Available online at www.sciencedirect.com
ScienceDirect
Applied Mathematical Modelling 33 (2009) 579–596
APPLIED MATHEMATICAL MODELLING
www.elsevier.com/locate/apm

A hybrid genetic algorithm with dominance properties for single machine scheduling with dependent penalties

Pei Chann Chang [a,b,*], Shih Hsin Chen [b,1], V. Mani [c]

[a] Department of Information Management, Yuan Ze University, 135, Yuan-Tung Road, 32026 Tao-Yuan, Taiwan, ROC
[b] Department of Industrial Engineering and Management, Yuan Ze University, 135, Yuan-Tung Road, 32026 Tao-Yuan, Taiwan, ROC
[c] Department of Aerospace Engineering, Indian Institute of Science, Bangalore, India

Received 11 June 2007; received in revised form 30 December 2007; accepted 4 January 2008
Available online 15 January 2008

Abstract

In this paper, a hybrid genetic algorithm is developed to solve the single machine scheduling problem with the objective to minimize the weighted sum of earliness and tardiness costs. First, dominance properties of (the conditions on) the optimal schedule are developed based on the switching of two adjacent jobs *i* and *j*. These dominance properties are only necessary conditions and not sufficient conditions for any given schedule to be optimal. Therefore, these dominance properties are further embedded in the genetic algorithm and we call it genetic algorithm with dominance properties (GADP). This GADP is a hybrid genetic algorithm. The initial populations of schedules in the genetic algorithm are generated using these dominance properties. GA can further improve the performance of these initial solutions after the evolving procedures. The performances of hybrid genetic algorithm (GADP) have been compared with simple genetic algorithm (SGA) using benchmark instances. It is shown that this hybrid genetic algorithm (GADP) performs very well when compared with DP or SGA alone.

Keywords: Single machine scheduling; Earliness/tardiness; Dominance properties; Genetic algorithm; Optimal schedule

1. Introduction

In this paper, a deterministic single machine scheduling problem without release date is investigated and the objective is to minimize the total sum of earliness and tardiness penalties. A detailed formulation of the problem is described as follows: a set of n independent jobs $\{J_1, J_2, \ldots, J_n\}$ has to be scheduled without preemptions on a single machine that can handle at most one job at a time. The machine is assumed to be continuously available from time zero onwards and unforced machine idle time is not allowed. Job $J_j, j = 1, 2, \ldots, n$ becomes available for processing at the beginning, requires a processing time p_j and should be completed on its due date d_j. For any given schedule, the earliness and tardiness of J_j can be, respectively, defined as $E_j = \max(0, d - C_j)$ and

* Corresponding author. Address: Department of Information Management, Yuan Ze University, 135, Yuan-Tung Road, 32026 Tao-Yuan, Taiwan, ROC. Tel.: +886 936101320; fax: +886 34638884.
E-mail address: iepchang@saturn.yzu.edu.tw (P.C. Chang).
[1] Current address: Department of Electronic Commerce Management, Nanhua University, 32, Chungkeng, Dalin, Chiayi 62248, Taiwan, ROC.

0307-904X/$ - see front matter
doi:10.1016/j.apm.2008.01.006

$T_j = \max(0, C_j - d)$, where C_j is the completion time of J_j. The objective is then to find a schedule that minimizes the sum of the earliness and tardiness penalties of all jobs $\sum_{j=1}^{n}(\alpha_j E_j + \beta_j T_j)$ where α_j and β_j are the earliness and tardiness penalties of job J_j. 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. It is assumed that no unforced machine idle time is allowed, so the machine is only idle if no job is currently available for processing. This assumption reflects a production setting where the cost of machine idleness is higher than the early cost incurred by completing any job before its due date, or the capacity of the machine is limited when compared with its demand, so that the machine must indeed be kept running.

Some specific examples of production settings with these characteristics are provided by Ow and Morton [1], Azizoglu et al. [2], Wu et al. [3] and Su and Chang [4,5]. 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 Lenstra et al. [6]. To the best of our knowledge, the earlier work in this problem is due to Chang and Lee [7,8], Wu et al. [3], and Chang [9]. Belouadah et al. [10] dealt with the similar problem however with a different objective in minimizing the total weighted completion time and the problem is the same as discussed in Hariri and Potts [11]. Kim and Yano [12] discussed some properties of the optimal solution, and these properties are used to develop both optimal and heuristic algorithms. Valente and Alves [13] presented a branch-and-bound algorithm based on a decomposition of the problem into weighted earliness and weighted tardiness subproblems. Two lower bound procedures were presented for each subproblem, and the lower bound for the original problem is then simply the sum of the lower bounds for the two subproblems. In Valente and Alves [14], they analyzed the performance of various heuristic procedures, including dispatch rules, a greedy procedure and a decision theory search heuristic.

The early/tardy problem with equal release dates and no idle time, however, has been considered by several authors, and both exact and heuristic approaches have been proposed. Among the exact approaches, branch-and-bound algorithms were presented by Abdul-Razaq and Potts [15], Li [16] and Liaw [17]. The lower bounding procedure of Abdul-Razaq and Potts was based on the subgradient optimization approach and the dynamic programming state-space relaxation technique, while Li and Liaw used Lagrangean relaxation and the multiplier adjustment method. Among the heuristics, Ow and Morton [18] developed several dispatch rules and a filtered beam search procedure. Valente and Alves [14] presented an additional dispatch rule and a greedy procedure, and also considered the use of dominance rules to further improve the schedule obtained by the heuristics. A neighborhood search algorithm was also presented by Li [16].

Genetic algorithm is a well-known technique and is used for many combinatorial optimization problems as in Holland [19], Goldberg [20] and David [21]. A good discussion of using genetic algorithms to problems that are encountered in production systems and operations research areas are available in Michalewicz [22]. Many researchers Chang et al. [23–25] started using genetic algorithms for scheduling problems and a survey of genetic algorithms for job-shop scheduling is given in Chang et al. [26].

In this paper, we present a hybrid genetic algorithm approach that considers the single machine scheduling problem with job dependent penalties. First, we derive the dominance properties of (the conditions on) the optimal schedule based on the processing times, due dates, and the job dependent penalties. These dominance properties are only necessary conditions and not sufficient conditions for any given schedule to be optimal. In our hybrid genetic algorithm, we start with a randomly generated population of solutions. First, we use the dominance properties to obtain better starting solutions for the genetic algorithm. We call this a genetic algorithm with dominance properties (GADP). This GADP is a hybrid genetic algorithm and the dominance properties are applied to generate initial solutions to obtain better starting schedules. We have compared the performance of hybrid genetic algorithm (GADP) with simple genetic algorithm (SGA). We present the results to show that this hybrid genetic algorithm (GADP) performs very well for the test problems available in the literature.

2. Dominance properties of two adjacent jobs

In this section, we derive the dominance properties for two adjacent jobs (i and j), which has distinct due dates (d_i and d_j), earliness penalties (α_i and α_j), and tardiness penalties (β_i and β_j). The processing time of these jobs are p_i and p_j. The dominance properties provide the precedence relationship between any two adjacent jobs in a schedule. In the optimal schedule, all the adjacent jobs will satisfy the dominance properties.

We consider a schedule Π, in which two adjacent jobs i and j are in positions k and $k+1$, respectively. We consider the objective function $Z(\Pi)$ for schedule Π. We rewrite the objective function $Z(\Pi)$, in such a way that only terms corresponding to jobs (i and j) in positions k and $k+1$ are present explicitly in the objective function $Z(\Pi)$. The other terms are absorbed in constants defined below

$$(Z(\Pi) = G_1 + G_2 + \gamma_i|d_i - f_i| + \gamma_j|d_j - f_j|, \tag{1}$$

where

$$G_1 = \sum_{l=1}^{k-1} \gamma_1 |d_l - f_l|,$$

$$G_2 = \sum_{l=k+2}^{n} \gamma_1 |d_l - f_l|.$$

In the above expressions, the value of γ_p is defined as follows:

- $\gamma_p = \alpha_p$, if $d_p > f_p$; this means that job p is an early job.
- $\gamma_p = \beta_p$, if $d_p < f_p$; this means that job p is a tardy job.
- $\gamma_p = 0$, if $d_p = f_p$; this means that job p is an on time job.

Consider schedule Π_x given as

$$\Pi_x = \{* * * \cdots ij * \cdots * *\}.$$

Consider schedule Π_x, the jobs i and j are in positions k and $k+1$, respectively. In schedule Π_x, '*' denotes some other jobs (other than i and j) are in that positions 1 to n (other than k and $k+1$). In schedule Π_x, the finish time (f_i) of job i is ($A + p_i$) and finish time (f_j) of job j is ($A + p_i + p_j$). The value of A is the finish time of the job in position ($k-1$) and is

$$A = \sum_{l=1}^{k-1} p_i.$$

When the jobs i and j are interchanged schedule Π_x, the resulting schedule is Π_y and is

$$\Pi_y = \{* * * \cdots ji * \cdots * *\}.$$

Note that in schedule Π_y only the jobs i and j are interchanged and all other jobs are in the same positions as in schedule Π_x. In Π_y, the finish time (f_j) of job j is ($A + p_j$) and finish time (f_i) of job i is ($A + p_j + p_i$). We will compare the schedules Π_x and Π_y and find the conditions under which Π_x is better than Π_y. These conditions are the dominance properties.

In schedule Π_x, the jobs i and j are in one of the following statuses:

1. Job i is early and jobs j is early.
2. Job i is tardy and jobs j is tardy.
3. Job i is early and jobs j is tardy.
4. Job i is tardy and jobs j is early.
5. Job i is early and jobs j is on time.
6. Job i is on time and jobs j is early.
7. Job i is tardy and jobs j is on time.
8. Job i is tardy and jobs j is on time.
9. Job i is on time and jobs j is on time.

Let P be the sum of processing time of all the jobs ($P = \sum_{j=1}^{n} p_j$). We assume that $d_j < P$, for all jobs ($j = 1, 2, \ldots, n$). We discuss the case when assumption is not true later. With this assumption, we consider the above mentioned statuses one by one in detail and derive the dominance properties.

Status 1: Consider two adjacent early jobs i (in position k) and j (in position $k+1$) in the schedule Π_x. This two adjacent jobs are early means that $d_i > (A + p_i)$ and $d_j = (A + p_i + p_j)$ This $d_j = (A + p_i + p_j)$ implies that $d_j > (A + p_j)$. Hence there are two possibilities on d_i as given below

- Possibility (i). $d_i > (A + p_i + p_j)$.
- Possibility (ii). $d_i < (A + p_i + p_j)$.

Possibility (i). Here in schedule Π_x jobs i and j (in positions k and $k+1$) are also early jobs. After interchange, in schedule Π_y, the jobs j and i (in positions k and $k+1$) are also early jobs. This means that $d_i > (A + p_i)$, $d_j > (A + p_j)$, $d_i > (A + p_i + p_j)$ and $d_j > (A + p_i + p_j)$. The total absolute deviation $Z(\Pi_x), Z(\Pi_y)$ for schedules Π_x, Π_y are

$$Z(\Pi_x) = G_1 + G_2 + \alpha_i(d_i - A - p_i) + \alpha_j(d_j - A - p_i - p_j),$$
$$Z(\Pi_y) = G_1 + G_2 + \alpha_j(d_j - A - p_j) + \alpha_i(d_i - A - p_j - p_i).$$

We now derive the condition under which $Z(\Pi_x) \leqslant Z(\Pi_y)$. Let $X = Z(\Pi_y) - Z(\Pi_x)$ and is given by

$$X = -\alpha_i p_j + \alpha_j p_i.$$

From the above expression, we see that $X \geqslant 0$ when the following condition is satisfied:

$$\frac{p_i}{\alpha_i} \geqslant \frac{p_j}{\alpha_j}.$$

From the above condition, we see that if $X > 0$, then schedule Π_x is better than schedule Π_y; i.e., $Z(\Pi_x) < Z(\Pi_y)$. If $X = 0$ then $Z(\Pi_x) = Z(\Pi_y)$. For this case, job i will come before job j only when $\frac{p_i}{a_i} \geqslant \frac{p_j}{a_j}$. Based on this analysis, we state the following property.

Property 1. *In schedule Π_x, for two adjacent early jobs i (in position k) and j (in position k+ 1), and if $d_i > A + p_i + p_j$, then schedule Π_x is better than schedule Π_y, only when $\frac{p_i}{a_i} \geqslant \frac{p_j}{a_j}$.*

Conjecture. *Now, we discuss a special case when $d_i = A + p_i + p_j$. Here in the schedule Π_x jobs i and j (in positions k and k + 1) are early jobs. After interchange, in schedule Π_y, the job j (in position k) is an early job, and job i (in position k + 1) is an on time job. This means that $d_i > A + p_i$, $d_i > A + p_j$, $d_i = A + p_i + p_j$ and $d_i > A + p_i + p_j$. Here also, job i will come before job j only when $\frac{p_i}{a_i} \geqslant \frac{p_j}{a_j}$.*

This can be easily proved by considering the fact that $d_i = A + p_i + p_j$ in the above analysis.

Fig. 1 is a pictorial representation of the statuses of two adjacent jobs i and j in schedule Π_x. Fig. 1 is in (d_i, d_j) plane. For any two adjacent jobs i and j, we know the values of A, p_i, p_j, d_i, and d_j. Once we know these values, we can see that this is a point in (d_i, d_j) plane. Then, the finish time of job i in schedule Π_x is $(A + p_j)$ and the finish time of job j is $(A + p_i + p_j)$. They are marked in d_i axis. Similarly, in schedule Π_y the finish time of job j is $(A + p_j)$ and the finish time of job i is $(A + p_i + p_j)$. They are marked in d_j axis.

Hence, in Fig. 1, all the nine status are shown. Also in Fig. 1, the property for schedule Π_x has to be better than schedule Π_y, is also given. Note that this property is true only when $d_i > A + p_j$, $d_j > A + p_j$, $d_i \geqslant A + p_i + p_j$, and $d_j > A + p_i + p_j$. Region R_1 in which this property is true is shown in Fig. 1. This above conjecture, when $d_i = A + p_i + p_j$ is a point on the left side boundary of region R_1.

Note that after interchange, in schedule Π_y, job j cannot be a tardy job or an on time job because $(d_j > A + p_i + p_j)$, which implies that $(d_j > A + p_j)$.

Possibility (ii). Here in schedule Π_x jobs i and j (in positions k and $k+1$) are early jobs. After interchange, in schedule Π_y, the job j (in position k) is an early job and job i (in position $k+1$) is a tardy job. This means that $d_i > A + p_j$, $d_j > A + p_j$, $d_i < A + p_i + p_j$, and $d_j > A + p_i + p_j$. The total absolute deviation $Z(\Pi_x)$, $Z(\Pi_y)$ for schedules Π_x, Π_y are

$$Z(\Pi_x) = G_1 + G_2 + \alpha_i(d_i - A - p_i) + \alpha_j(d_j - A - p_i - p_j),$$
$$Z(\Pi_y) = G_1 + G_2 + \alpha_j(d_j - A - p_j) + \alpha_i(A + p_j + p_i - d_i).$$

P	Region R_9 π_x is better than π_y.	Region R_2 Property 2 should satisfy $d_i \le \{A + p_i \frac{\alpha_i+\beta_i+\alpha_j}{\alpha_i+\beta_i} + p_j \frac{\beta_i}{\alpha_i+\beta_i}\}$	Region R_1 Property 1 should satisfy $\frac{p_i}{\alpha_i} \ge \frac{p_j}{\alpha_j}$.
$A + p_i + p_j$	Region R_4 Property 4 should satisfy $d_j \ge \{A + p_i \frac{\alpha_j+\beta_j-\beta_i}{\alpha_j+\beta_j} + p_j \frac{\beta_j}{\alpha_j+\beta_j}\}$.	Region R_5 Property 6 should satisfy $(\alpha_j+\beta_j)\{d_j - p_j - A\} + p_j\beta_i \ge (\alpha_i+\beta_i)\{d_i - p_i - A\} + p_i\beta_j$.	Region R_7 Property 7 should satisfy $d_j \ge \{A + \frac{p_i\beta_j}{\alpha_i+\beta_i} + \frac{p_j(\alpha_j+\beta_j+\alpha_i)}{\alpha_i+\beta_i}\}$.
$A + p_j$	Region R_3 Property 3 should satisfy $\frac{p_i}{\beta_i} \le \frac{p_j}{\beta_j}$.	Region R_6 Property 6 should satisfy $d_i \le \{A + p_i \frac{\alpha_i+\beta_i-\beta_j}{\alpha_i+\beta_i} + p_j \frac{\beta_i}{\alpha_i+\beta_i}\}$.	Region R_8 π_y is better than π_x.
	(A, A)		$A + p_i$
$A + p_i + p_j$		P	

Fig. 1. Pictorial representation of various status of two adjacent jobs i and j.

We now derive the condition under which $Z(\Pi_x) \leqslant Z(\Pi_y)$. For this purpose, we obtain the value of $Z(\Pi_y) - Z(\Pi_x)$. Let $X = Z(\Pi_y) - Z(\Pi_x)$ and is given by

$$X = \alpha_j p_i + \beta_i p_j + (\alpha_i + \beta_i)(A + p_i - d_i).$$

From the above expression, we see that $X \geqslant 0$ when the following condition is satisfied:

$$d_i \leqslant \left\{A + p_i\left(\frac{\alpha_i + \beta_i + \alpha_j}{\alpha_i + \beta_i}\right) + p_j\left(\frac{\beta_i}{\alpha_i + \beta_i}\right)\right\}.$$

From the above condition, if $X > 0$, then schedule Π_x is better than schedule Π_y; i.e., $Z(\Pi_x) < Z(\Pi_y)$. If $X = 0$ then $Z(\Pi_x) = Z(\Pi_y)$. For this case, job i will come before job j only when $d_i \leqslant \{A + p_i(\frac{\alpha_i+\beta_i+\alpha_j}{\alpha_i+\beta_i}) + p_j(\frac{\beta_i}{\alpha_i+\beta_i})\}$. Based on this analysis, the following property exists.

Property 2. *In schedule Π_x, for two adjacent early jobs i (in position k) and j (in position $k + 1$), and if $d_i < A + p_i + p_j$, then schedule Π_x is better than schedule Π_y only when $d_i \leqslant \{A + p_i(\frac{\alpha_i+\beta_i+\alpha_j}{\alpha_i+\beta_i}) + p_j(\frac{\beta_i}{\alpha_i+\beta_i})\}$.*

Conjecture. *Now, we discuss a special case when $d_i = (A + p_i + p_j)$. Here in schedule Π_x job i and j (in positions k and $k + 1$) are early jobs. After interchange, in schedule Π_y, the job j (in position k) is an early job, and job j (in position $k + 1$) is an on time job. This means that $d_i > (A + p_i)$, $d_j > (A + p_j)$, $d_i = (A + p_i + p_j)$ and $d_j > (A + p_i + p_j)$. Here also, job i will come before job j only when $\frac{p_i}{\alpha_i} \geqslant \frac{p_j}{\alpha_j}$. This can be easily proved by considering the fact that d_i=$(A + p_i + p_j)$ in the above analysis.*

Note that this property is true only when $d_i > (A + p_i)$, $d_j > (A + p_j)$, $d_i \leqslant (A + p_i + p_j)$, and $d_j > (A + p_i + p_j)$. Region R_2 in which this property is true is shown in Fig. 1. The above conjecture $d_i = (A + p_i + p_j)$ is a point on the right side boundary of region R_2 and is also the left side boundary of region R_1. Hence, the same property as in Property 1, i.e., $\frac{p_i}{\alpha_i} \geqslant \frac{p_j}{\alpha_j}$, is derived.

Status 2: Consider two adjacent tardy jobs i (in position k) and j (in position $k+1$) in the schedule Π_x. This two adjacent jobs are tardy means that $d_i < (A + p_i)$ and $d_j < (A + p_i + p_j)$. This $d_i < (A + p_i)$ implies that $d_i < (A + p_i + p_j)$. Hence, there are two possibilities on d_j as given below

- Possibility (i). $d_j < (A + p_j)$.
- Possibility (ii). $d_j > (A + p_j)$.

Possibility (i). Here in schedule Π_x jobs i and j (in positions k and $k+1$) are tardy jobs. After interchange, in schedule Π_y, the jobs j and i (in positions k and $k+1$) are also tardy jobs. This means that $d_i < (A + p_i)$, $d_i < (A + p_j)$ which implies $d_i < (A + p_i + p_j)$ and $d_j < (A + p_i + p_j)$. The total absolute deviation $Z(\Pi_x)$, $Z(\Pi_y)$ for schedules Π_x, and Π_y are

$$Z(\Pi_x) = G_1 + G_2 + \beta_i(A + p_i - d_i) + \beta_j(A + p_i + p_j - d_j),$$
$$Z(\Pi_y) = G_1 + G_2 + \beta_j(A + p_j - d_j) + \beta_i(A + p_j + p_i - d_i).$$

We now derive the condition under which $Z(\Pi_x) \leqslant Z(\Pi_y)$. For this purpose, we obtain the value of $Z(\Pi_y) - Z(\Pi_x)$. Let $X = Z(\Pi_y) - Z(\Pi_x)$ and is given by

$$X = -\beta_j p_i + \beta_i p_j.$$

Form the above expression, we see that $X \geqslant 0$ when the following condition is satisfied:

$$\frac{p_i}{\beta_i} \leqslant \frac{p_j}{\beta_j}.$$

Form the above condition, if $X > 0$, then schedule Π_x is better than schedule Π_y; i.e., $Z(\Pi_x) < Z(\Pi_y)$. If $X = 0$ then $Z(\Pi_x) = Z(\Pi_y)$. For this case, job i will come before job j only when $\frac{p_i}{\beta_i} \leqslant \frac{p_j}{\beta_j}$. Based on this analysis, we state the following property.

Property 3. *In schedule Π_x, for two adjacent tardy jobs i (in position k) and j (in position k + 1), and if $d_j < (A + p_j)$, then schedule Π_x is better then schedule Π_y only when $\frac{p_i}{\beta_i} \leqslant \frac{p_j}{\beta_j}$.*

Conjecture. *Now, we discuss a special case when $d_j = (A + p_j)$. Here in the schedule Π_x jobs i and j (in positions k and k + 1) are tardy jobs. After interchange, in schedule Π_y, the job j (in positions k) is an on time job, and job i (in positions k + 1) is an tardy job. This means that $d_i < (A + p_i)$, $d_j = (A + p_j)$, $d_i < (A + p_i + p_j)$ and $d_j < (A + p_i + p_j)$. Here also, job i will come before job j only when $\frac{p_i}{\beta_i} \leqslant \frac{p_j}{\beta_j}$. This can be easily proved by considering the fact $d_j = (A + p_j)$ in the above analysis.*

Note that this property is true only when $d_i < (A + p_i)$, $d_j = (A + p_j)$, $d_i < (A + p_i + p_j)$ and $d_j < (A + p_i + p_j)$. The region R_3 in which this property is true is shown in Fig. 1. This above conjecture $d_j = (A + p_j)$ is a point on the upper boundary of the region R_3.

Note that after interchange, in schedule Π_y, job i cannot be an early job or an on time job because $d_i < (A + p_i)$, which implies that $d_i < (A + p_i + p_j)$.

Possibility (ii). Here in schedule Π_x jobs i and j (in positions $k$and $k+1$) are tardy jobs. After interchange, in schedule Π_y, the job j (in position k) is an early job and job j (in position $k+1$) is a tardy job. This means that $d_i < (A + p_i)$, $d_j > (A + p_j)$, $d_i < (A + p_i + p_j)$, $d_j < (A + p_i + p_j)$. The total absolute deviation $Z(\Pi_x), Z(\Pi_y)$ for schedules Π_x, Π_y are

$$Z(\Pi_x) = G_1 + G_2 + \beta_i(A + p_i - d_i) + \beta_j(A + p_i + p_j - d_j),$$
$$Z(\Pi_y) = G_1 + G_2 + \alpha_j(d_j - A - p_j) + \beta_i(A + p_j + p_i - d_i).$$

We now derive the condition under which $Z(\Pi_x) \leqslant Z(\Pi_y)$. For this purpose, we obtain the value of $Z(\Pi_y) - Z(\Pi_x)$. Let $X = Z(\Pi_y) - Z(\Pi_x)$ and is given by

$$X = -\beta_j p_i + \beta_i p_j + (\alpha_j + \beta_j)(d_j - A - p_j).$$

From the above expression, we see that $X \geqslant 0$ when the following condition is satisfied:

$$d_j \geqslant \left\{ A + \frac{p_j(\alpha_j + \beta_j - \beta_i) + \beta_j p_i}{(\alpha_j + \beta_j)} \right\}.$$

From the above condition, we see that if $X > 0$, then the schedule Π_x is better than the schedule Π_y; i.e., $Z(\Pi_x) < Z(\Pi_y)$. If $X = 0$ then $Z(\Pi_x) = Z(\Pi_y)$. For this case, job i will come before job j only when $d_j \geqslant \{A + \frac{p_j(\alpha_j+\beta_j-\beta_i)+\beta_j p_i}{(\alpha_j+\beta_j)}\}$.

Based on this analysis, Property 4 can also be stated. However, to simplify the proving procedures the rest of the properties are listed in Appendix A.

3. Hybrid genetic algorithm

We now describe our hybrid genetic algorithm employed in this study. It is known that genetic algorithm with a good initial solution will give better results in less computation time. In order to obtain good initial solution, these dominance properties derived are applied.

Algorithm for generating initial schedules: Our algorithm is basically an adjacent pair wise interchange procedure. We start with a random initial schedule. In this schedule, we consider two adjacent jobs i and j. For these two adjacent jobs, the values of p_i, p_j, A, d_i, and d_j are known. These values correspond to a point in one of the nine regions in Fig. 1.

Once, we know the region in Fig. 1, we know the property these adjacent jobs have to satisfy. If this property is not satisfied, we interchange the jobs i and j. At the termination, we obtain a resulting schedule in which all the adjacent jobs satisfy their corresponding properties. We know that these properties are only necessary conditions and not sufficient conditions for any given schedule to be optimal. So, the resulting schedule may not be an optimal schedule but the resulting schedule is a better schedule than the random initial schedule.

In our hybrid genetic algorithm, we start with a randomly generated population of solutions. First, we apply our algorithm which uses the dominance properties to obtain better starting solutions for the genetic algorithm. We call this a genetic algorithm with dominance properties (GADP). This GADP is a hybrid genetic algorithm. By hybrid, we mean that in our GADP, the dominance properties are applied to the randomly generated initial solutions to obtain better starting schedules. The pseudo code of GADP is listed as follows:

Pseudo Code of GADP:
MainProcedure( )
Population: The population used in the Genetic Algorithm
Generations: The number of generations
1. Initiate *Population*
2. ConstructInitialPopulation(*Population*)
3. RemovedIdenticalSolution( )
4. counter ← 0
5. **while** counter < *generations* **do**
6. Evaluate Objectives and Fitness (i)
7. FindPareto(i)
8. Selection with Elitism Strategy(i)
9. Crossover(i)
10. Mutation(i)
11. Replacement(i)
12. counter ← counter + 1
13. **end while**

```
ConstructInitialPopulation(Population)
k: The number of initial solutions
initialSolution: We generate the random initial solutions.
for i = 1 to k
    set initialSolution ← Random_Solution ()
    set Population(k) ← DominanceProperties (initialSolution)
end for
```

```
DominanceProperties (schedule)
Iter: The number of iterations
Pos: The position in the sequence
n: the number of jobs
schedule: the sequence of the jobs
preSchedule: the previous schedule of the jobs
calculateFinishTime( )
set preSchedule ← schedule
for i = 1 to Iter
    for Pos = 1 to 2
        for j = Pos to n − 1
        job1 = schedule [j]
        job2 = schedule [j + 1]
            isSatisfyProperty = checkProperty (job1,job2)
            if(isSatisfyProperty == false)
                swapTwoJobs(j,j + 1);
                updateFinishTime( )
            end if
            j ← j + 1
        end for
end for
if (isIdenticalSolution (schedule,preSchedule))
        break;
else
        set preSchedule ← schedule
    end if
end for
```

Detailed implementations of the crossover and mutation operations are described as follows:

Crossover Operation: Two-point crossover

In the crossover step, two chromosomes are randomly selected and a random number r_c is generated first. If r_c is smaller than P_c, then crossover implements on this pair, else no crossover.

Detailed procedures of a two-point crossover are described as follows:

1. Select two chromosomes Parent 1 and Parent 2.
2. Randomly assign two cutting points, suppose the cutting points are located at ith and jth positions, respectively. Genes beyond the cutting points in Parent 1 are directly duplicated to the Offspring.
3. The vacant positions in the Offspring are duplicated from Parent 2.

For example, two 10-job chromosomes for Parent 1 and Parent 2 are shown in Fig. 2. Two cutting points are assigned at position 3 and position 7. Jobs before position 3 and jobs after position 7 in Parent 1 are duplicated to the tails of the new chromosome, i.e., the Offspring, as shown in Fig. 3. The sequences of vacant positions in the middle of Offspring are duplicated from Parent 2. Then the crossover operation is finished and a new chromosome generated, i.e., Offspring, is shown in Fig. 4.

	↓				↓					
Parent 1	1	2	3	4	5	6	7	8	9	10
Parent 2	1	6	4	7	2	3	9	10	8	5

Fig. 2. Two chromosomes for Parent 1 and Parent 2.

Offspring	1	2						8	9	10

Fig. 3. Duplication of genes in two tails from Parent 1.

Offspring	1	2	6	4	7	3	5	8	9	10

Fig. 4. Duplication of genes in the middle from Parent 2.

Mutation Operation: Swap mutation

In the mutation step, two chromosomes are randomly selected and a random number r_m is generated first. If r_m is smaller than P_m, mutate the selected chromosomes, else no mutation.

Swap mutation is to randomly select two positions in a chromosome and interchange the two positions. For example, position 1 and 2 are assigned as shown in Fig. 5. Before mutation, the sequence is 2-9-5-3-4-8-10-6-7-1. The corresponding jobs at position 1 and 2 are job 9 and job 10. Then the two jobs are interchanged and the sequence is 2-10-5-3-4-8-9-6-7-1 after mutation (see Fig. 5).

4. Experimental results

In the earlier studies Sourd et al. [27] and Sourd et al. [28], single machine scheduling problem is considered, and many test problems are provided with job dependent 20 earliness and tardiness penalties. These test problems are generated by Sourd et al. in the following manner and are available in the internet. For a given value of n (number of jobs), the processing times are generated randomly from the uniform distribution $U = [10; 100]$. The due dates d_j are generated from $U = [d_{\min}, d_{\min}, +\rho P]$, where $d_{\min} = \mathrm{MAX}(0, P(\tau - \rho/2))$ and $P = \sum_{j=1}^{n} p_j$. The two parameters ρ and τ are tardiness and range parameters. These test problems are available in Sourd et al. for $n \in \{20, 30, 40, 50\}$, $\tau \in \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ and for the value of $\rho \in \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. The earliness ($\alpha_j$) and tardiness ($\beta_j$) for all jobs are generated randomly from the uniform distribution $U = [1; 5]$. Simple Genetic Algorithm: (SGA) First, we solved these test problems with a simple genetic algorithm. SGA starts with an initial population of randomly generated solutions (schedules). These solutions are modified by using genetic operators and this process is repeated over a number of generations. At the termination of SGA, we obtain the best solution (schedule) for the problem.

We have included the dominance properties in the genetic algorithm to obtain better starting solutions. This hybrid genetic algorithm with dominance properties is GADP. In GADP approach also, we start with an initial population of randomly generated solutions (schedules). Before using the genetic operators, we first

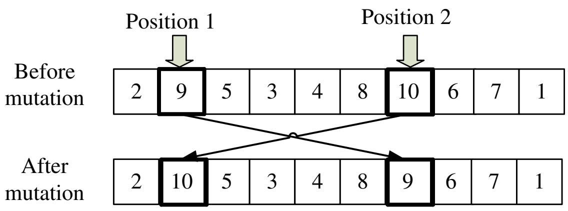

Fig. 5. Swap mutation.

Table 1
properties of optimal schedule

Property number	Processing times and due dates	Condition for Π_x to be better than Π_y
1	In Π_x: $d_i > (A+p_i)$, $d_j \geqslant (A+p_i+p_j)$ In Π_y: $d_i > (A+p_j)$, $d_i \geqslant (A+p_i+p_j)$	$\frac{p_i}{\alpha_i} \geqslant \frac{p_j}{\alpha_j}$
2	In Π_x: $d_i > (A+p_i)$, $d_j \geqslant (A+p_i+p_j)$ In Π_y: $d_j > (A+p_j)$, $d_i \leqslant (A+p_i+p_j)$	$d_i \leqslant \left\{A + p_i\left(\frac{\alpha_i+\beta_i+\alpha_j}{\alpha_i+\beta_i}\right) + p_j\left(\frac{\beta_i}{\alpha_i+\beta_i}\right)\right\}$
3	In Π_x: $d_i < (A+p_i)$, $d_j \leqslant (A+p_i+p_j)$ In Π_y: $d_j \leqslant (A+p_j)$, $d_i < (A+p_i+p_j)$	$\frac{p_i}{\beta_i} \leqslant \frac{p_j}{\beta_j}$
4	In Π_x: $d_i < (A+p_i)$, $d_j \leqslant (A+p_i+p_j)$ In Π_y: $d_j \geqslant (A+p_j)$, $d_i < (A+p_i+p_j)$	$d_j \geqslant \left\{A + \frac{p_j(\alpha_j+\beta_j-\beta_i)+\beta_j p_i}{(\alpha_j+\beta_j)}\right\}$
5	In Π_x: $d_i > (A+p_i)$, $d_j < (A+p_i+p_j)$ In Π_y: $d_j > (A+p_j)$, $d_j \leqslant (A+p_i+p_j)$	$(\alpha_j+\beta_j)\ \{d_j-p_j-A\} + \beta_i p_j \geqslant (\alpha_i+\beta_i)\ \{d_i-p_i-A\} + \beta_j p_i$
6	In Π_x: $d_i > (A+p_i)$, $d_j < (A+p_i+p_j)$ In Π_y: $d_j > \leqslant (A+p_j)$, $d_i \leqslant (A+p_i+p_j)$	$d_i \leqslant \left\{A + \frac{p_i(\alpha_i+\beta_i-\beta_j)}{(\alpha_i+\beta_j)}\right\}$
7	In Π_x: $d_i > (A+p_i)$, $d_j < (A+p_i+p_j)$ In Π_y: $d_j > (A+p_j)$, $d_i > (A+p_i+p_j)$	$d_j \geqslant \left\{A + \frac{p_j(\alpha_j+\beta_j+\alpha_i)+\beta_j p_i}{(\alpha_j+\beta_j)}\right\}$
8	In Π_x: $d_i > (A+p_i)$, $d_j < (A+p_i+p_j)$ In Π_y: $d_j < (A+p_j)$, $d_i > (A+p_i+p_j)$	*Always* Π_y *is better*
9	In Π_x: $d_i \leqslant (A+p_i)$, $d_j > (A+p_i+p_j)$ In Π_y: $d_j > (A+p_j)$, $d_i < (A+p_i+p_j)$	*Always* Π_x *is better*

check the dominance properties (using our algorithm) and obtain better starting solutions (schedules). Of course, the computational times of DP is taken into consideration as listed in Table 2 for reference. Since all the neighborhood structures of each job have to be evaluated, the time-complexity of DP is $O(n^2)$. To generate k initial solutions, the time-complexity will be $k\ O(n^2)$. After this, the solutions are modified using genetic operators and this process is repeated over a number of generations. At the termination of GADP, we obtain the best solution (schedule) for the problem.

We have compared the performance of GADP with SGA on these test problems. The value of all other parameters in the genetic algorithm such as population size, probability of crossover and mutation, selection method, and the maximum number of generations are kept the same for both SGA and GADP. These parameters are also given in Table 3. In addition, to observe the effectiveness of various approaches, DP, SGA,

Table 2
Average CPU seconds of different algorithms on each instance

Job sets	DP	SGA	GADP
20	0.021073	1.06082	1.05788
30	0.084101	1.67792	1.681973
40	0.284798	2.478637	2.482051
50	0.682626	3.556849	3.513418

Table 3
Parameter values used in SGA and GADP

Crossover operator	Two-point crossover
Crossover rate	0.8
Mutation operator	Swap mutation
Mutation rate	0.3
Population size	100
Generations	1000
Clone strategy	Swap mutation

GADP are also compared with the optimal solutions by B&B approach as shown in Table 4. All solutions generated by GADP in these small instances are very close to the optimal solutions. We have used the objective function value at the termination of SGA and GADP, as the performance measure in our study. For each

Table 4
DP, SGA, and GADP compared with the optimal approach (B&B) for small instances

Instance	DP			SGA objective value			GADP objective value			B&B		CPU time (Avg)		
	Min	Mean	Max	Min	Mean	Max	Min	Mean	Max	Opt.	CPU time	DP	SGA	GADP
sks222a	5298	5362	5393	5286	5401.7	5643	5286	5291	5298	5286	1.06	0.0095	1.0573	1.0631
sks223a	4546	4546	4546	4442	4511.4	4695	4442	4442	4442	4442	0.53	0.03377	1.063	1.0587
sks224a	3840	3840	3840	3840	3884.3	4144	3840	3840	3840	3840	1.51	0.01297	1.053	1.0594
sks225a	3989	4089.5	4113	3958	4173.5	4389	3958	3958.6	3977	3958	0.58	0.01317	1.059	1.0588
sks226a	3053	3053	3053	3020	3148.1	3481	3020	3020	3020	3020	0.38	0.02803	1.049	1.0645
sks227a	2048	2048	2048	2001	2114.7	2795	2001	2030.8	2048	2001	0.53	0.0106	1.0547	1.0613
sks228a	2111	2193.9	2199	2085	2155.6	2749	2085	2085	2085	2085	4.19	0.016033	1.0504	1.0592
sks232a	4357	4357	4357	4319	4461.7	4801	4319	4320.4	4321	4319	0.94	0.0157	1.0645	1.0615
sks233a	4411	4578.2	4628	4411	4552.5	5120	4411	4411	4411	4411	0.36	0.03693	1.0536	1.0625
sks234a	5076	5076	5076	5060	5184.1	5458	5040	5062.8	5076	5040	1.19	0.0145	1.0719	1.0532
sks235a	3183	3217.1	3219	3118	3211.3	3462	3118	3118	3118	3118	0.66	0.01663	1.064	1.0583
sks236a	2887	2927.8	2932	2806	2977.6	3379	2801	2801.7	2806	2801	2.06	0.01557	1.0739	1.0558
sks237a	1873	1873	1873	1809	1950	2430	1809	1809	1809	1809	0.56	0.015067	1.0605	1.0506
sks238a	1915	1968.3	1972	1872	2004.8	2575	1872	1872	1872	1872	4.2	0.02247	1.0651	1.0541

Table 5
Objective function values in SGA and GADP, for 20, 30, 40, and 50 jobs

Job	Instance	SGA objective value			GADP objective value		
		Minimum	Mean	Maximum	Minimum	Mean	Maximum
20	sks272a	6420	6663.3	7556	6420	6445.1	7173
	sks273a	8007	8488	9277	8007	8074.7	8358
	sks274a	6030	6505.8	6948	6030	6095.4	6357
	sks275a	9513	9721	10437	9513	9513	9513
	sks276a	3526	3975.6	5014	3526	3612.8	3629
	sks277a	6066	7155	8785	6066	6135.1	6584
	sks278a	7700	8236.2	9765	7700	7701.3	7738
30	sks372a	20,917	22,860	24,935	20,818	21,066	21,601
	sks373a	13,557	14,491	16,558	13,465	13,535	13,723
	sks374a	12,171	14,371	18,357	12,171	12,243	12,610
	sks375a	13,686	14,503	16,599	13,662	13,825	13,922
	sks376a	12,718	13,917	16,381	12,594	12,691	12,834
	sks377a	10,300	11,394	13,380	10,206	10,236	10,397
	sks378a	14,213	14,812	15,910	13,988	13,996	14,111
40	sks472a	31,841	34,300	40,461	31,426	31,692	32,298
	sks473a	31,832	33,742	36,054	31,474	31,675	32,472
	sks474a	29,487	31,866	35,967	28,688	28,761	29,031
	sks475a	27,433	31,348	37,175	24,904	25,566	27,090
	sks476a	22,907	25,399	30,520	21,983	22,136	22,426
	sks477a	26,884	29,945	34,366	26,548	26,635	26,803
	sks478a	19,111	22,625	31,300	18,387	18,502	18,995
50	sks572a	50,929	54,373	57,042	49,387	49,705	50,384
	sks573a	45,007	47,290	50,441	44,664	44,889	45,743
	sks574a	42,715	47,354	53,077	40,998	41,371	42,516
	sks575a	28,902	32,658	46,119	25,476	25,585	25,878
	sks576a	34,389	37,516	41,303	32,565	33,151	33,572
	sks577a	30,546	37,738	45,969	28,646	28,891	29,307
	sks578a	49,851	54,063	59,652	49,087	49,207	49,335

value of (n, τ, ρ) the test problems were run 30 times. The average value of the objective function at the termination of SGA and GADP are obtained. Taking the average of 30 runs give a better picture rather than running only once and comparing the objective function value. The average value of the objective function in GADP is always less than the average value of the objective function in SGA. This shows that GADP performs better than SGA. A sample of results comparing the objective function value is given in Table 5. As expected, in some runs both the SGA and GADP gives the same objective function value. This is shown in pictorial form in Fig. 6.

A comparison of SGA and GADP with the optimal values of the objective function is shown in Table 6. The optimal value is obtained by branch and bound technique. In Table 4, we present the minimum, mean and maximum value of the objective function obtained in 30 runs for both SGA and GADP. This also clearly shows that GADP performs better than SGA. We have considered the case with 100 and 200 jobs and observed that the performance of GADP is better than SGA.

We have assumed in our analysis that $d_j \prec P$ for all the jobs, where $P = \sum_{i=1}^{n} p_i$. When $d_j \geqslant \sum_{i=1}^{n} p_i$ for some jobs, the following approach is used. First consider the jobs for which the due date is more than P. These jobs will appear at the end of the schedule. So remove these jobs from n. Let the new number of jobs is n_1. Find

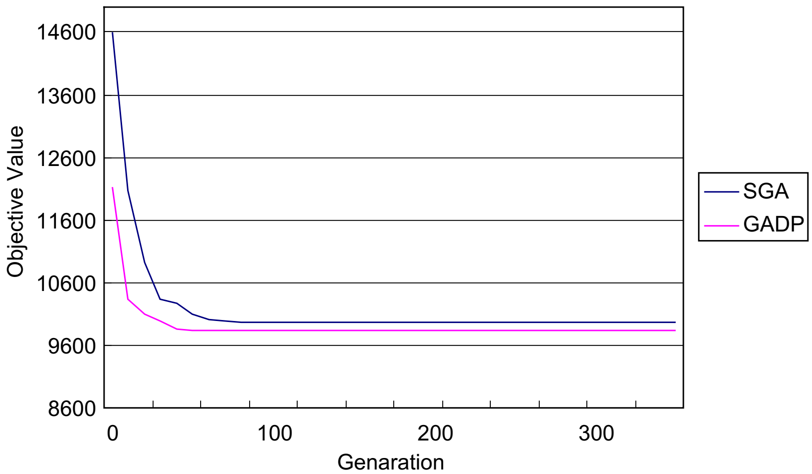

Fig. 6. Performance comparison of SGA and GADP (Instance sks225a of 20 jobs).

Table 6
Comparisons of objective function

Job	Instance	SGA objective value			GADP objective value		
		Minimum	Mean	Maximum	Minimum	Mean	Maximum
20	sks225a	9838	10,117	11,371	9838	9843.1	9860
	sks235a	7116	7352.3	8977	7116	7195.3	7285
	sks245a	4686	5064.7	6313	4686	4698.8	4807
30	sks325a	19,693	20,484	22,869	19,654	19,737	20,193
	sks335a	19,008	20,085	24,740	19,008	19,023	19,123
	sks345a	14,138	16,583	18,904	14,095	14,319	15,184
40	sks425a	22,410	26,040	31,484	22,124	22,230	22,373
	sks435a	24,098	27,623	33,686	24,055	24,187	24,351
	sks445a	23,222	26,336	36,176	22,551	22,556	22,686
50	sks525a	47,669	50,895	58,513	47,382	47,882	48,475
	sks535a	45,486	51,907	67,124	44,319	44,923	45,950
	sks545a	20,839	26,824	32,957	20,570	20,604	20,729

the value of P for these n_1 jobs. In these n_1 jobs also the due date for some jobs may be greater than the sum of processing times. Remove these jobs also. Let the new number of jobs be n_2. Find the value of P for these n_2 jobs. In these n_2 jobs if the due date of all the jobs is less than P, then use GADP with n_2 jobs and obtain the schedule. The remaining $(n - n_2)$ jobs are arranged in the decreasing order of their processing times at the end of the schedule. This is also suggested in Sourd et al. [27,28].

5. Conclusions and future researches

Single machine scheduling problem with n jobs, in which each job j ($j = 1, 2, \ldots, n$) has a processing time p_j, a due date d_j, earliness penalty α_j, and tardiness penalty β_j. The objective is to minimize the weighted sum of earliness and tardiness costs, without considering machine idle times. In this research, a set of dominance properties of (the conditions on) the optimal schedule based on the processing times, due dates, and the job dependent penalties are derived. In addition, a hybrid genetic algorithm that uses the dominance properties (GADP) is presented.

The performance of this hybrid genetic algorithm is compared with simple genetic algorithm (SGA). Although genetic algorithms are able to solve different kind of combinatorial optimization problems, it is observed that the convergence of genetic algorithm is slow. One way of improving the convergence in genetic algorithms is to include the knowledge from problem domain. Dominance Properties are included in genetic algorithms to improve the convergence. The dominance properties obtain efficient solutions before the genetic operations are carried out. Once the DP generates good initial solutions efficiently, the genetic algorithms are able to converge faster. It is true that DP needs additional computational efforts, but helps GA to converge faster so that GAs requires less number of generations. The only disadvantage is the proofs of dominance properties for the scheduling problem that is necessary. It is the reason why the numerical results on test problems are presented to show that this hybrid genetic algorithm (GADP) performs better than simple genetic algorithm (SGA).

There are some new directions to further improve the hybrid algorithm. The first will be to applied local search to improve the performance of the hybrid algorithm and the idea is pretty similar to memetic algorithm. Chromosomes generated after crossover or mutation operators can be further improve by the local search procedure. The second is apply the same approach for more complicated problems such as single machine scheduling with setups, parallel machine or flow shop problems. These are interesting new applications to the academic researchers or industrial practitioners.

Appendix A

Detailed proving procedures for Properties 4–12 are listed in the following:

Property 4. *In schedule Π_x for two adjacent tardy jobs i (in position k) and j (in position k + 1), and if $d_j > (A + p_j)$, than the schedule Π_x is better than the schedule Π_y only when* $d_j > (A + \frac{p_j(\alpha_j+\beta_j-\beta_i)+\beta_j p_i}{(\alpha_j+\beta_j)})$.

Conjecture. *Now, we discuss a special case when $d_j = (A + p_j)$. Here in the schedule Π_x jobs i and j (in position k and k + 1) are tardy jobs. After interchange, in schedule Π_y, the job j (in position k) is an on time job, and job i (in position k + 1) is an tardy job. This means that $d_i < (A + p_i)$, $d_j = (A + p_j)$,$d_i < (A + p_i + p_j)$ and $d_j < (A + p_i + p_j)$. Here also, job i will come before job j only when $\frac{p_i}{\beta_i} \leqslant \frac{p_j}{\beta_j}$.*

This can be easily proved by considering the fact that $d_j = (A + p_j)$ in the above analysis.

Note that this property is true only when $d_i < (A + p_i)$, $d_j > (A + p_j)$, $d_i < (A + p_i + p_j)$, and $d_j < (A + p_i + p_j)$. The region, R_4, in which this property is true is shown in Fig. 1. This above conjecture $d_j = (A + p_j)$ is a point on the lower boundary of region j, and is also the upper boundary of region R_3. Hence, we get the same property as in Property 3, i.e., $\frac{p_i}{\beta_i} \leqslant \frac{p_j}{\beta_j}$.

Status 3: Consider two adjacent jobs i (in position k) is an early job, and j (in position $k + 1$) is a tardy job, in the schedule Π_x. This two adjacent jobs are early and tardy means that $d_i > (A + p_i)$ and $d_j < (A + p_i + p_j)$.

This $d_j < (A + p_i + p_j)$ implies the two possibilities $d_j > (A + p_j)$ and $d_j < (A + p_j)$. Also, $d_i > (A + p_i)$ implies the two possibilities $d_i > (A + p_i + p_j)$ and $d_i < (A + p_i + p_j)$. Hence, there are four possibilities as given below

- Possibility (i). $d_j > (A + p_j)$ and $d_i < (A + p_i + p_j)$.
- Possibility (ii). $d_j < (A + p_j)$ and $d_i < (A + p_i + p_j)$.
- Possibility (iii). $d_j > (A + p_j)$ and $d_i > (A + p_i + p_j)$.
- Possibility (iv). $d_j < (A + p_j)$ and $d_i < (A + p_i + p_j)$.

Possibility (i). Here, in schedule Π_x job i (in position k) is an early job and job j (in position $k + 1$) is a tardy job. After interchange, in schedule Π_y job j (in position k) is an early job and job i (in position $k + 1$) is a tardy job. This means that $d_i > (A + p_i)$, $d_j > (A + p_j)$, $d_i < (A + p_i + p_j)$ and $d_j < (A + p_i + p_j)$. The total absolute deviation $Z(\Pi_x)$, $Z(\Pi_y)$ for schedule Π_x and Π_y are

$$Z(\Pi_x) = G_1 + G_2 + \alpha_i(d_i - A - p_i) + \beta_j(A + p_i + p_j - d_j),$$
$$Z(\Pi_y) = G_1 + G_2 + \alpha_j(d_j - A - p_j) + \beta_i(A + p_j + p_i - d_i).$$

We now derive the condition under which $Z(\Pi_x) \leqslant Z(\Pi_y)$. For this purpose, we obtain the value of $Z(\Pi_y) - Z(\Pi_x)$. Let $X = Z(\Pi_y) - Z(\Pi_x)$ and is given by

$$X = (\alpha_j + \beta_j)\{d_j - p_j - A\} + \beta_i p_j - (\alpha_i + \beta_i)\{d_i - p_i - A\} - \beta_j p_i.$$

From the above expression, we see that $X \geqslant 0$ when the following condition is satisfied:

$$(\alpha_j + \beta_j)\{d_j - p_j - A\} + \beta_i p_j - (\alpha_i + \beta_i)\{d_i - p_i - A\} - \beta_j p_i.$$

From the above condition, we see that if $X \succ 0$, then the schedule Π_x is better than the schedule Π_y; i.e., $Z(\Pi_x) < Z(\Pi_y)$. If $X = 0$ then $Z(\Pi_x) = Z(\Pi_y)$. For this case, job j will come before job j only when $(\alpha_j + \beta_j)\{d_j - p_j - A\} + \beta_i p_j - (\alpha_i + \beta_i)$ $\{d_i - p_i - A\} - \beta_j p_i$. Based on this analysis, we state the following property.

Property 5. *In schedule Π_x, for two adjacent jobs i (in position k) an early job, and j (in position k + 1) a tardy job, and if $d_j > (A + p_j)$,, $d_i < (A + p_i + p_j)$, then schedule Π_x is better than schedule Π_y only when $(\alpha_j + \beta_j)$ $\{d_j - p_j - A\} + \beta_i p_j - (\alpha_i + \beta_i)\{d_i - p_i - A\} - \beta_j p_i$.*

Note that this property is true only when $d_i > (A + p_i)$, $d_j > (A + p_j)$, $d_i < (A + p_i + p_j)$ and $d_j < (A + p_i + p_j)$. Region R_5, in which this property is true, is shown in Fig. 1.

Possibility (ii). Here, in schedule Π_x job i (in position k) is an early job and job j (in position $k + 1$) is a tardy job. After interchange, in schedule Π_y both the jobs j (in position k) and i (in position $k + 1$) are tardy job. This means that j and $d_j < (A + p_i + p_j)$. The total absolute deviation $Z(\Pi_x)$, $Z(\Pi_y)$ for schedule Π_x and Π_y are

$$Z(\Pi_x) = G_1 + G_2 + \alpha_i(d_i - A - p_i) + \beta_j(A + p_i + p_j - d_j),$$
$$Z(\Pi_y) = G_1 + G_2 + \beta_j(A + p_j - d_j) + \beta_i(A + p_j + p_i - d_i).$$

We now derive the condition under which $Z(\Pi_x) \leqslant Z(\Pi_y)$. For this purpose, we obtain the value of $Z(\Pi_y) - Z(\Pi_x)$. Let $X = Z(\Pi_y) - Z(\Pi_x)$ and is given by

$$X = -\beta_j p_i - \beta_i p_i - \alpha_i d_i + \beta_i p_j + (\alpha_i + \beta_i)(A + p_i).$$

From the above expression, we see that $X \geqslant 0$ when the following condition is satisfied:

$$d_i \leqslant \left\{ A + \frac{p_i(\alpha_i + \beta_i - \beta_j)}{(\alpha_i + \beta_j)} \right\}.$$

From the above condition, we see that if $X > 0$, then schedule Π_x is better than schedule Π_y; i.e., $Z(\Pi_x) < Z(\Pi_y)$. If $X = 0$ then $Z(\Pi_x) = Z(\Pi_y)$. For this case, job i will come before job j only when $d_i \leqslant \{A + \frac{p_i(\alpha_i+\beta_i-\beta_j)}{(\alpha_i+\beta_j)}\}$. Based on this analysis, we state the following property.

Property 6. *In schedule Π_x, for two adjacent jobs i (in position k) an early job, and j (in position k + 1) a tardy job, and if $d_j < (A + p_j)$ and $d_i < (A + p_i + p_j)$, then schedule Π_x is better than schedule $Z(\Pi_y)$ only when* $d_i \leqslant \{A + \frac{p_i(\alpha_i+\beta_i-\beta_j)}{(\alpha_i+\beta_j)}\}$.

Conjecture. *Now, we discuss a special case when $d_j = (A + p_j)$. Here in the schedule Π_x jobs i (in position k) is an early job and job j (in position k + 1) is a tardy job. After interchange, in schedule Π_y, the job j (in position k) is an on time job, and job i (in position k + 1) is an tardy job. This means that $d_i > (A + p_i)$, $d_j < (A + p_j)$, $d_i < (A + p_i + p_j)$ and $d_j < (A + p_i + p_j)$. Here also, job i will come before job j only when $d_i \leqslant \{A + \frac{p_i(\alpha_i+\beta_i-\beta_j)}{(\alpha_i+\beta_j)}\}$. This can be easily proved by considering the fact that d_j=$(A + p_j)$ in the above analysis. Note that we get the same result when $d_j = (A + p_j)$ in Property 5 also.*

Note that this property is true only when $d_i > (A + p_i)$, $d_j < (A + p_j)$, $d_i < (A + p_i + p_j)$ and $d_j < (A + p_i + p_j)$. Region R_6, in which this property is true is shown in Fig. 1. This above conjecture $d_j = (A + p_j)$ is a point on the upper boundary of region R_6 and is the lower boundary of region R_5. Hence, we get the same result as in Property 5 also.

Possibility (iii). Here, in the schedule Π_x jobs i (in position k) is an early job and job j (in position $k+1$) is a tardy job. After interchange, in schedule Π_y both jobs j (in position k) and i (in position $k+1$) are early jobs. This means that $d_i > (A + p_i)$, $d_j < (A + p_j)$, $d_i < (A + p_i + p_j)$ and $d_j < (A + p_i + p_j)$. The total absolute deviation $Z(\Pi_x)$, $Z(\Pi_y)$ for schedules Π_x, Π_y are

$$Z(\Pi_x) = G_1 + G_2 + \alpha_i(d_i - A - p_i) + \beta_j(A + p_i + p_j - d_j),$$
$$Z(\Pi_y) = G_1 + G_2 + \alpha_j(d_j - A - p_j) + \alpha_i(d_i - A - p_j - p_i).$$

We now derive the condition under which $Z(\Pi_x) \leqslant Z(\Pi_y)$. For this purpose, we obtain the value of $Z(\Pi_y) - Z(\Pi_x)$. Let $X = Z(\Pi_y) - Z(\Pi_x)$ and is given by

$$X = (\alpha_j + \beta_j)(d_j - A) - p_j(\alpha_j + \beta_j + \alpha_i) - \beta_j p_i.$$

From the above expression, we see that $X \geqslant 0$ when the following condition is satisfied:

$$d_j \geqslant \left\{A + \frac{p_j(\alpha_j + \beta_j + \alpha_i) + \beta_j p_i}{(\alpha_j + \beta_j)}\right\}.$$

From the above condition, we see that if $X \geqslant 0$, then schedule Π_x is better than schedule Π_y; i.e., $Z(\Pi_x) < Z(\Pi_y)$. If $X = 0$ then $Z(\Pi_x) = Z(\Pi_y)$. For this case, job i will come before job j only when $d_j \geqslant \{A + \frac{p_j(\alpha_j+\beta_j+\alpha_i)+\beta_j p_i}{(\alpha_j+\beta_j)}\}$. Base on this analysis, we state the following property.

Property 7. *In schedule Π_x, for two adjacent jobs i (in position k) an early job, and j (in position k + 1) a tardy job, and if $d_j > (A + p_j)$ and $d_i > (A + p_i + p_j)$, then schedule Π_x is better than Π_y only when* $d_j \geqslant \{A + \frac{p_j(\alpha_j+\beta_j+\alpha_i)+\beta_j p_i}{(\alpha_j+\beta_j)}\}$.

Conjecture. *Now, we discuss a special case when $d_i = (A + p_i + p_j)$. Here in the schedule Π_x jobs i (in position k) is an early job and job j (in position k + 1) is a tardy job, After interchange, in the schedule Π_y, the job j (in position k) is a tardy job, and job i (in position k + 1) is an on time job. This means that $d_i > (A + p_i)$, $d_j > (A + p_j)$, d_i=$(A + p_i + p_j)$ and $d_j < (A + p_i + p_j)$. Here also, job i will come before job j only when $d_j \geqslant \{A + \frac{p_j(\alpha_j+\beta_j+\alpha_i)+\beta_j p_i}{(\alpha_j+\beta_j)}\}$. Note that we get the same result when $d_i = (A + p_i + p_j)$ in Property 5 also.*

Note that this property is true only when $d_i > (A + p_i)$, $d_j > (A + p_j)$, $d_i = (A + p_i + p_j)$ and $d_j < (A + p_i + p_j)$. Region R_7, in which this property is true is shown in Fig. 1. This above conjecture $d_i = (A + p_i + p_j)$ is a point on the left boundary of region R_7 and is also the right boundary of region R_5. Hence, we get the same result from Property 5 also.

Possibility (iv). Here, in schedule Π_x job i (in position k) is an early job and job j (in position $k+1$) is a tardy job. After interchange, in schedule Π_y job j (in position k) is a tardy job, and job i (in position $k+1$) is an early jobs. This means that $d_i > (A + p_i)$, $d_j < (A + p_j)$, $d_i > (A + p_i + p_j)$ and $d_j < (A + p_i + p_j)$. The total absolute deviation $Z(\Pi_x)$, $Z(\Pi_y)$ for schedules Π_x, Π_y are

$$Z(\Pi_x) = G_1 + G_2 + \alpha_i(d_i - A - p_i) + \beta_j(A + p_i + p_j - d_j),$$
$$Z(\Pi_y) = G_1 + G_2 + \beta_j(A + p_j - d_j) + \beta_i(d_i - A - p_j - p_i).$$

We now derive the condition under which $Z(\Pi_x) \leqslant Z(\Pi_y)$. For this purpose, we obtain the value of $Z(\Pi_y) - Z(\Pi_x)$. Let $X = Z(\Pi_y) - Z(\Pi_x)$ and is given by

$$X = -p_i\beta_j - p_j\alpha_i.$$

From the above expression, we see that $X = -p_i\beta_j - p_j\alpha_i$, which implies that schedule Π_y is always better than schedule Π_x. Based on this analysis, we state the following property.

Property 8. *In schedule Π_x, for two adjacent jobs i (in position k) an early job, and j (in position k + 1) a tardy job, and if $d_j < (A + p_j)$ and $d_i > (A + p_i + p_j)$, then schedule Π_y is always a better schedule than schedule Π_x.*

Note that this property is true only when $d_i > (A + p_i)$, $d_j < (A + p_j)$, $d_i > (A + p_i + p_j)$ and $d_j < (A + p_i + p_j)$. Region R_8, in which this property is true, is shown in Fig. 1.

Status 4: Here, in Π_x job i (in position k) is a tardy job and job j (in position $k + 1$) is an early job. After interchange, in Π_y the job j (in position k) is an early job, and job i (in position $k + 1$) is a tardy job. This means that $d_i < (A + p_i)$, $d_j > (A + p_j)$, $d_i < (A + p_i + p_j)$ and $d_j > (A + p_i + p_j)$. The total absolute deviation $Z(\Pi_x)$, $Z(\Pi_y)$ for schedules Π_x, Π_y are

$$Z(\Pi_x) = G_1 + G_2 + \beta_i(A + p_i - d_i) + \alpha_j(d_j - A - p_i - p_j),$$
$$Z(\Pi_y) = G_1 + G_2 + \alpha_j(d_j - A - p_j) + \beta_i(A + p_j + p_i - d_i).$$

We now derive the condition under which $Z(\Pi_x) \leqslant Z(\Pi_y)$. For this purpose, we obtain the value of $Z(\Pi_y) - Z(\Pi_x)$. Let $X = Z(\Pi_y) - Z(\Pi_x)$ and is given by

$$X = p_i\alpha_j - p_j\beta_i.$$

From the above expression, we see that $X = p_i\alpha_j - p_j\beta_i$, which implies that schedule Π_x is always better than schedule Π_y. Based on this analysis, we state the following property.

Property 9. *In schedule Π_x, for two adjacent jobs i (in position k) a tardy job, and j (in position k + 1) an early job, then schedule Π_x is always better than schedule Π_y.*

Note that this property is true only when $d_i < (A + p_i)$, $d_j > (A + p_j)$, $d_i < (A + p_i + p_j)$ and $d_j > (A + p_i + p_j)$. Region R_9, in which this property is true, is shown in Fig. 1.

Now, we consider the job Status from 5 to 8. In these statuses, one of these two jobs is an on-time job. The on time jobs are special cases of the status 1–4. The on-time job represents a line in Fig. 1. Now, we consider on-time jobs.

Status 5: Consider two adjacent jobs i (in position k) an early job, and j (in position $k + 1$) is an on time job, in schedule Π_x. This means that $d_i > (A + p_i)$ and $d_j = (A + p_i + p_j)$. This $d_j = (A + p_i + p_j)$ implies that $d_j > (A + p_j)$. Hence, there are two possibilities on d_i as given below.

- Possibility (i). $d_i > (A + p_i + p_j)$.
- Possibility (ii). $d_i < (A + p_i + p_j)$.

Probability (i). Here in the schedule Π_x job (in position k) is an early job, and j (in position $k + 1$) is an on time job. After interchange, in the schedule Π_y, the jobs j and i (in position k and $k + 1$) are also early jobs. We can easily see that this Status 5 is a special case of Status.1, where $d_j = (A + p_i + p_j)$. Hence, we obtain the same property as in Property 1, for this possibility also.

Property 10. *In schedule Π_x, for two adjacent jobs i (in position k) an early job and j (in position k + 1) an on-time job, and if $d_i > (A + p_i + p_j)$, then schedule Π_x is better than schedule Π_y only when $\frac{p_i}{\alpha_i} \geqslant \frac{p_j}{\alpha_j}$.*

Note that this property is true only when $d_i > (A + p_i)$, $d_j > (A + p_j)$, $d_i = (A + p_i + p_j)$, and $d_j = (A + p_i + p_j)$. In Fig. 1, it is a point on the boundary between region R_1 and region R_7.

Possibility (ii). Here in the schedule Π_x job i (in position k) is an early job, and job j (in position $k+1$) is an on-time job. After interchange, in schedule Π_y, job j (in position k) is an early job and job i (in position $k+1$) is a tardy job. Here also, we see that this possibility is a special case of Status.1, where $d_j = (A + p_i + p_j)$. Hence, we obtain the same property as in Property 2, for this possibility also.

Property 11. *In schedule Π_x, for two adjacent jobs i (in position k) an early job, and j (in position k + 1) is an on time job, and if $d_i < (A + p_i + p_j)$, then the schedule Π_x is better than the schedule Π_y only when $d_i \leqslant \{A + p_i(\frac{\alpha_i+\beta_i+\alpha_j}{\alpha_i+\beta_i}) + p_j(\frac{\beta_i}{\alpha_i+\beta_i})\}$. Note that this property is true only when $d_i > (A + p_i)$, $d_j > (A + p_j)$, $d_i < (A + p_i + p_j)$, and $d_j = (A + p_i + p_j)$. In Fig. 1, this is a point on the boundary between regions R_2 and R_5.*

Status 6: Here, in Π_x job i (in position k) is an on time job and job j (in position $k+1$) is an early job. After interchange, in Π_y the job j (in position k) is an early job, and job i (in position $k+1$) is a tardy. This means that $d_i = (A + p_i)$, $d_j > (A + p_j)$, $d_i < (A + p_i + p_j)$, and $d_j > (A + p_i + p_j)$. We see that this Status 6, is a special case of Status 4, where $d_i = (A + p_i)$. Hence, we obtain the same Property 9, for this Status 6 also.

Property 12. *In schedule Π_x, for two adjacent jobs i (in position k) an on-time job, and j (in position k + 1) an early job, then schedule Π_x is always better schedule Π_y.*

Note that this property is true only when $d_i = (A + p_i)$, $d_j > (A + p_j)$, $d_i < (A + p_i + p_j)$, and $d_j > (A + p_i + p_j)$. In Fig. 1, it is a point on the boundary between regions R_2 and R_9.

Statuses 7 and 8: These two status 7 and 8 are also special cases of Status 2, with $d_j = (A + p_i + p_j)$, and $d_i = (A + p_i)$. These special cases represent a point on the line between two regions. T can be easily seen that these special cases are points on the boundary between regions R_3 and R_6, R_4 and R_9.

Status 9: Now, we discuss a special case when $d_i = (A + p_i)$, and $d_j = (A + p_i + p_j)$. Here in the schedule Π_x jobs (in position k) is an on time job and j (in position $k+1$) is also an on time job. The total absolute deviation for the schedule $Z(\Pi_x)$ is zero. Hence, this schedule Π_x is always better. This is point in Fig. 1.

The summary of the properties of early and tardy jobs are given in Table 1. In this Table 1, the status of job i, and j in schedule Π_x, and the interchanged schedule Π_y are given in terms of p_i, p_j, A, d_i, and d_j. In Fig. 1, we see that all the Statuses 1–4 are represented as regions R_1 to R_9. The special cases (on time jobs) are lines in this Fig. 1. The on time jobs are included in Properties 1–9. Also, in Fig. 1, for each the property (condition) schedule Π_x has to satisfy is given.

If the property is satisfied, then schedule Π_x is better than the schedule Π_y. If the property is not satisfied, then schedule Π_y is better than schedule Π_x.

References

[1] P.S. Ow, E.T. Morton, The single machine early/tardy problem, Manage. Sci. 35 (1989) 171–191.

[2] M. Azizoglu, SM. Kondakci, Ömer Kirca, Bicriteria scheduling problem involving total tardiness and total earliness penalties, Int. J. Prod. Econom. 23 (1–3) (1991) 17–24.

[3] S.D. Wu, R.H. Storer, P.C. Chang, One machine heuristic with efficiency and stability as criteria, Comput. Operat. Res. 20 (1993).

[4] L.H. Su, P.C. Chang, A heuristic to minimize a quadratic function of job lateness on a single machine, Int. J. Prod. Econom. 55 (1988) 169–175.

[5] L.H. Su, P.C. Chang, Scheduling *n* jobs on one machine to minimize the maximum lateness with a minimum number of tardy jobs, Comput. Ind. Eng. 40 (2001) 349–360.

[6] J.K. Lenstra, A.H.G. Rinnooy Kan, P. Brucker, Complexity of machine scheduling problems, Ann. Discret. Math. 1 (1977) 343–362.

[7] P.C. Chang, H.C. Lee, A greedy heuristic for bi-criterion single machine scheduling problems, Comput. Ind. Eng. 22 (2) (1992) 121–131.

[8] P.C. Chang, H.C. Lee, A two phase approach for single machine scheduling: minimizing the total absolute deviation, J. Chin. Inst. Eng. 15 (6) (1992) 735–742.

[9] P.C. Chang, A branch and bound approach for single machine scheduling with earliness and tardiness penalties, Comput. Math. Appl. 37 (1999) 133–144.

[10] H. Belouadah, M.E. Posner, C.N. Potts, Scheduling with release dates on a single machine to minimize total weighted completion time, Discret. Appl. Math. 36 (1992) 213–231.

[11] A.M.A. Hariri, C.N. Potts, Scheduling with release dates on a single machine to minimize total weighted completion time, Discret. Appl. Math. 36 (1983) 99–109.
[12] Y.D. Kim, C.A. Yano, Minimizing mean tardiness and earliness in single-machine scheduling problems with unequal due dates, Naval Res. Logist. 41 (1994) 913–933.
[13] J.M.S. Valente, R.A.F.S. Alves, Heuristics for the early/tardy scheduling problem with release dates, Working Paper 129, Faculdade de Economia do Porto, Portugal, 2003.
[14] J.M.S. Valente, R.A.F.S. Alves, Improved heuristics for the early/tardy scheduling problem with no idle time, Working Paper 126, Faculdade de Economia do Porto, Portugal, 2003.
[15] T. Abdul-Razaq, C.N. Potts, Dynamic programming state-space relaxation for single machine scheduling, J. Operat. Res. Soc. 39 (1988) 141–152.
[16] G. Li, Single machine earliness and tardiness scheduling, Eur. J. Operat. Res. 96 (1997) 46–558.
[17] C.-F. Liaw, A branch-and-bound algorithm for the single machine earliness and tardiness scheduling problem, Comput. Operat. Res. 26 (1999) 679–693.
[18] P.S. Ow, T.E. Morton, Filtered beam searches in scheduling, Int. J. Prod. Res. 26 (1988) 35–62.
[19] H.J. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, USA, 1975.
[20] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, New York, USA, 1989.
[21] L. David, Handbook of Genetic Algorithms, Van Nostrand Reingold, New York, USA, 1991.
[22] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, AI Series, Springer-Verlag, New York, USA, 1994.
[23] P.C. Chang, J.C. Hsieh, Y.W. Wang, Genetic algorithms applied in BOPP film scheduling problems, Appl. Soft Comput. 3 (2003) 139–148.
[24] P.C. Chang, S.H. Chen, K.L. Lin, Two phase sub-population genetic algorithm for parallel machine scheduling problem, Expert Syst. Appl. 29 (3) (2005) 705–712.
[25] P.C. Chang, J.C. Hsieh, C.H. Liu, A case-injected genetic algorithm for single machine scheduling problems with release time, Int. J. Prod. Econom. 103 (2) (2006) 551–564.
[26] R. Chang, M. Gen, T. Tsujimura, A tutorial survey of job shop scheduling problems using genetic algorithms – I: Representation, Comput. Ind. Eng. 30 (1996) 983–987.
[27] F. Sourd, S.K. Sidhoum, The one machine problem with earliness and tardiness penalties, J. Sched. 6 (2003) 533–549.
[28] F. Sourd, S.K. Sidhoum, An efficient algorithm for the earliness tardiness, 2005, <http://www-poleia.lip6.fr/sourd/>.