International Journal of Innovative Computing, Information and Control
Volume **7**, Number **4**, 4 **2011**
ICIC International ©2010 ISSN 1349-4198
pp. 1–23

A GENETIC ALGORITHM ENHANCED BY DOMINANCE PROPERTIES FOR SINGLE MACHINE SCHEDULING PROBLEMS WITH SETUP COSTS

Pei-Chann Chang

Department of Information Management
Yuan-Ze University
135 Yuan-Dong Rd., Taoyuan 32026, Taiwan, R.O.C
iepchang@saturn.yzu.edu.tw

Shih-Hsin Chen

Department of Electronic Commerce Management
Nanhua University
No. 32, Chungkeng, Dalin, Chiayi 62248, Taiwan R.O.C.
shihhsin@mail.nhu.edu.tw

Ting Lie

Department of Information Management
Yuan-Ze University
135 Yuan-Dong Rd., Taoyuan 32026, Taiwan, R.O.C
tinglie@saturn.yzu.edu.tw

Julie Yu-Chih Liu

Department of Information Management
Yuan-Ze University
135 Yuan-Dong Rd., Taoyuan 32026, Taiwan, R.O.C
imyuchih@saturn.yzu.edu.tw

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. In this research, dominance properties are developed by swapping the neighborhood jobs. The time complexity of the dominance properties is in $O(n^2)$ and it is very efficient when combined with the GA. To prevent earlier convergence of a Simple Genetic Algorithm (SGA), these dominance properties are further embedded in SGA to improve the efficiency and effectiveness of the global searching procedure. Analytical results in benchmark problems are presented and the computational algorithms are developed.

1. Introduction.

Single-machine scheduling problems are one of the well-studied problems by many researchers. The application of single machine scheduling with setups can be found in minimizing the cycle time for pick and place (PAP) operations in Printed Circuit Board manufacturing company [24]; in a steel wire factory in China [22] and a sequencing problem in the weaving industry [2]. The results developed in the literature not only provide the insights into the single machine problem but also for more complicated environment such as flow shop or job shop.

The problem considered in this paper is to schedule a set of n jobs $\{j_1, j_2, \cdots, j_n\}$ on a single machine that is capable of processing only one job at a time without preemption. As explained in [6], and [30], all jobs are available at time zero, and a job j requires a processing time P_j. Job j belongs to a group $g_j \in \{1, \ldots, q\}$ (with $q \leq n$). Setup or changeover times, which are given as two $q \times q$ matrices, are associated to these groups. This means that in a schedule where j_j is processed immediately after j_i where $i, j \in \{1, 2, \cdots, n\}$, there must be a setup time of at least S_{ij} time units between the completion time of j_i, denoted by C_i, and the start time of j_j , which is $C_j - P_j$. During this setup period, no other task can be performed by the machine and we assume that the cost of the setup operation is $c\,(g_i; g_j) \geq 0$ and let it be equal to Machine setup time S_{ij} which is included as sequence dependent. 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 denoted as $E_j = \max(0, d - C_j)$ and its tardiness as $T_j = \max(C_j - d, 0)$, 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

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

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 a 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 [28], [31], [32] and [34]. The set of jobs is assumed to be ready for processing at the beginning which is a characteristic of the deterministic problem. 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 in [25]. Consequently, the early/tardy problem is also a strong NP-hard problem.

The single-machine E/T problem was first introduced by [23]. Since then many researchers worked on various extensions of the problem. Baker and Scudder [6] published a comprehensive state-of-the-art review for different versions of the E/T problem. Kanet [23] 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 et al. [18] extended Kanet's work and developed an algorithm that finds a set of optimal solutions for the problem based on some optimality conditions. Hall and Posner [19] solved the weighted version of the problem with no setup times. Azizoglu and Webster [4] introduced a Branch-and-Bound algorithm 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 due date (see for example [1], [5], [14], [19], [26], and [27]). Other interesting applications of scheduling problems with intelligent approaches can also be found in [20], [21], [29], [30], and [33].

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) in [16], which is also NP-hard [25]. Coleman [15] 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 for a small number of jobs. Chen [13] 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 unrestrictedly large. Allahverdi et al. [3] reviewed the scheduling literature that involved setup times. In their review, very few papers addressed the E/T problem with setup times, and no paper tackled the problem addressed in this research with the development of dominance properties. Application of Genetic Algorithm (GA) in various scheduling problems can be referred in [7, 8, 9,10, 11 and 12], however, as observed by most researchers, the simple GA will be trapped into local optimality in the earlier stages and cannot be converged into global optimal in most of the cases. The problems with the steady states GAs having premature convergence led to the desire to further improve the convergence of the algorithm. Therefore, in this research dominance properties are developed according to the sequence swapping of two neighborhood jobs and these dominance properties are further embedded in the Simple Genetic Algorithm to improve the efficiency and effectiveness of the global searching procedure. The time complexity of the dominance properties is in $O(n^2)$ and it is very efficient when combined with the GA.

2. **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 in [17]. It is shown in [23] 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 common due-date (k^*) is the sum of processing times of jobs in the first b positions in 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 groups that are early and tardy which are at position from 1 to b and b+1 to n respectively. The following notations are employed in the latter section.

$[j]$: job in position j

A: the job set of tardy jobs

B: the job set of early jobs

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

b: the median position

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

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

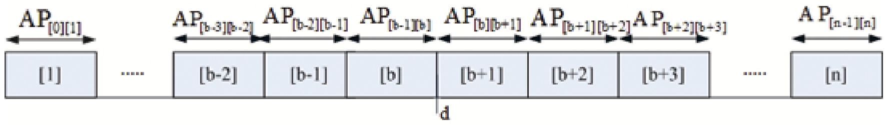
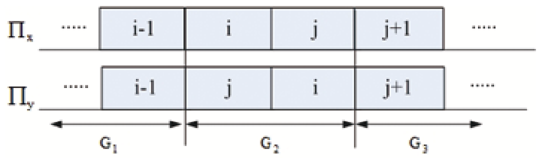
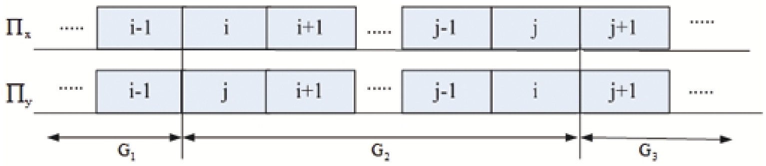


FIGURE 1. The total earliness and total tardiness for a pre-assigned due-date d



(a) Adjacent interchange



(b) Nonadjacent interchange

FIGURE 2. Two different types of interchanging methods

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

where
TT: Total tardiness for a job sequence
TE: Total earliness for a job sequence

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

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

3. **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 for the schedule $\prod$. To develop these dominance properties, we will consider interchanging two adjacent jobs and nonadjacent jobs in 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.

Thus, there are two schedules, i.e., $\prod_X$ for schedule X and $\prod_Y$ for the modified schedule Y. The corresponding objective functions of $\prod_X$ and $\prod_Y$, i.e., $Z(\prod_X)$ and $Z(\prod_Y)$, are listed as follows:

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

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

where

1. G_1: the objective of job(s) before job i
2. G_2: the objective between job i and job j
3. G_3: the objective of job(s) after job j
4. G_1': the objective job(s) before job j
5. G_2': the objective between job j and job i
6. G_3': the objective of job(s) after job i

We compare schedules $\prod_X$ and $\prod_Y$ by finding the conditions under which $\prod_X$ is better than $\prod_Y$. For a pair of jobs, i.e., job i and job j in a schedule, no matter for adjacent interchange or nonadjacent interchange, they are 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. t: Job i is tardy and job j is tardy

Because the objective values of a schedule with adjacent or nonadjacent interchange are different, there are totally 8 conditions corresponding to these two types of exchanges. Other than the cases discussed above, there is one extra case to be discussed in nonadjacent interchange which is the following:

1. Job i is early and job j is tardy

According to the cases discussed above, there are four dominance properties for the adjacent interchange which are explained at section 3.1 and five dominance properties for the nonadjacent interchange which are shown at section 3.2.

3.1. **Dominance Properties for Adjacent Interchange.** When we exchange two adjacent jobs as shown in Figure 3, the objective values of related jobs in position i, i+1, and i+2 are changed while the others are still the same. These objective terms in position i, i+1, and i+2 are different. Consequently, when we subtract $Z(\prod_Y)$ from $Z(\prod_X)$, redundant terms are reduced.

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

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

Proof:
Figure 3 shows the relationships among these jobs.

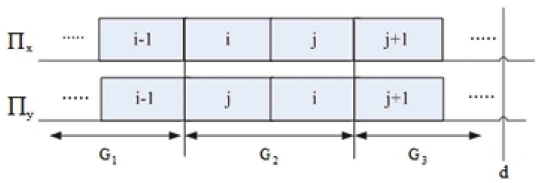


FIGURE 3. Swapping job i and job j when both of them are adjacent and early.

The difference of the objective between $Z(\prod_X)$ and $Z(\prod_Y)$ are shown as follows:

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

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

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

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

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

To derive the condition under which $Z(\prod_X) \geq Z(\prod_Y)$, the value of $Z(\prod_Y) - Z(\prod_X)$ is calculated. Let X = $Z(\prod_Y) - Z(\prod_X)$ and is equal to$(G_2' - G_2) + (G_3' - G_3)$. From the above expression, we can derive the following:

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

Therefore $X \leq 0$, the schedule $\prod_Y$ is better than schedule $\prod_X$; i.e., $Z(\prod_Y) < Z(\prod_X)$. Then, job j should be scheduled before job i.

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

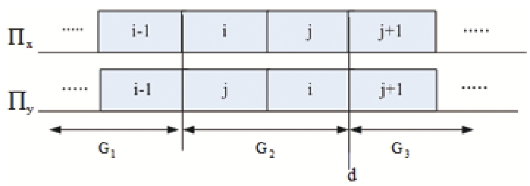


FIGURE 4. Swapping job i and job j when one job is on-time and the other is early.

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

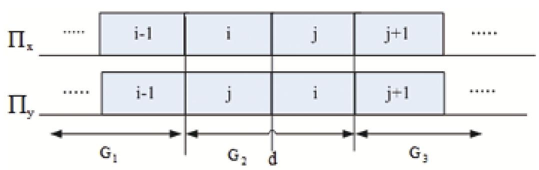


FIGURE 5. Swapping job i and job j when one job is on-time and the other is tardy.

Lemma 4a. In a given schedule $\prod_X$, for any two adjacent 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 $(n-i+1)(AP_{[i-1][j]})+(n-j+1)(AP_{[j][i]})+(n-j)(AP_{[i][j+1]}) \leq (n-i+1)(AP_{[i-1][i]})+(n-j+1)(AP_{[i][j]})+(n-j)(AP_{[j][j+1]})$

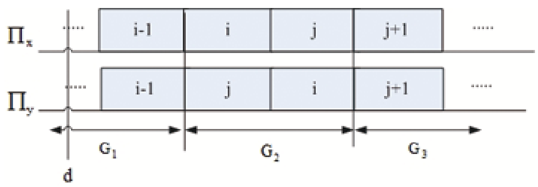


FIGURE 6. Swapping job i and job j when both of them are tardy and nonadjacent.

Lemmas discussed above are the properties for adjacent exchange between any two jobs. The next section considers the dominance properties for any two jobs which are not adjacent.

3.2. **Dominance Properties for Nonadjacent Interchange.** If the pair of jobs are nonadjacent, the jobs to be considered will be in positions i, i+1, k, and k +1. Therefore, when compared with the adjacent neighborhood interchange, there is an extra term in the objective function, i.e., when we compare the $Z(\prod_X)$ with $Z(\prod_Y)$.
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

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

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

Proof.
Figure 7 shows the relationship among these jobs.

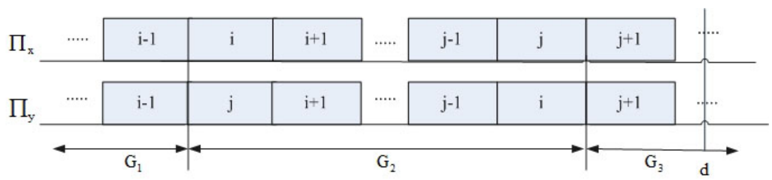


FIGURE 7. Swapping nonadjacent job i and job j when both of them are early.

The difference between $Z(\prod_X)$ and $Z(\prod_Y)$ are shown as follows:

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

$$G_2 = (i-1)AP_{[i-1][i]} + i{*}AP_{[i][i+1]} + \sum_{k=i+2}^{j-1}(k-1)AP_{[k-1][k]} + (j-1)AP_{[j-1][j]}$$

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

$$G_1' = G_1$$

$$G_2' = (j-1)AP_{[i-1][j]} + j{*}AP_{[j][i+1]} + \sum_{k=j+2}^{i-1}(k-1)AP_{[k-1][k]} + (i-1)AP_{[j-1][i]}$$

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

Let X = $Z(\prod_Y) - Z(\prod_X)$ and if X < 0, then the following condition hold:

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

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

Therefore, job i and job j are interchanged.

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

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

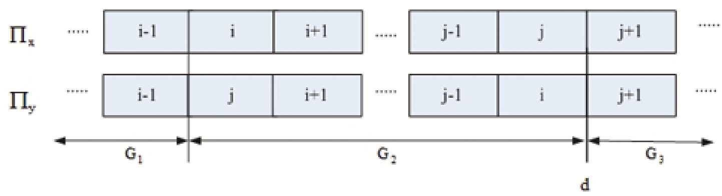


FIGURE 8. Swapping nonadjacent job i and job j when one job is on-time and the other is early.

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

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

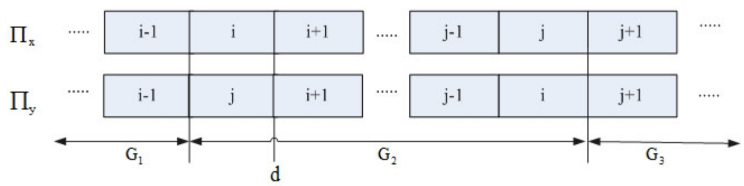


FIGURE 9. Swapping nonadjacent job i and job j when one job is on-time and the other is tardy.

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

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

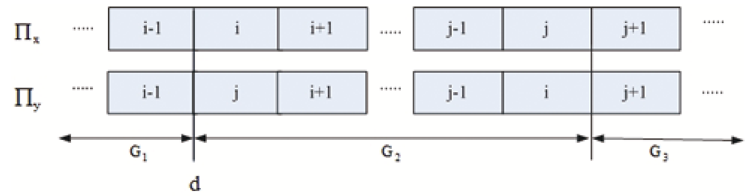


FIGURE 10. Swapping nonadjacent job i and job j when both of them are tardy.

$$(n-i+1)(AP_{[i-1][i]})+(n-i)(AP_{[i][i+1]})+(n-j+1)(AP_{[j-1][j]})+(n-j)(AP_{[j][j+1]}).$$

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

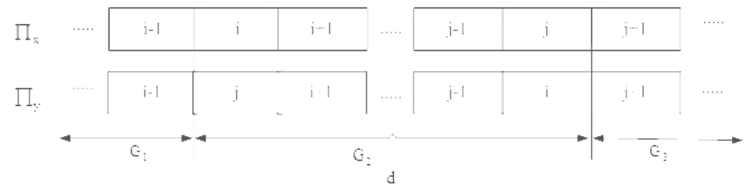


FIGURE 11. Swapping nonadjacent job i and job j when one job is early and the other is tardy.

4. **IMPLEMENTATION OF GENETIC ALGORITHM WITH DOMINANCE PROPERTIES.** Dominance properties for the single machine problem have been developed in this study and these DPs can work alone as a heuristic or to be integrated with meta-heuristic. According to our preliminary experiments, the stand alone heuristic adopting DPs explores the solution space effectively in an efficient way. However, this stand alone heuristic will be stuck in local optimal easily. This paper makes an attempt to combine the dominance properties with a meta-heuristic, i.e., Genetic Algorithm. Therefore, a two-phase hybrid algorithm is proposed and it is named genetic algorithm with dominance properties, i.e., GADP in short. The detailed procedures of GADP are explained at section 4.1 and section 4.2, respectively.

4.1. **The First Phase of GADP.** The first phase is to establish the initial solutions by employing dominance properties developed above. Given a set of random generated solutions, a set of initial solutions can be derived by applying these DPs to each sequence. Since the scheduling problem is a sequential problem, path-representation will be adopted as an encoding technique. The following figure shows an eight-job example for this encoding representation. This encoding method is applied in phase 2 as well.

After the random solution is generated, the heuristic applies a general pair-wise interchange (GPI) which is a neighborhood search method to exploit the solution space. The GPI procedure will pick two jobs randomly to swap and then evaluate the performance of the new schedule based on the dominance properties. If the new solution is better than the original one, the new one will replace the original solution. The process will continue until all jobs have been interchanged.

For a given sequence, an initial solution is obtained by applying GPI and DPs. Therefore, a set of initial solutions can be generated by using the heuristic iteratively in the first phase. The time-complexity of the first phase is $O(n^2)$ and the set of solutions generated are employed in the second phase by the genetic algorithm. The pseudo code of the main procedure and the first phase are demonstrated as the following:

Notation:

- *Population*: A set of solutions represent the chromosomes in genetic algorithm.
- n: The population size.

Algorithm 1: Main ()

```
initializePopulationSize()
for i = 1 to n do
    GPI(Population[i])
end for
Genetic Algorithm() //The second phase
```

Algorithm 2: GPI()

```
sequence = generateRandomSolution()
for i = 1 to n do
    for increment = 1 to 3 do
        for pos = 0 to n − increment do
            dominanceProperty(sequence, pos, pos + increment)
        end for
        if sequence has not been changed then
            break;
        end if
    end for
    return sequence
end for
```

4.2. **The Second Phase of GADP.** In the second phase, GA will be applied to further improve the solution quality. The pseudo code of the genetic algorithm are listed as follows:

Algorithm 3: Genetic Algorithm())

```
Adopt the solutions from Phase 1()
counter ← 0
while counter < maxGeneration do
    Evaluate Fitness()
    Elitism()
    Selection()
    Crossover()
    Mutation()
    counter ← counter+1
end while
```

The genetic operators applied in the Genetic Algorithm including the selection, crossover, and mutation operator will be explained in the following section.

4.2.1. *Fitness and Selection Operator.* Because the single machine scheduling with setups is a single objective problem, the objective value of each chromosome can be used as fitness directly. Then, the binary tournament selection is employed in the selection operation. The criterion to select better offspring is depended on their own fitness; the individual whose fitness is better will be selected. As a result, the selection procedure selects better chromosomes into the mating pool.

4.2.2. *Crossover Operator.* The crossover procedure is randomly selecting two chromosomes to mate. There are several crossover methods for combinatorial problem. This study employed the two-point crossover and the procedures of the two-point crossover are listed as follows:

1. Select two chromosomes and named it as parent 1 and parent 2.
2. Determine the two cut points, suppose they are at position i and j, copy the genes which outside the range from i to j to the offspring in the same position.
3. Copy the remaining genes which inside the range of parent1 in the order of relative gene position of parent 2.

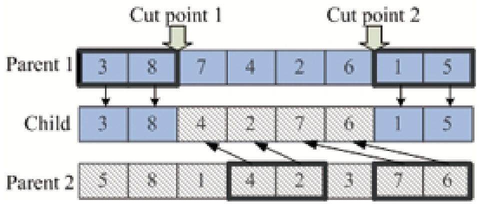


FIGURE 12. Two-point crossover

4.2.3. *Mutation.* The purpose of mutation is to generate a new chromosome with a better fitness by changing the gene position of the current chromosome. Swap mutation is applied here because it is easy to implement by setting two positions and exchanging the two values of these positions.

5. **EXPERIMENTAL RESULTS.** The bench mark test will base on the instances designed by [30] and the job size of each instance includes 10, 15, 20, and 25. The range of the processing time contains low, median, and high, which are based on the generation functions of Uniform(10, 60), Uniform(10, 110), and Uniform(10, 160), accordingly. Because each combination has 15 similar instances, the total number of instances is 180 (4*3*15) and each instance is replicated 30 times for each algorithm tested. This study utilized the design-of-experiment (DOE) to select the best parameter setting of GA. Table 2 shows the result generated by the DOE experiments. The proposed algorithm is to improve the effectiveness of the GA approach. Therefore, GADP is compared with the original GA and DP approaches to demonstrate its effectiveness. These experimental results are shown at section 5.1.

Sourd [30] only provided the instances of 10, 15, 20, and 25 jobs and these instances might not be sufficient to demonstrate the complexity of the problem. Consequently, we apply similar concept by Sourd [30] and generate large size of problems, which include 50, 100, 150, and 200 jobs. The distribution of these instances is also based on the processing time range that includes low, median, and high. Therefore, there are totally 180 combinations in these large size instances as well.

TABLE 1. GA parameters setting

Factor	Default
Crossover Rate	06
Mutation Rate	0.5
Population Size	100
Generations	1000

5.1. **The Small Size Problems.** The stopping criterion of SGA and GADP is to examine 100,000 solutions. Because the first phase is used to construct initial solutions for GA, there are totally 100 initial solutions generated at the first phase. To compare the performance of these algorithms, the research employs the average relative error ratio, which is $((avgObj - Opt)/Opt) * 100$ where the $avgObj$ is the average objective value and the Opt solution is obtained from literature. Table 3 is the empirical results of this experiment, which includes some selected instances. Because there are 15 combinations of each instance type, they are denoted as k in table 3. Owing to there are 180 combinations, it is not possible to demonstrate all the empirical results. This study selects partial results of k from 1 to 3. The complete results of these tests are available on our website[1]. Finally, the optimal solution is available by [30] who applied Branch-and-Bound algorithm to derive the solution.

Then, Table 3 shows the average relative error ratio of all the 180 instances for each algorithm tested. Table 2 and Table 3 shows GADP is totally 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. The instance is job size 10 and the type is high at Table 3.

An ANOVA test is applied to show if there is a significant difference among these three algorithms. Table 4 shows the Duncan grouping result that examines the pair-wise relationship among these three algorithms tested. The Duncan test shows that GADP is the best and SGA is the second. DP only performs the worst.

To show the convergence process for these algorithms, i.e., DP, SGA and GADP, instance of job 25 with high variation of job processing time is applied as a demonstration. It shows that GADP is significantly outperform DP and SGA because these three algorithms do not share the same alphabet in Duncan test. As a result, GADP performs the best in solving the single machine scheduling problem with setup cost.

5.2. **The Large Size Problems.** This study designs larger size instances for this scheduling problem and these experimental results are shown in the section. The parameter settings of genetic algorithm are the same as in section 5.1. The testing instances and the complete result table are available on our website. Table 5 represents the empirical result of SGA and GADP under different distributions of processing times and problem sizes. According to the results in Table 6, the average performance of GADP is better than SGA and the differences are very obvious. Finally, Table 6 show the result of Duncan test between the two algorithms and GADP is statistically better than SGA.

Finally, although it is not possible to obtain optimal solutions for large size of instances in a limited time, this study ran the GADP for 1,000,000 solutions for each instance to derive a near-optimal solutions. Then, we ran SGA and GADP for 100,000 solutions to derive the current best objective value for each instance. The average relative error ratio$((avgObj - currentMin)/currentMin) * 100$ is applied to distinguish the performance of SGA and GADP. The result is shown at Table 7. We can find out that GADP outperforms the other two methods for all different instances.

6. **DISCUSSIONS AND CONCLUSIONS.** This research studied the single machine scheduling problem with sequence dependent setup times and the objective is to minimize the total tardiness. This is a very important problem that is encountered in a wide variety of practical situations. A set of dominance properties are developed in this research to determine the relationship between a pair of jobs. The time complexity of the dominance properties is in $O(n^2)$ and it is very efficient when combined with the GA. To speed

[1] http://mail.nhu.edu.tw/~shihhsin/download/

TABLE 2. The experimental results for three different algorithms compared (partial instance

				DP			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

up the convergence of GA, these dominance properties are further integrated with a genetic algorithm which is named GADP in short. From the experimental results, these dominance properties are able to generate a set of very good initial solutions and the GA procedures can further evolve these solutions into near-optimal solutions. The solution quality of GADP is much better than that of a simple GA. It can be concluded that these dominance properties are very effective in generating good quality of initial solutions. Therefore, GADP is more efficient and effective when compared with a simple GA. For a

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

Type	Size	DP	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

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

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

set of large instances such as 150, 200 or even larger job sizes, GADP still performs the best among others.

TABLE 5. The experimental results for the three different algorithms (partial instance

Type	Size	k	SGA Obj Value			GADP Obj Value		
			Min	Mean	Max	Min	Mean	Max
Low	50	1	8229	8794.6	9424	7397	8211.2	8767
		2	7753	8339.1	8850	7489	8012.2	8444
		3	7750	8405	9037	7666	8106.8	8561
	100	1	32914	34183	35711	30268	31168	32171
		2	32665	34477	35942	30062	31361	32694
		3	32776	34446	36296	30452	31392	32278
	150	1	78688	81818	85833	67865	70177	71874
		2	76316	80938	83605	67999	69723	71263
		3	79011	81352	85989	68478	69910	72001
	200	1	145254	150104	154839	120452	122850	125300
		2	143730	149496	154344	119157	122564	124625
		3	142487	148548	157893	120217	122368	125074
Med	50	1	9391	10491	11957	9017	9814.5	10399
		2	9308	10543	11504	8938	9852.1	10558
		3	9157	10454	11873	8991	9772.9	10741
	100	1	41240	43497	46031	35844	38266	40717
		2	41955	44270	47300	36176	38408	40083
		3	41874	43835	48138	34872	37627	39840
	150	1	97679	102975	111951	79865	83056	85680
		2	99005	105683	113085	80886	83903	86699
		3	97373	103708	110166	81139	83820	86736
	200	1	184194	197482	204554	141130	146198	150477
		2	182665	194459	200625	142235	145641	148678
		3	186634	196829	206540	142708	146056	151177
High	50	1	11577	12751	14026	10611	11673	12849
		2	11289	12572	13540	10179	11841	13886
		3	12178	13780	15899	11125	12371	13703
	100	1	48572	52023	56383	41878	44339	47249
		2	48016	53415	57755	42233	44492	47300
		3	47989	51751	55985	41059	43745	46483
	150	1	117097	126981	134396	91331	96999	102956
		2	119629	124796	129865	91850	96628	101090
		3	121124	128037	135771	89284	96857	101771
	200	1	227667	240781	254946	162533	168965	175379
		2	223495	239886	260765	162936	169169	177366
		3	228726	243118	270010	161974	170394	177203

TABLE 6. The Duncan grouping result for the three algorithms in mean

Duncan Grouping	Mean	N	method
A	88585.7	5400	SGA
B	69323.3	5400	GADP

7. REFERENCES.

Table 7. The performance relative average error ratio SGA and GADP

Type	Size	k	Current Average			Error (%)	Ratio
			Min	SGA	GADP	SGA	GADP
Low	50	1	7397	8794.6	8211.2	18.89	11.01
		2	7489	8339.1	8012.2	11.35	6.99
		3	7666	8405	8106.8	9.64	5.75
	100	1	30268	34183	31168	12.93	2.97
		2	30003	34477	31361	14.91	4.53
		3	30452	34446	31392	13.12	3.09
	150	1	67820	81818	70177	20.64	3.48
		2	66918	80938	69723	20.95	4.19
		3	66043	81352	69910	23.18	5.86
	200	1	120452	150104	122850	24.62	1.99
		2	116040	149496	122564	28.83	5.62
		3	118261	148548	122368	25.61	3.47
Med	50	1	9017	10491	9814.5	16.35	8.84
		2	8938	10543	9852.1	17.96	10.23
		3	8991	10454	9772.9	16.27	8.70
	100	1	34320	43497	38266	26.74	11.50
		2	36176	44270	38408	22.37	6.17
		3	34872	43835	37627	25.70	7.90
	150	1	77324	102975	83056	33.17	7.41
		2	80886	105683	83903	30.66	3.73
		3	78748	103708	83820	31.70	6.44
	200	1	139254	197482	146198	41.81	4.99
		2	133589	194459	145641	45.57	9.02
		3	141397	196829	146056	39.20	3.29
High	50	1	10611	12751	11673	20.17	10.01
		2	10179	12572	11841	23.51	16.33
		3	11125	13780	12371	23.87	11.20
	100	1	40560	52023	44339	28.26	9.32
		2	42233	53415	44492	26.48	5.35
		3	41059	51751	43745	26.04	6.54
	150	1	89926	126981	96999	41.21	7.87
		2	86745	124796	96628	43.87	11.39
		3	89284	128037	96857	43.40	8.48
	200	1	161560	240781	168965	49.04	4.58
		2	155609	239886	169169	54.16	8.71
		3	154185	243118	170394	57.68	10.51

1. B. Alidaee, I. Dragan, 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 vol., 96, no.3, pp.559-563, 1997.
2. M.H. Al-Haboubi and Shokri Z. Selim, A sequencing problem in the weaving industry, European Journal of Operational Research Vol.66, no. 1, pp. 65-71, 1993.
3. A. Allahverdi, J.N.D. Gupta, T. Aldowaisan, A review of scheduling research involving setup consideration, OMEGA, vol. 27, no.2, pp.219-239, 1999.

4. M. Azizoglu, S. Webster, Scheduling job families about an unrestricted common due date on a single machine, International Journal of Production Research, vol.35, no.5, pp.1321-1330, 1997.
5. U. Bagchi, R. Sullivan, Y-L. Chang, Minimizing mean absolute deviation of completion times about a common due date, Naval Research Logistics Quarterly vol. 33, no.2, pp.227-240, 1986.
6. K.R. Baker, G.D. Scudder, Sequencing with earliness and tardiness penalties: a review. Operations Research vol. 38, no.1, pp.22-36, 1990.
7. P.C. Chang, J.C. Hsieh, and Y.W. Wang, Genetic Algorithms Applied in BOPP Film Scheduling Problems, Applied Soft Computing, vol. 3, no.2, pp. 139-148, 2003.
8. P.C. Chang, J.C. Hsieh, and Y.W. Wang, Adaptive multi-objective genetic algorithms for scheduling of drilling operation in printed circuit board industry. Applied Soft Computing, vol. 7, no. 3, pp.800-806, 2007.
9. P.C. Chang, S.H. Chen and K.L. Lin, Two Phase Sub-Population Genetic Algorithm for Parallel Machine Scheduling problem, Expert Systems with Applications, vol. 29, no.3, pp.705-712, 2005.
10. P.C. Chang, J.C. Hsieh and C.H. Liu, A Case-Injected Genetic Algorithm for Single Machine Scheduling Problems with Release Time, International Journal of Production Economics, vol. 103, no.2, pp.551-564, 2006.
11. P.C. Chang, H.S. Chen and V. Mani, Parametric Analysis of Bi-criterion Single Machine Scheduling with A Learning Effect, International Journal of Innovational Computing Information and Control, vol.4, no.8, pp.2033-2043, 2008.
12. S.H. Chen, P.C. Chang, Qingfu Zhang and C.-B. Wang, A guided Memetic algorithm with probabilistic models, International Journal of Innovational Computing Information and Control, vol. 5, no.12 (B), 4753-4764, 2009.
13. Z-L. Chen, Scheduling with batch setup times and earliness-tardiness penalties, European Journal of Operational Research, vol.96, no.3, pp.518-537, 1997.
14. T.C.E. Cheng, Optimal single-machine sequencing and assignment of common due-dates, Computers and Industrial Engineering, vol.22, no.2, pp.115-120, 1992.
15. B.J. Coleman, A simple model for optimizing the single machine early/tardy problem with sequence-dependent setups, Production and Operation Management vol. 1, no.2 , pp.225-228, 1992.
16. S. French, Sequencing and scheduling: an introduction to the mathematics of the job-shop, New York, Wiley, 1982.
17. V. Gordon, J. Proth, and C. Chu, A survey of the state-of-the-art of common due date assignment and scheduling research, European Journal of Operational Research, vol.139, no.1, pp.1-25, 2002.
18. N.G. Hall, W. Kubiak, S.P. Sethi, Earliness-tardiness scheduling problems, II: deviation of completion times about a restrictive common due date, Operations Research, vol.39, no.5, pp.847-856, 1991.
19. N.G. Hall, StateM.E. Posner, Earliness-tardiness scheduling problems, I: weighted deviation of completion times about a common due date, Operations Research, vol.39, no.5, pp.836-846, 1991.
20. Y. Hirashima, A Q-learning system for container transfer scheduling based on shipping order at container terminals, International Journal of Innovative Computing, Information and Control, vol.4, no.3, pp.547-558, 2008.
21. X. Hu, M. Huang and A. Zeng, An intelligent solution system for a vehicle routing problem in urban distribution, International Journal of Innovative Computing, Information and Control, vol.3, no.1, pp.189-198, 2007.

22. F. Jin , J.N.D. Gupta , S. Song, C. Wu , Single machine scheduling with sequence-dependent family setups to minimize maximum lateness, Journal of the Operational Research Society, doi:10.1057/jors.2009.63, 2009.
23. J.J. Kanet, Minimizing the average deviation of job completion times about a common due date, Naval Research Logistics, vol.28, no.4, pp.643-651, 1981.
24. O.I. Kulak, O. Yilmaz, PCB assembly scheduling for collect-and-place machines using genetic algorithms. International Journal of Production Research, vol. 45, no. 17, pp. 3949-3969, 2007.
25. J.K. Lenstra, A.H.G. Rinnooy placeStateKan, and P. Brucker, Complexity of machine scheduling problems, Annals of Discrete Mathematics, vol.1, pp.343-362, 1977.
26. Tinghuai Ma, Qiaoqiao Yan, Donghai Guan and Sungyoung Lee, Research on Task Scheduling Algorithm in Grid Environment, ICIC Express Letters, vol.4, no.1, pp. 1-6, 2010.
27. S. A. Mondal, A.K. Sen, Single machine weighted earliness-tardiness penalty problem with a common due date, Computers & Operation Research, vol.28, no.7, pp.649-669, 2001.
28. P. S. Ow, and E.T. Morton, The single machine early/tardy problem, Management Science, vol.35, no.2, pp.177-191, 1989.
29. G. Rabadi, M. Mollaghasemi, G.C. Anagnostopoulos, A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time, Computers & Operation Research, vol.31, no.10, pp.1727-1751, 2001.
30. F. Sourd, Earliness-tardiness scheduling with setup considerations, Computers & Operations Research, vol.32, no.7, pp.1849-1865, 2005.
31. L. H. Su and P.C. Chang, A Heuristic to Minimize A Quadratic Function of Job Lateness on A Single Machine, International Journal of Production Economics, vol.55, no.2, pp.169-175, 1998.
32. L. H. Su and P.C. Chang, 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, pp.349-360, 2001.
33. J. J. Wang, F. Liu and P. He, Rescheduling under Predictive Disruption of WSPT Schedule for Single Machine Scheduling, ICIC Express Letters, vol.4, no.2, pp.467-472, 2010.
34. S.D. Wu, R.H. Storer, and P.C. Chang, One Machine Rescheduling Heuristic with Efficiency and Stability as Criteria, Computers & Operations Research, vol.20, no.1, pp.1-14, 1993.

Appendix 1: The detail proofs of dominance properties

The following is the detail proofs of the dominance properties in section 3.

1.1 Dominance Properties of Adjacent Interchange

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

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

Proof.

Figure 4 shows this condition and the objective of $Z(\prod_X)$ and $Z(\prod_Y)$.

$$\because G_1=\sum_{k=1}^{b-2}(k-1)AP_{[k-1][k]}$$

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

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

$$G_1^{'}=G_1$$

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

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

Let X $=Z(\prod_Y)-Z(\prod_X)$ and if X <0, it means $\prod_Y$ is better than $\prod_X$ which satisfies $(i-1)(AP_{[i-1][j]})+(j-1)(AP_{[j][i]})+(n-j)(AP_{[i][j+1]})\leq(i-1)(AP_{[i-1][i]})+(j-1)(AP_{[i][j]})+(n-j)(AP_{[j][j+1]})$.

So job i and job j are interchanged.

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

Proof.

Figure 5 shows this condition and the objective of $Z(\prod_X)$ and $Z(\prod_Y)$.

$$\because G_1=\sum_{k=1}^{b-1}(k-1)AP_{[k-1][k]}$$

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

$$G_3=(n-j)AP_{[j][j+1]}+\sum_{k=b+2}^{n-1}(n-k)AP_{[k][k+1]}$$

$$G_1^{'}=G_1$$

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

$$G_3' = (n-j)AP_{[i][j+1]} + \sum_{k=b+2}^{n-1}(n-k)AP_{[k][k+1]}$$

Let X = $Z(\prod_Y) - Z(\prod_X)$ and if X < 0, then the following condition hold:

$$(i-1)(AP_{[i-1][j]}) + (n-i)(AP_{[j][i]}) + (n-j)(AP_{[i][j+1]}) \leq$$
$$(i-1)(AP_{[i-1][i]}) + (n-i)(AP_{[i][j]}) + (n-j)(AP_{[j][j+1]}).$$

So job i and job j are interchanged.

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

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

Proof.

Figure 6 shows this condition and the objective of $Z(\prod_X)$ and $Z(\prod_Y)$.

$$\because G_1 = \sum_{k=1}^{b}(k-1)AP_{[k-1][k]} + \sum_{k=b}^{i-1}(n-k)AP_{[k][k+1]}$$

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

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

$$G_1' = G_1$$

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

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

Let X = $Z(\prod_Y) - Z(\prod_X)$ and if X < 0, then the following condition hold:

$$(n-i+1)(AP_{[i-1][j]}) + (n-j+1)(AP_{[j][i]}) + (n-j)(AP_{[i][j+1]}) \leq$$
$$(n-i+1)(AP_{[i-1][i]}) + (n-j+1)(AP_{[i][j]}) + (n-j)(AP_{[j][j+1]}).$$

So job i and job j are interchanged.

1.2 Dominance Properties of Non-adjacent Interchange

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

$$(i-1)(AP_{[i-1][j]}) + (i)(AP_{[j][i+1]}) + (j-1)(AP_{[j-1][i]}) + (n-j)(AP_{[i][j+1]}) \leq$$
$$(i-1)(AP_{[i-1][i]}) + (i)(AP_{[i][i+1]}) + (j-1)(AP_{[j-1][j]}) + (n-j)(AP_{[j][j+1]}).$$

Proof.

Figure 8 shows this condition and the objective of $Z(\prod_X)$ and $Z(\prod_Y)$.

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

$$G_2 = (i-1)AP_{[i-1][i]} + i{*}AP_{[i][i+1]} + \sum_{k=i+2}^{j-1}(k-1)AP_{[k-1][k]} + (j-1)AP_{[j-1][j]}$$

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

$$G_1^{'} = G_1$$

$$G_2^{'} = (j-1)AP_{[i-1][j]} + j{*}AP_{[j][i+1]} + \sum_{k=j+2}^{i-1}(k-1)AP_{[k-1][k]} + (i-1)AP_{[j-1][i]}$$

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

Let X = $Z(\prod_Y) - Z(\prod_X)$ and if X < 0, then the following condition hold: $(i-1)(AP_{[i-1][j]})+(i)(AP_{[j][i+1]})+(j-1)(AP_{[j-1][i]})+(n-j)(AP_{[i][j+1]}) \leq (i-1)(AP_{[i-1][i]})+(i)(AP_{[i][i+1]})+(j-1)(AP_{[j-1][j]})+(n-j)(AP_{[j][j+1]})$. So job i and job j are interchanged.

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

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

Proof.

Figure 9 shows this condition and the objective of $Z(\prod_X)$ and $Z(\prod_Y)$.

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

$$G_2 = (i-1)AP_{[i-1][i]} + (n-i)AP_{[i][i+1]} + \sum_{k=i+2}^{j-2}(n-k+1)AP_{[k-1][k]} + (n-j+1)AP_{[j-1][j]}$$

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

$$G_1^{'} = G_1$$

$$G_2^{'} = (j-1)AP_{[i-1][j]} + (n-j)AP_{[j][i+1]} + \sum_{k=j+2}^{i-2}(n-k+1)AP_{[k-1][k]} + (n-i+1)AP_{[j-1][i]}$$

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

Let X = $Z(\prod_Y) - Z(\prod_X)$ and if X < 0, then the following condition hold:

$$(i-1)(AP_{[i-1][j]})+(n-i)(AP_{[j][i+1]})+(n-j+1)(AP_{[j-1][i]})+(n-j)(AP_{[i][j+1]}-AP_{[j][j+1]}) \leq (i-1)(AP_{[i-1][i]})+(n-i)(AP_{[i][i+1]})+(n-j+1)(AP_{[j-1][j]})+(n-j)(AP_{[j][j+1]}).$$

So job i and job j are interchanged.

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

$$(n-i+1)(AP_{[i-1][j]})+(n-i)(AP_{[j][i+1]})+(n-j+1)(AP_{[j-1][i]})+(n-j)(AP_{[i][j+1]}) \leq (n-i+1)(AP_{[i-1][i]})+(n-i)(AP_{[i][i+1]})+(n-j+1)(AP_{[j-1][j]})+(n-j)(AP_{[j][j+1]}).$$

Proof.

Figure 10 shows this condition and the objective of $Z(\prod_X)$ and $Z(\prod_Y)$.

$$\because G_1 = \sum_{k=1}^{b}(k-1)AP_{[k-1][k]} + \sum_{k=b}^{i-1}(n-k)AP_{[k][k+1]}$$

$$G_2 = (n-i+1)AP_{[i-1][i]} + (n-i)AP_{[i][i+1]}$$

$$+ \sum_{k=i+2}^{j-1}(n-k+1)AP_{[k-1][k]} + (n-j+1)AP_{[j-1][j]}G_3 = \sum_{k=j}^{n-1}(n-k)AP_{[k][k+1]}$$

$$G_1' = G_1$$

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

$$+ \sum_{k=j+2}^{j-1}(n-k+1)AP_{[k-1][k]} + (n-i+1)AP_{[j-1][i]}G_3' = \sum_{k=i}^{n-1}(n-k)AP_{[k][k+1]}$$

Let X = $Z(\prod_Y) - Z(\prod_X)$ and if X < 0, it means $\prod_Y$ is better than $\prod_X$ which satisfies $(n-i+1)(AP_{[i-1][j]}) + (n-i)(AP_{[j][i+1]}) + (n-j+1)(AP_{[j-1][i]}) + (n-j)(AP_{[i][j+1]}) \leq (n-i+1)(AP_{[i-1][i]}) + (n-i)(AP_{[i][i+1]}) + (n-j+1)(AP_{[j-1][j]}) + (n-j)(AP_{[j][j+1]})$.

So job i and job j are swapped.

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

$$(i-1)(AP_{[i-1][j]})+(i)(AP_{[j][i+1]})+(n-j+1)(AP_{[j-1][i]})+(n-j)(AP_{[i][j+1]}) \leq (i-1)(AP_{[i-1][i]})+(i)(AP_{[i][i+1]})+(n-j+1)(AP_{[j-1][j]})+(n-j)(AP_{[j][j+1]}).$$

Proof.

Figure 11 shows this condition and the objective of $Z(\prod_X)$ and $Z(\prod_Y)$.

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

$$G_2 = (i-1)AP_{[i-1][i]} + i{*}AP_{[i][i+1]} + \textstyle\sum_{k=i+2}^{b}(k-1)AP_{[k-1][k]} + \sum_{k=b}^{j-1}(n-k)AP_{[k][k+1]} + (j-1)AP_{[j-1][j]}$$

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

$G_1' = G_1 G_2' = (j-1)AP_{[i-1][j]} + j{*}AP_{[j][i+1]} + \sum_{k=j+2}^{b}(k-1)AP_{[k-1][k]} + \sum_{k=b}^{j-1}(n-k)AP_{[k][k+1]} \quad +$

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

Let X $= Z(\prod_Y) - Z(\prod_X)$ and if X < 0, it means $\prod_Y$ is better than $\prod_X$ which satisfies $(i-1)(AP_{[i-1][j]})+(i)(AP_{[j][i+1]})+(n-j+1)(AP_{[j-1][i]})+(n-j)(AP_{[i][j+1]}) \leq (i-1)(AP_{[i-1][i]})+(i)(AP_{[i][i+1]})+(n-j+1)(AP_{[j-1][j]})+(n-j)(AP_{[j][j+1]})$. Therefore, job i and job j are exchanged.