduction

Single-machine scheduling problems are one of the well-known problems that have been studied by many researchers. The results not only provide the insights into the single machine problem but also for more complicated environment.

The problem considered in this paper is scheduling a set of n jobs *{*1,2,….,n*}* on a single machine that is capable of processing only one job at a time without preemption. All jobs are available at time zero, and a job j requires processing time P_j. Machine setup time S_{ij} is included as sequence dependent. That is, the amount of machine setup required if job i proceeds j may be different from when job j precedes i. The objective is to complete all the jobs as close as possible to a large, common due date d. To accomplish this objective, the summation of earliness and tardiness is minimized. The earliness of job j is defined as $E_j = \max\left(0, d - C_j\right)$ and its tardiness as $T_j = \max\left(0, C_j - d\right)$, where C_j is the completion time of job j. Earliness and tardiness penalties for job j are weighted equally. The objective function is given by

Manuscript received October 30, 2006.

Shih-Shin Chen., is with Department of Industrial Engineering and Management, Yuan-Ze University. (e-mail: s939506@mail.yzu.edu.tw).
Pei-Chann Chang is with the Department of information Management, Yuan-Ze University. 135 Yuan-Dong Rd., Taoyuan 32026, Taiwan, R.O.C (+886 03 4638800#2503 e-mail: iepchang@saturn.yzu.edu.tw).
Shih-Min Hsiung,is with Department of Industrial Engineering and Management, Yuan-Ze University (e-mail: s945409@mail.yzu.edu.tw).
Chin-Yuan Fan is with Department of Industrial Engineering and Management, Yuan-Ze University (e-mail: s948906@mail.yzu.edu.tw)

$$\min Z = \sum_{j=1}^{n}\left(E_j + T_j\right) = \sum_{j=1}^{n}\left|d - C_j\right| \qquad (1)$$

The inclusion of both earliness and tardiness costs in the objective function is compatible with the philosophy of just-in-time production, which emphasizes producing goods only when they are needed. The early cost may represent the cost of completing a product early, the deterioration cost for perishable goods or a holding (stock) cost for finished goods. The tardy cost can represent rush shipping costs, lost sales and loss of goodwill. Some specific examples of production settings with these characteristics are provided by Ow and Morton[15], Wu et al.[21], Su and Chang [17] and Su and Chang[18]. The set of jobs is assumed to be ready for processing at the beginning which is a characteristic of the deterministic problem. As a generalization of weighted tardiness scheduling, the problem is strongly NP-hard.

The single-machine E/T problem was first introduced by Kanet[13]. Since then many researchers worked on various extensions of the problem. Baker and Scudder [5] published a comprehensive state-of-the-art review for different versions of the E/T problem. Kanet[13] examined the E/T problem with equal penalties and unrestricted common due date. A problem is considered unrestricted when the due date is large enough not to constrain the scheduling process. He introduced a polynomial-time algorithm to solve the problem optimally. Hall [10] extended Kanet's work and developed an algorithm that a set of optimal solutions for the problem based on some optimality conditions. Hall and Posner [12] solved the weighted version of the problem with no setup times. Azizoglu and Webster [3] introduced a B&B approach to solve the problem with setup times; however, they assumed that setup times are not sequence dependent. Other researchers worked on the same problem but with a restricted (small) due date (see for example Bagchi et al.[4], Szwarc [19], Szwarc [20], Hall et al. [10], Alidaee and Dragan [1], and Mondal and Sen [14]. However, none of the previous papers considered sequence-dependent setup times.

In most of the E/T literature, it has been assumed that no setup time is required. In many realistic situations, however, setup times are needed and are sequence-dependent. In general, scheduling problems with sequence-dependent setup times are similar to the traveling salesman problem (TSP), which is NP-hard [9]. Coleman [7] presented a 0/1 mixed integer programming model (MIP) for the single-machine E/T problem with job-dependent penalties, distinct due dates, and sequence-dependent setup times. Coleman's work was one of the few papers that dealt with the E/T problem with sequence-dependent setup times, but only for a small number of jobs. Chen [6]addressed the E/T problem with batch sequence-dependent setup times. He showed that the problem with unequal penalties is NP-hard even when there are only two batches of jobs and two due dates that are unrestrictively large. Allahverdi et al. [2] reviewed the scheduling literature that involved setup times. In their review, very few papers addressed the E/T problem with setup times, and no papers ever tackled the problem as addressed in this research with the development of dominance properties. As a result, this paper fills the gape of proving some dominance properties for E/T problem with setup times in a single machine scheduling problem.

II. PROBLEM STATEMENTS

We consider the sequence-dependent scheduling problem with a common due date. The common due date model corresponds; for instance, to an assembly system in which the components of the product should be ready at the same time, or to a shop where several jobs constitute a single customer's order (Gordon et al., [10]). It is shown in Cheng [7] that an optimal sequence in which the *b*-th job is completed at the due-date. The value of *b* is given by:

$$b = \begin{cases} n/2 & \text{if } n \text{ is even,} \\ (n+1)/2 & \text{if } n \text{ is odd.} \end{cases} \tag{2}$$

The optimal common due-date (*k**) is the sum of processing times of jobs in the first *b* position of the sequence; i.e.,

$$k^* = C_b \tag{3}$$

As soon as the common due date is assigned, see Fig. 1, jobs can be classified into two different groups that are either early or tardy. The early group is from position 1 to position b and the late group is from position b+1 to position n respectively. The following notations are employed in the latter section.

[j]: job in position j

A: the set of tardy jobs

B: the set of early jobs

$AP_{[j][j+1]}$: Adjusted processing time for the job in position [j +1] preceded by the job in position [j]

b : the median position

$AP_{[j][j+1]}$ is actually the processing time of job j+1 with setup time. Thus, $AP_{[j][j+1]}$ is equal to $S_{[j][j+1]} + P_{j+1}$.

Our objective is to minimize the earliness/tardiness cost. The formulation is given below.

$$\text{Minimize } Z = \sum_{i=1}^{n}(E_i + T_i) = TT + TE \tag{4}$$

where

TT: Total tardiness for a job sequence

TE: Total earliness for a job sequence

$$TT = \sum_{j=b}^{n-1}(n-j)AP_{[j][j+1]} \tag{5}$$

$$TE = \sum_{j=1}^{b}(j-1)AP_{[j-1][j]} \tag{6}$$

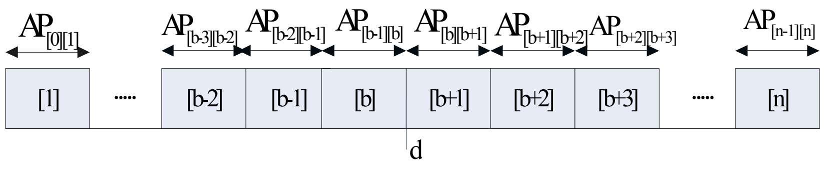
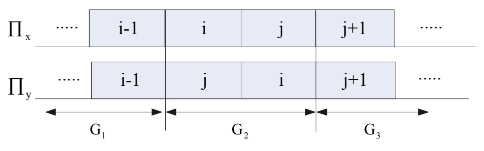


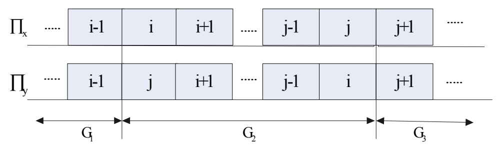
Fig. 1. A figure demonstrates the total earliness and total tardiness in our problem

(a) Adjacent interchange

III. DERIVATIONS OF DOMINANCE PROPERTIES

We consider the problem of scheduling n jobs in a single machine and derive the dominance properties (necessary conditions) of the optimal schedule. In this section, we use the objective function ($Z(\prod)$) for total absolute deviation of schedule $\prod$. To derive the dominance properties for schedule $\prod$, we consider interchanging two adjacent jobs and nonadjacent jobs within the schedule, and prove some intermediate results. The adjacent interchange and nonadjacent interchange of job *i* and job *j* are depicted at figure 2(a) and 2(b) respectively.





(b) Nonadjacent interchange

Fig 2. The two types of interchanging method

$$Z(\prod_x) = G_1 + G_2 + G_3 \tag{7}$$

$$Z(\prod_y) = G_1 + G_2' + G_3' \tag{8}$$

where

G_1 : the objective value of $\prod_X$ for job(s) before job *i*

G_2 : the objective value of $\prod_X$ for jobs between job *i* and job *j*

G_3 : the objective value of $\prod_X$ for job(s) after job *j*

G_2' : the objective value of $\prod_Y$ for jobs between job *j* and job i

G_3' : the objective value of $\prod_Y$ for job(s) after job i

We will compare schedules $\prod_X$ with schedule $\prod_Y$ by finding the conditions under which $\prod_X$ is better than $\prod_Y$. It does not matter for adjacent interchange or nonadjacent interchange, for a given schedule jobs i and job j will be in one of the following status:

1. Job i is early and job j is early
2. Job i is early and job j is on-time
3. Job i is on-time and job j is tardy
4. Job i is tardy and job j is tardy

Because the objective of adjacent and nonadjacent interchange is different, there are totally 8 conditions corresponding to the two exchanging type. Except for the above statuses, there exist one status that is considered in nonadjacent interchange which is:

5. Job i is early and job j is tardy

With above descriptions, we prove the dominance properties under these statuses one by one in detail. There are four dominance properties of the adjacent interchange are proofed at section A and five dominance properties is for the nonadjacent interchange shown at section B.

A. Dominance Properties of Adjacent Interchange

When we exchange two adjacent jobs (see the figure 2), the objective corresponding to jobs (job [i] and job [j]) in position i, i+1, and i+2 are changed while others are the same. These objective terms in position i, i+1, and i+2 are different. Consequently, when we subtract the $Z(\prod_Y)$ with $Z(\prod_X)$, redundant terms are reduced.

Lemma 1a. In a given schedule $\prod_X$, for any two adjacent jobs (job i and job j) are early, then the total deviation of $Z(\prod_Y)$ is better than $Z(\prod_X)$ only when

$$\text{(i-1)}(AP_{[i-1][j]}) + \text{(j-1)}(AP_{[j][i]}) + \text{(j)}(AP_{[i][j+1]})$$
$$\le \text{(i-1)}(AP_{[i-1][i]}) + \text{(j-1)}(AP_{[i][j]}) + \text{(j)}(AP_{[j][j+1]})$$

Proof:

The objective terms of $Z(\prod_X)$ and $Z(\prod_Y)$ are shown as following:

$$G_1 = \sum_{k=1}^{i-2} \text{(k-1)} AP_{[k][k+1]}$$

$$G_2 = \text{(i-1)}AP_{[i-1][i]} + \text{(j-1)}AP_{[i][j]}$$

$$G_3 = \sum_{k=j+1}^{b} \text{(k-1)}AP_{[k-1][k]} + \sum_{k=b}^{n-1} \text{(n-k)}AP_{[k][k+1]}$$

$$G_2' = \text{(i-1)}AP_{[i-1][j]} + \text{(j-1)}AP_{[i][j]}$$

$$G_3' = \sum_{k=i+1}^{b} \text{(k-1)}AP_{[k-1][k]} + \sum_{k=b}^{n-1} \text{(n-k)}AP_{[k][k+1]}$$

We now derive the condition under which $Z(\prod_X) \ge Z(\prod_Y)$. For this purpose, we obtain the value of $Z(\prod_Y) - Z(\prod_X)$. Let $X = Z(\prod_Y) - Z(\prod_X)$ and is given by $X = (G_2' - G_2) + (G_3' - G_3)$. From the above expression, we see that $X \le 0$ when the following condition is satisfied.

$$\text{(i-1)}(AP_{[i-1][j]}) + \text{(j-1)}(AP_{[j][i]}) + \text{(j)}(AP_{[i][j+1]})$$
$$\le \text{(i-1)}(AP_{[i-1][i]}) + \text{(j-1)}(AP_{[i][j]}) + \text{(j)}(AP_{[j][j+1]})$$

From the above condition, we see that if X<0, then the schedule $\prod_Y$ is better than the schedule $\prod_X$; i.e., $Z(\prod_Y) < Z(\prod_X)$. For this case, job j will come before job i.

After we show the first lemma in detail, the latter 8 lemmas can be proven in the same way

Lemma 2a. In a given schedule $\prod_X$, for any two adjacent jobs (job i and job j) are early and on-time, then the total deviation of $Z(\prod_Y)$ is better than $Z(\prod_X)$ only when

$$\text{(i-1)}(AP_{[i-1][j]}) + \text{(j-1)}(AP_{[j][i]}) + (n-\text{j})(AP_{[i][j+1]})$$
$$\le \text{(i-1)}(AP_{[i-1][i]}) + \text{(j-1)}(AP_{[i][j]}) + (n-\text{j})(AP_{[j][j+1]})$$

.**Lemma 3a.** In a given schedule $\prod_X$, for any two adjacent jobs (job i and job j) are on-time and tardy, then the total deviation of $Z(\prod_Y)$ is better than $Z(\prod_X)$ only when

$$\text{(i-1)}(AP_{[i-1][j]}) + \text{(n-i)}(AP_{[j][i]}) + (n-j)(AP_{[i][j+1]})$$
$$\le \text{(i-1)}(AP_{[i-1][i]}) + \text{(n-i)}(AP_{[i][j]}) + (n-\text{j})(AP_{[j][j+1]})$$

Lemma 4a. In a given schedule $\prod_X$, for any two adjacent jobs (job i and job j) are tardy and tardy, then the total deviation of $Z(\prod_Y)$ is better than $Z(\prod_X)$ only when

$$\text{(n-i+1)}(AP_{[i-1][j]}) + \text{(n-j+1)}(AP_{[j][i]}) + (n-\text{j})(AP_{[i][j+1]})$$
$$\le \text{(n-i+1)}(AP_{[i-1][i]}) + \text{(n-j+1)}(AP_{[i][j]}) + (n-\text{j})(AP_{[j][j+1]})$$

B. Dominance Properties of Nonadjacent Interchange

If we consider the jobs are nonadjacent, the corresponding objective terms are changed in position i, i+1, k, and k+1. Thus, compared with the adjacent neighborhood interchange, there is an extra term when we compare the $Z(\prod_X)$ and $Z(\prod_Y)$.

Lemma 1b. In a given schedule $\prod_X$, for any two nonadjacent jobs (job i and job j) are both early, then the total deviation of $Z(\prod_Y)$ is better than $Z(\prod_X)$ only when

$$\text{(i-1)}(AP_{[i-1][j]}) + \text{(i)}(AP_{[j][i+1]}) + \text{(j-1)}(AP_{[j-1][i]}) + \text{(j)}(AP_{[i][j+1]}) .$$
$$\le \text{(i-1)}(AP_{[i-1][i]}) + \text{(i)}(AP_{[i][i+1]}) + \text{(j-1)}(AP_{[j-1][j]}) + \text{(j)}(AP_{[j][j+1]})$$

Lemma 2b. In a given schedule $\prod_X$, for any two nonadjacent jobs (job i and job j) are early and on-time, then the total deviation of $Z(\prod_Y)$ is better than $Z(\prod_X)$ only when

$$\text{(i-1)}(AP_{[i-1][j]}) + \text{(i)}(AP_{[j][i+1]}) + \text{(j-1)}(AP_{[j-1][i]}) + (n-\text{j})(AP_{[i][j+1]})$$
$$\le \text{(i-1)}(AP_{[i-1][i]}) + \text{(i)}(AP_{[i][i+1]}) + \text{(j-1)}(AP_{[j-1][j]}) + (n-\text{j})(AP_{[j][j+1]})$$

.

Lemma 3b. In a given schedule $\prod_X$, for any two nonadjacent jobs (job i and job j) are on-time and tardy, then the total deviation of $Z(\prod_Y)$ is better than $Z(\prod_X)$ only when

$$\text{(i-1)}(AP_{[i-1][j]})+\text{(n-i)}(AP_{[j][i+1]})+\text{(n-j+1)}(AP_{[j-1][i]})+\text{(n-j)}(AP_{[i][j+1]})$$
$$\le \text{(i-1)}(AP_{[i-1][i]})+\text{(n-i)}(AP_{[i][i+1]})+\text{(n-j+1)}(AP_{[j-1][j]})+\text{(n-j)}(AP_{[j][j+1]})$$

Lemma 4b. In a given schedule $\prod_X$, for any two nonadjacent jobs (job *i* and job *j*) are both tardy, then the total deviation of $Z(\prod_Y)$ is better than $Z(\prod_X)$ only when

$$(n-\mathrm{i}+1)(\mathrm{AP}_{[\mathrm{i}-1][\mathrm{j}]})+(n-\mathrm{i})(\mathrm{AP}_{[\mathrm{j}][\mathrm{i}+1]})+(\mathrm{n}\text{-}\mathrm{j}+1)(\mathrm{AP}_{[\mathrm{j}-1][\mathrm{i}]}) +(n-\mathrm{j})(\mathrm{AP}_{[\mathrm{i}][\mathrm{j}+1]}) \le (n-\mathrm{i}+1)(\mathrm{AP}_{[\mathrm{i}-1][\mathrm{i}]})+(n-\mathrm{i})(\mathrm{AP}_{[\mathrm{i}][\mathrm{i}+1]})+(\mathrm{n}\text{-}\mathrm{j}+1)(\mathrm{AP}_{[\mathrm{j}-1][\mathrm{j}]}) +(n-\mathrm{j})(\mathrm{AP}_{[\mathrm{j}][\mathrm{j}+1]}).$$

Lemma 5. In a given schedule $\prod_X$, for any two jobs (job *i* and job *j*) are early and tardy, then the total deviation of $Z(\prod_Y)$ is better than $Z(\prod_X)$ only when

$$(\mathrm{i}\text{-}1)(\mathrm{AP}_{[\mathrm{i}-1][\mathrm{j}]})+(\mathrm{i})(\mathrm{AP}_{[\mathrm{j}][\mathrm{i}+1]})+(\mathrm{n}\text{-}\mathrm{j}+1)(\mathrm{AP}_{[\mathrm{j}-1][\mathrm{i}]})+(n-\mathrm{j})(\mathrm{AP}_{[\mathrm{i}][\mathrm{j}+1]}) \le (\mathrm{i}\text{-}1)(\mathrm{AP}_{[\mathrm{i}-1][\mathrm{i}]})+(\mathrm{i})(\mathrm{AP}_{[\mathrm{i}][\mathrm{i}+1]})+(\mathrm{n}\text{-}\mathrm{j}+1)(\mathrm{AP}_{[\mathrm{j}-1][\mathrm{j}]})+(n-\mathrm{j})(\mathrm{AP}_{[\mathrm{j}][\mathrm{j}+1]}).$$

IV. IMPLEMENTATION OF GENETIC ALGORITHM WITH DOMINANCE PROPERTIES

A. Development of the Hybrid Algorithm

These dominance properties can work as a standalone heuristic or to be integrated with exact algorithm or metaheuristic. This paper proposes a two-phase hybrid algorithm that combines these dominance properties with genetic algorithm, which is genetic algorithm with dominance properties as GADP in short. The first phase is the solution construction phase where a random solution is generated. At the same time, a general pair-wise interchange (GPI), which is a neighborhood search method, is applied. When we use GPI iteratively, we can obtain a set of constructed solutions. The pseudo code of the main procedure and the first phase are demonstrated as the following:

Algorithm 1: Main()

Population: It represents the solutions (chromosomes) in genetic algorithm.

popSize: The population size

```
1.  initializePopulationSize()
2.  for i = 0 to popSize do //The first phase
3.      Population [i] ← GPI()
4.  End for
5.  Genetic Algorithm() //The second phase
```

Algorithm 2: GPI()

```
1.  sequence ← generateRandomSolution()
2.  for i = 0 to n do
3.      for increment = 1 to 3 do
4.          for pos = 0 to n - increment do
5.              dominanceProperty(sequence, pos, pos+ increment)
6.          End for
7.          if sequence doesn't be changed do
8.              break;
9.          End if
10.     End for
11. End for
12. return sequence
```

Consequently, the time-complexity of the phase 1 is O(n^2) and the constructed solutions are employed into the second phase while the genetic algorithm does selection, crossover, and mutation.

B. A Small Example of the First Phase

The primary idea of the dominance properties is to compare the result of $Z(\prod_X)$ and $Z(\prod_Y)$. If $Z(\prod_Y) < Z(\prod_X)$, the exchanged schedule is better than previous one. In order to explain the first phase clearly, there is an eight jobs example that is presented. The following is the adjusted processing time matrix and the AP_{25} means the job 2 before job 5 whose corresponding processing time is 3.

Table 1.
The adjusted processing time of eight jobs

	1	2	3	4	5	6	7	8
1		27	6	18	21	7	14	11
2	25		29	13	9	17	20	22
3	8	15		24	29	3	21	10
4	21	16	12		18	27	5	13
5	23	13	4	16		22	30	8
6	15	9	23	17	20		6	16
7	4	14	21	28	17	19		7
8	18	24	7	13	15	21	28	

Suppose a sequence [0, 5, 1, 6, 4, 2, 7, 3] is generated randomly which means the first job is job 0 and the second job is the job 5, and so on. In the beginning, the first job interchanges with the second job, the second job exchanges with the third job, and so on. Thus, the following steps illustrated these results step-by-step.

Iteration 1:

Step 1: (Apply Lemma 1a.)

$\prod_X$: [**0**, **5**, 1, 6, 4, 2, 7, 3]

$\prod_Y$: [**5**, **0**, 1, 6, 4, 2, 7, 3]

$Z(\prod_Y)$ - $Z(\prod_X)$ = (1)(15-7) + (2)(27-9) = 44 (>0)

So we do not exchange the job 0 and job 5.

Step 2: (Apply Lemma 1a.)

$\prod_X$: [0, **5**, **1**, 6, 4, 2, 7, 3]

$\prod_Y$: [0, **1**, **5**, 6, 4, 2, 7, 3]

$Z(\prod_Y)$ - $Z(\prod_X)$ = (1)(27-7) + (2)(17-9) + (3)(6-20) = -6 (<0)

Because $\prod_Y$ is better than $\prod_X$, job 5 and job 1 are swapped, and the new sequence is [0, 1, 5, 6, 4, 2, 7, 3].

Step 3: (Apply Lemma 2a.)

$\prod_X$: [0, 1, **5**, **6**, 4, 2, 7, 3]

$\prod_Y$: [0, 1, **6**, **5**, 4, 2, 7, 3]

$Z(\prod_Y)$ - $Z(\prod_X)$ = (2)(20-17) + (3)(19-6) + (4)(20-17) = 57 (>0)

There is no change of the sequence.

Step 4: (Apply Lemma 3a.)

$\prod_X$: [0, 1, 5, **6**, **4**, 2, 7, 3]

$\prod_Y$: [0, 1, 5, **4**, **6**, 2, 7, 3]

$Z(\prod_Y) - Z(\prod_X) = (3)(20\text{-}6) + (4)(30\text{-}17) + (3)(21\text{-}4) = 145\ (>0)$

There is no action on the current sequence.

Step 5: (Apply Lemma 4a.)

$\prod_X$: [0, 1, 5, 6, **4**, **2**, 7, 3]

$\prod_Y$: [0, 1, 5, 6, **2**, **4**, 7, 3]

$Z(\prod_Y) - Z(\prod_X) = (4)(21\text{-}17) + (3)(29\text{-}4) + (2)(8\text{-}10) = 87\ (>0)$

There is no action on the current sequence.

Step 6: (Apply Lemma 4a.)

$\prod_X$: [0, 1, 5, 6, 4, **2**, **7**, 3]

$\prod_Y$: [0, 1, 5, 6, 4, **7**, **2**, 3]

$Z(\prod_Y) - Z(\prod_X) = (3)(8\text{-}4) + (2)(7\text{-}10) + (1)(24\text{-}13) = 17\ (>0)$

We do not exchange the job 2 and job 7.

Step 7: (Apply Lemma 4a.)

$\prod_X$: [0, 1, 5, 6, 4, 2, **7**, **3**]

$\prod_Y$: [0, 1, 5, 6, 4, 2, **3**, **7**]

$Z(\prod_Y) - Z(\prod_X) = (2)(24\text{-}10) + (1)(13\text{-}13) = 28\ (>0)$

There is no action on the current sequence.

Iteration 2:

Step 1: (Apply Lemma 1b.)

$\prod_X$: [**0**, 1, **5**, 6, 4, 2, 7, 3]

$\prod_Y$: [**5**, 1, **0**, 6, 4, 2, 7, 3]

$Z(\prod_Y) - Z(\prod_X) = (1)(9\text{-}27)+(2)(25\text{-}17)+(3)(14\text{-}6) = 22\ (>0)$

Because $\prod_Y$ is not better than $\prod_X$, we don't exchange job 0 and job 5.

Step 2: (Apply Lemma 2b.)

$\prod_X$: [0, **1**, 5, **6**, 4, 2, 7, 3]

$\prod_Y$: [0, **6**, 5, **1**, 4, 2, 7, 3]

$Z(\prod_Y) - Z(\prod_X) = (1)(14\text{-}27) + (2)(19\text{-}17) + (3)(9\text{-}6) + (5)(9\text{-}17) = \text{-}40\ (<0)$

Because $\prod_Y$ is better than $\prod_X$, job 1 and job 6 are swapped, and the new sequence is [0, 6, 5, 1, 4, 2, 7, 3].

Step 3: (Apply Lemma 5.)

$\prod_X$: [0, 6, **5**, 1, **4**, 2, 7, 3]

$\prod_Y$: [0, 6, **4**, 1, **5**, 2, 7, 3]

$Z(\prod_Y) - Z(\prod_X) = (2)(17\text{-}19) + (3)(13\text{-}9) + (4)(17\text{-}9) + (3)(23\text{-}4) = 97\ (>0)$

We do nothing on the current sequence.

Step 4: (Apply Lemma 3b.)

$\prod_X$: [0, 6, 5, **1**, 4, **2**, 7, 3]

$\prod_Y$: [0, 6, 5, **2**, 4, **1**, 7, 3]

$Z(\prod_Y) - Z(\prod_X) = (3)(23\text{-}9) + (4)(29\text{-}9) + (3)(13\text{-}4) + (2)(22\text{-}10) = 173\ (>0)$

Thus, it is not necessary to change the current schedule.

Step 5: (Apply Lemma 4b.)

$\prod_X$: [0, 6, 5, 1, **4**, 2, **7**, 3]

$\prod_Y$: [0, 6, 5, 2, **7**, 1, **4**, 3]

$Z(\prod_Y) - Z(\prod_X) = (4)(22\text{-}9) + (3)(7\text{-}4) + (2)(29\text{-}10) + (1)(16\text{-}13) = 102\ (>0)$

There is no action on the current sequence.

Step 6: (Apply Lemma 4b.)

$\prod_X$: [0, 6, 5, 1, 4, **2**, 7, **3**]

$\prod_Y$: [0, 6, 5, 1, 4, **3**, 7, **2**]

$Z(\prod_Y) - Z(\prod_X) = (3)(16\text{-}4) + (2)(13\text{-}10) + (1)(7\text{-}13) = 36\ (>0)$

Because the difference is greater than 0, the schedule remains the same.

In the iteration 3, the first job may exchange with the fourth job, the second job might be swapped by the fifth job, and so on. Because the procedure is the same, the remaining procedure is the same in iteration 2. After we describe the procedures,

V. EXPERIMENT RESULTS

The testing instances are designed by Rabadi et al.[16]and the job size of each instance includes 10, 15, 20, and 25. The property of the processing time range contains low, median, and high. Because each combination has 15 similar instances, the total number of instance is 180 (4*3*15). Finally, the proposed algorithm GADP is compared with Simple Genetic Algorithm. The stopping criterion of SGA and GADP is to have examined 100,000 solutions. Because the first phase is used to construct initial solutions for GA, there are 100 solutions examined at the first phase. To compare the performance of these algorithms, the research employs the average relative error ratio, which is $(avgObj - Opt)/Opt * 100\%$ while the avgObj is the average objective value and the Opt solution is obtained from literature. The table 2 is the empirical results of this experiment, which includes some selected instances. The complete result table is available on our website at http://ppc.iem.yzu.edu.tw/publication/sourceCodes/singleMachineWithSetupTime/. Then, table 3 shows the average relative error ratio of all the 180 instances for each algorithm.

The table 2 and table 3 show GADP is completely superior to SGA for all instances in average. Moreover, the total relative average error ratio of SGA and GADP are 12.748% and 7.917% respectively. There is only one exception that SGA is better than GADP in the first instance of job size is 10 and type is high at table 2.

Table 2.
The experimental results for the three different algorithms (partial instance)

				First Phase			SGA			GADP		
Type	*Size*	*k*	Opt	Min	Mean	Max	Min	Mean	Max	Min	Mean	Max
Low	10	1	423	433	442.2	462	423	436.42	467	423	423.67	443
	10	2	378	392	405.6	421	378	378.97	393	378	378	378
	10	3	384	387	391.73	398	384	393.13	410	384	387.07	392
	15	1	801	950	990.9	1032	805	852.45	914	801	837.83	876
	15	2	794	916	983.67	1011	794	856.48	934	794	821.63	854
	15	3	753	887	928.2	975	753	798.39	870	753	775.33	837
	20	1	1293	1683	1835.1	1942	1312	1391.3	1478	1310	1375.8	1458
	20	2	1306	1723	1828.6	1894	1320	1427	1528	1312	1365.7	1425
	20	3	1299	1752	1838.2	1910	1363	1444.6	1607	1312	1387.5	1490
	25	1	1830	2694	2879	2997	1968	2076.7	2210	1900	1990.6	2085
	25	2	1828	2758	2979.2	3150	1874	2051.3	2252	1864	2005.5	2173
	25	3	1903	2900	3034.1	3145	1996	2154.4	2380	1990	2088.4	2187
Med	10	1	372	375	394.77	430	372	398.19	462	372	389	406
	10	2	510	513	539	571	510	518.55	534	510	515.23	529
	10	3	495	495	518.17	551	495	512.61	542	495	502.23	526
	15	1	982	1155	1246	1347	982	1073.5	1219	982	1034	1125
	15	2	949	1124	1261.2	1364	949	1064.7	1249	949	1023	1140
	15	3	837	953	1061.1	1140	837	907.7	1048	837	862.1	921
	20	1	1732	2308	2551.2	2690	1785	1947.4	2154	1801	1883.5	2009
	20	2	1499	2194	2356.9	2504	1599	1749.2	1985	1539	1684.3	1864
	20	3	1484	2020	2192.7	2341	1558	1697.2	1946	1496	1614.4	1801
	25	1	2149	3381	3687.6	3872	2436	2747.3	3094	2358	2548.8	2742
	25	2	2293	3506	3865.7	4119	2451	2818.7	3293	2450	2656.2	2845
	25	3	2271	3504	3847	4045	2460	2826	3225	2403	2632.8	2884
High	10	1	710	740	745.07	764	710	720.32	728	710	722.27	728
	10	2	606	606	644.7	758	606	643.35	753	606	606	606
	10	3	508	508	517.7	551	508	519.42	580	508	512.8	523
	15	1	990	1212	1351.5	1446	996	1145.5	1448	993	1099.1	1192
	15	2	1346	1700	1793.6	1905	1346	1440.2	1588	1350	1438.2	1611
	15	3	1012	1142	1317.3	1466	1012	1220.1	1475	1012	1086.7	1156
	20	1	1664	2296	2651.1	2924	1792	2087.6	2380	1760	1941.3	2227
	20	2	1505	2133	2518.4	2780	1711	1998.2	2371	1569	1785.5	2065
	20	3	1654	2288	2676.7	2953	1805	2111.9	2376	1740	1965.9	2259
	25	1	2493	3849	4211.3	4485	2583	3037.5	3513	2649	2892.4	3094
	25	2	2772	4415	4887.4	5256	2901	3499	4045	2994	3303.8	3666
	25	3	2537	4124	4640.9	5134	2845	3376.4	3894	2742	3134.2	3528

Table 3.
The average relative error ratio for the three algorithms (%)

Type	Size	First Phase	SGA	GADP
Low	10	4.32	2.07	0.3117
	15	24.345	6.177	3.217
	20	43.821	10.636	7.055
	25	59.314	13.67	9.74
Median	10	4.941	2.983	1.007
	15	30.078	10.367	5.075
	20	50.933	16.083	10.281
	25	70.427	22.553	15.5
High	10	7.46	3.408	0.662
	15	33.975	13.73	7.067
	20	58.63	23.47	14.635
	25	78.99	27.83	20.454

An ANOVA test is done, which shows that there is a significant difference among the three algorithms. Table 4 shows Duncan group result that examines the pair-wise relationship between three of them. It indicates that the GADP is the best and SGA is second while first phase is the worst. Though first phase sounds not good enough, it is because the first phase is designed to generate a population of initial solution for genetic algorithm and it examines only 100 solutions.

To show the convergence process for these algorithms, i.e., SGA and GADP, instance of job 25 and type is high that is applied as a demonstration. Figure 3 shows a single run result that GADP has the quickest convergence than SGA.

Table 4.
The Duncan grouping result for the three algorithms in mean

Duncan Grouping	Mean	N	Method
A	1961.921	5400	First Phase
B	1513.179	5400	SGA
C	1439.981	5400	GADP

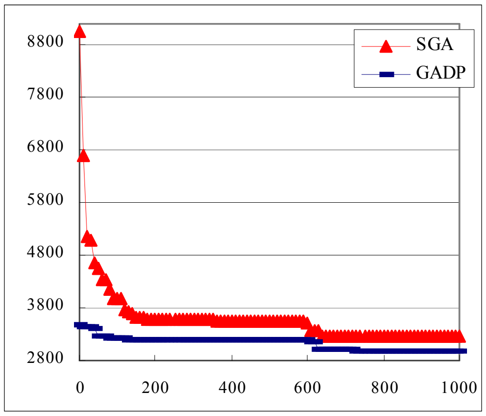


Fig. 3. The convergence diagram of the GADP and SGA in first instance of 25 jobs set with high range processing time

VI. DISCUSSION AND CONCLUSIONS

This paper examined the problem of scheduling a single machine with sequence dependent setup times to minimize the total tardiness, and some dominance properties are developed for an optimal schedule. These dominance properties determine relationship between a pair of jobs mathematically. In addition, these dominance properties apply the general pair-wise interchange and this method is further integrated with genetic algorithm, which is called GADP. From the experimental results, the first phase is able to construct better initial solutions so that while GA applies these initial solutions, it enables genetic algorithm to quickly converge to optimal solution or near optimal solution. So the hybrid framework, GADP, works more efficiently than a simple genetic algorithm alone. This is an important problem that is encountered in a wide variety of practical situations. For future research, these dominance properties can be integrated into a branch-and-bound algorithm which can further reduce the number of nodes to be branched. Thus a more efficient algorithm can be developed.

REFERENCES

[1] Alidaee B, Dragan I. “ A note on minimizing the weighted sum of tardy and early completion penalties in a single machine: a case of small common due date.” European Journal of Operational Research 1997;96:559–63.

[2] Allahverdi A, Gupta JND, Aldowaisan T. “ A review of scheduling research involving setup consideration.” OMEGA1999;27(2):219–39.

[3] Azizoglu M, Webster S. “ Scheduling job families about an unrestricted common due date on a single machine.” International Journal of Production Research 1997;35(5):1321–30.

[4] Bagchi U, Sullivan R, Chang Y-L. “ Minimizing mean absolute deviation of completion times about a common due date.” Naval Research Logistics Quarterly 1986;33:227–40.

[5] Baker KR, Scudder GD. “ Sequencing with earliness and tardiness penalties: a review.” Operations Research 1990;38(1):22–36.

[6] Chen Z-L. “ Scheduling with batch setup times and earliness-tardiness penalties.” European Journal of Operational Research 1997;96:518–37.

[7] Cheng, T.C.E., 1992. “ Optimal single-machine sequencing and assignment of common due-dates.” Computers and Industrial Engineering 1992 ; 22, 115-120.

[8] Coleman BJ. “A simple model for optimizing the single machine early/tardy problem with sequence-dependent setups. Production and Operation Management 1992;1:225–8.

[9] French S. Sequencing and scheduling: an introduction to the mathematics of the job-shop. New York: Wiley, 1982.

[10] Gordon, V., Proth, J.M., and Chu, C. “ A survey of the state-of-the-art of common due date assignment and scheduling research. ” European Journal of Operational research 2002; 139, 1-25.

[11] Hall NG, Kubiak W, Sethi SP. “Earliness-tardiness scheduling problems, II: deviation of completion times about a restrictive common due date.”Operations Research 1991;39(5):847–56.

[12] Hall NG, Posner ME. “Earliness-tardiness scheduling problems, I: weighted deviation of completion times about a common due date.” Operations Research 1991;39(5):836–46.

[13] Kanet JJ. “ Minimizing the average deviation of job completion times about a common due date.” Naval Research Logistics 1981;28:643–51.

[14] Mondal SA, Sen AK.“Single machine weighted earliness-tardiness penalty problem with a common due date.” Computers and Operation Research 2001;28(7):649–69.

[15] Ow, P. S., and Morton, E. T. “The single machine early/tardy problem.” *Management Science* 35 ,1989.pp 177—191.

[16] Rabadi, G. Mollaghasemi, M. and Anagnostopoulos, G.C, “ A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time. ” *Computers & Operations Research Journal*,2004; 31,pp. 1727-1731.

[17] Su, L.H. and Chang, P.C. “A Heuristic to Minimize A Quadratic Function of Job Lateness on A Single Machine,” *International Journal of Production Economics*, Vol. 55, No. 2, 1998,pp. 169-175.

[18] Su, L.H. and Chang, P.C.,”Scheduling n jobs on one machine to minimize the maximum lateness with a minimum number of tardy jobs,” *Computers and Industrial Engineering*, Vol.40, No.4, 2001.pp.349-360.

[19] Szwarc W. “Sngle machine scheduling to minimize absolute deviation of completion times from a common due date.”Naval Research Logistics 1989;36:663–73.

[20] Szwarc W. “The weighted common due date single machine scheduling problem revisited. Computers and Operations Research” 1996;23(3):255–62.

[21] Wu, S.D., Storer, R.H. and Chang, P.C.," One Machine Rescheduling Heuristic With Efficiency and Stability as Criteria," *Computers and Operations Research*, Vol. 20, No. 1, 1993.pp. 1-14.