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

Pei Chann Chang\textsuperscript{a,b,\*}, Shih Hsin Chen\textsuperscript{b,1}, V. Mani\textsuperscript{c}

\textsuperscript{a}Department of Information Management, Yuan Ze University, 135, Yuan-Tung Road, 32026 Tao-Yuan, Taiwan, ROC
\textsuperscript{b}Department of Industrial Engineering and Management, Yuan Ze University, 135, Yuan-Tung Road, 32026 Tao-Yuan, Taiwan, ROC
\textsuperscript{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 \(E_j = \max(0, d_j - C_j)\) and

\[0307-904X/S - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.apm.2008.01.006\]
\( T_i = \max(0, C_i - d) \), where \( C_i \) is the completion time of \( J_i \). The objective is then to find a schedule that minimizes the sum of the earliness and tardiness penalties of all jobs \( \sum_{i=1}^{n} (z_i E_i + \beta_i T_i) \) where \( z_i \) and \( \beta_i \) are the earliness and tardiness penalties of job \( J_i \). 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 (\( z_i \) and \( z_j \)), and tardiness penalties (\( \beta_i \) and \( \beta_j \)). 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 \( \Pi_i \), in which two adjacent jobs \( i \) and \( j \) are in positions \( k \) and \( k + 1 \), respectively. We consider the objective function \( Z(\Pi_i) \) for schedule \( \Pi_i \). We rewrite the objective function \( Z(\Pi_i) \), in such a way that only terms corresponding to jobs \((i\text{ and }j)\) in positions \( k \) and \( k + 1 \) are present explicitly in the objective function \( Z(\Pi_i) \). The other terms are absorbed in constants defined below

\[
(Z(\Pi_i) = G_1 + G_2 + \gamma_1|d_i - f_i| + \gamma_2|d_j - f_j|,
\]

where
\[
G_1 = \sum_{i=1}^{k-1} \gamma_1|d_i - f_i|,
\]
\[
G_2 = \sum_{i=k+2}^{n} \gamma_1|d_i - f_i|.
\]

In the above expressions, the value of \( \gamma_p \) 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 \( \Pi_x \) given as
\[
\Pi_x = \{ \ast \cdots i j \cdots \ast \}.
\]

Consider schedule \( \Pi_x \), the jobs \( i \) and \( j \) are in positions \( k \) and \( k + 1 \), respectively. In schedule \( \Pi_x \), \( * \) denotes some other jobs (other than \( i \) and \( j \)) are in that positions 1 to \( n \) (other than \( k \) and \( k + 1 \)). In schedule \( \Pi_x \), the finish time \((f)\) of job \( i \) is \((A + p_i)\) and finish time \((f)\) of job \( j \) is \((A + p_j + p_i)\). The value of \( A \) is the finish time of the job in position \((k - 1)\) and is
\[
A = \sum_{i=1}^{k-1} p_i.
\]

When the jobs \( i \) and \( j \) are interchanged schedule \( \Pi_x \), the resulting schedule is \( \Pi_y \) and is
\[
\Pi_y = \{ \ast \cdots \ast i j \cdots \ast \}.
\]

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

In schedule \( \Pi_x \), 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 \(\Pi_x\). 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 \(\Pi_x\) jobs \(i\) and \(j\) (in positions \(k\) and \(k + 1\)) are also early jobs. After interchange, in schedule \(\Pi_y\), 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 \(\Pi_x, \Pi_y\) are

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

We now derive the condition under which \(Z(\Pi_x) < Z(\Pi_y)\). 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 \(X \geq 0\) when the following condition is satisfied:

\[
\frac{p_i}{\alpha_i} \geq \frac{p_j}{\alpha_j}.
\]

From the above condition, we see that if \(X > 0\), then schedule \(\Pi_x\) is better than schedule \(\Pi_y\); 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 \(\frac{p_i}{\alpha_i} \geq \frac{p_j}{\alpha_j}\).

Based on this analysis, we state the following property.

**Property 1.** In schedule \(\Pi_x\), 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 \(\Pi_x\) is better than schedule \(\Pi_y\), only when \(\frac{p_i}{\alpha_i} \geq \frac{p_j}{\alpha_j}\).

**Conjecture.** Now, we discuss a special case when \(d_i = A + p_i + p_j\). Here in the schedule \(\Pi_x\) jobs \(i\) and \(j\) (in positions \(k\) and \(k + 1\)) are early jobs. After interchange, in schedule \(\Pi_y\), 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 \(\frac{p_i}{\alpha_i} \geq \frac{p_j}{\alpha_j}\).

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

Fig. 1 is a pictorial representation of the statuses of two adjacent jobs \(i\) and \(j\) in schedule \(\Pi_x\). Fig. 1 is in \((d_i, d_j)\) plane. For any two adjacent jobs \(i\) and \(j\), we know the values of \(A, p_i, p_j, a_i, d_i, a_j, d_j\). 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 \(\Pi_x\) 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 \(\Pi_y\), 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 \(\Pi_x\) has to be better than schedule \(\Pi_y\), is also given. Note that this property is true only when \(d_i > A + p_j\), \(d_j > A + p_j\), \(d_i > A + p_i + p_j\), and \(d_j > A + p_i + p_j\). 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 \(\Pi_y\), 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 \(\Pi_x\) jobs \(i\) and \(j\) (in positions \(k\) and \(k + 1\)) are early jobs. After interchange, in schedule \(\Pi_y\), 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 \(\Pi_x, \Pi_y\) are

\[
Z(\Pi_x) = G_1 + G_2 + \alpha_i(d_i - A - p_i) + \alpha_j(d_j - A - p_i - p_j),
Z(\Pi_y) = G_1 + G_2 + \alpha_i(d_i - A - p_j) + \alpha_j(A + p_j + p_i - d_i).
\]
We now derive the condition under which $Z(P_x) < Z(P_y)$. For this purpose, we obtain the value of $Z(P_y)/C_0 - Z(P_x)/C_0$. Let $X = Z(P_y)/C_0 - Z(P_x)/C_0$ and is given by

$$X = a_jp_i + b_ip_j + (a_i + b_i)(A + p_i - d_i).$$

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

$$d_i \leq \left( A + p_i \left( \frac{a_i + b_i + a_j}{a_i + b_i} \right) + p_j \left( \frac{b_j}{a_i + b_i} \right) \right).$$

From the above condition, if $X > 0$, then schedule $P_x$ is better than schedule $P_y$; i.e., $Z(P_x) < Z(P_y)$. If $X = 0$ then $Z(P_x) = Z(P_y)$. For this case, job $i$ will come before job $j$ only when $d_i \leq \left\{ A + p_i \left( \frac{a_i + b_i + a_j}{a_i + b_i} \right) + p_j \left( \frac{b_j}{a_i + b_i} \right) \right\}$. Based on this analysis, the following property exists.

**Property 2.** In schedule $P_x$, 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 $P_x$ is better than schedule $P_y$ only when $d_i \leq \left\{ A + p_i \left( \frac{a_i + b_i + a_j}{a_i + b_i} \right) + p_j \left( \frac{b_j}{a_i + b_i} \right) \right\}$. Based on this analysis, the following property exists.

**Conjecture.** Now, we discuss a special case when $d_i = (A + p_i + p_j)$. Here in schedule $P_x$ job $i$ and $j$ (in positions $k$ and $k + 1$) are early jobs. After interchange, in schedule $P_y$, 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 $d_j > (A + p_i)$, $d_j > (A + p_j)$, $d_j = (A + p_i + p_j)$ and $d_j > (A + p_i + p_j)$. Here also, job $i$ will come before job $j$ only when $\frac{d_j}{x_i} \geq \frac{d_j}{x_j}$. This can be easily proved by considering the fact that $d_j = (A + p_i + p_j)$ in the above analysis.
Note that this property is true only when \( d_i > (A + p_i) \), \( d_j > (A + p_j) \), \( d_i < (A + p_i + p_j) \), 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., \( \frac{p_i}{x_i} \geq \frac{p_j}{x_j} \), is derived.

**Status 2:** Consider two adjacent tardy jobs \( i \) (in position \( k \)) and \( j \) (in position \( k + 1 \)) in the schedule \( \Pi_x \). 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 two adjacent jobs are tardy means that \( d_i < (A + p_i + p_j) \). After interchange, in schedule \( \Pi_x \) and \( \Pi_y \) are as given below

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

**Possibility (i).** Here in schedule \( \Pi_x \), jobs \( i \) and \( j \) (in positions \( k \) and \( k + 1 \)) are tardy jobs. After interchange, in schedule \( \Pi_y \), the jobs \( j \) and \( i \) (in positions \( k \) and \( k + 1 \)) are also tardy jobs. This means that \( d_i < (A + p_i) \), \( d_j < (A + p_i) \) 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 \( \Pi_x \), \( \Pi_y \) are

\[
Z(\Pi_x) = G_1 + G_2 + \beta_i(A + p_i - d_i) + \beta_j(A + p_j + p_j - d_j),
\]

\[
Z(\Pi_y) = G_1 + G_2 + \beta_j(A + p_j - d_j) + \beta_i(A + p_j + p_i - d_i).
\]

We now derive the condition under which \( Z(\Pi_x) \leq Z(\Pi_y) \). 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

\[
X = -\beta_i p_i + \beta_j p_j
\]

Form the above expression, we see that \( X \geq 0 \) when the following condition is satisfied:

\[
\frac{p_i}{\beta_i} \leq \frac{p_j}{\beta_j}
\]

Form the above condition, if \( X > 0 \), then schedule \( \Pi_x \) is better than schedule \( \Pi_y \); 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 \( \frac{p_i}{\beta_i} \leq \frac{p_j}{\beta_j} \). Based on this analysis, we state the following property.

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

**Conjecture.** Now, we discuss a special case when \( d_j = (A + p_j) \). Here in the schedule \( \Pi_x \), jobs \( i \) and \( j \) (in positions \( k \) and \( k + 1 \)) are tardy jobs. After interchange, in schedule \( \Pi_y \), the job \( j \) (in position \( k \)) 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 \( \frac{p_i}{\beta_i} \leq \frac{p_j}{\beta_j} \). 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 \( \Pi_y \), 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 \( \Pi_x \), jobs \( i \) and \( j \) (in positions \( k \) and \( k + 1 \)) are tardy jobs. After interchange, in schedule \( \Pi_y \), 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 \( \Pi_x, \Pi_y \) are

\[
Z(\Pi_x) = G_1 + G_2 + \beta_i(A + p_i - d_i) + \beta_j(A + p_j + p_j - d_j),
\]

\[
Z(\Pi_y) = G_1 + G_2 + \beta_j(A + p_i - d_i) + \beta_i(A + p_j + p_i - d_i).
\]
We now derive the condition under which \( Z(P_x) \leq Z(P_y) \). For this purpose, we obtain the value of \( Z(P_y)/C_0 - Z(P_x)/C_0 \) and is given by

\[
X = -\beta_x p_i + \beta_y p_j + (x_j + \beta_j)(d_j - A - p_j).
\]

From the above expression, we see that \( X \geq 0 \) when the following condition is satisfied:

\[
d_j \geq \left\{ A + \frac{p_j(x_j + \beta_j - \beta_i) + \beta_j p_j}{(x_j + \beta_j)} \right\}.
\]

From the above condition, we see that if \( X > 0 \), then the schedule \( P_x \) is better than the schedule \( P_y \); i.e., \( Z(P_x) < Z(P_y) \). If \( X = 0 \) then \( Z(P_x) = Z(P_y) \). For this case, job \( i \) will come before job \( j \) only when

\[
d_j \geq \left\{ A + \frac{p_j(x_j + \beta_j - \beta_i) + \beta_j p_j}{(x_j + \beta_j)} \right\}.
\]

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.
1. for $i = 1$ to $k$
2. set $initialSolution \leftarrow $ Random_Solution ()
3. set $Population(k) \leftarrow $ DominanceProperties ($initialSolution$)
4. 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

1. calculateFinishTime()
2. set $preSchedule \leftarrow $ schedule
3. for $i = 1$ to $Iter$
4. for $Pos = 1$ to 2
5. for $j = Pos$ to $n - 1$
6. $job1 = schedule[j]$
7. $job2 = schedule[j + 1]$
8. $isSatisfyProperty = $ checkProperty ($job1, job2$)
9. if ($isSatisfyProperty == false$)
10. swapTwoJobs($j, j + 1$);
11. updateFinishTime()
12. end if
13. $j \leftarrow j + 1$
14. end for
15. end for
16. if (isIdenticalSolution ($schedule, preSchedule$))
17. break;
18. else
19. set $preSchedule \leftarrow $ schedule
20. end if
21. 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 $i$th and $j$th 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.
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 \( U = [d_{\text{min}}, d_{\text{min}} + \rho P] \), where \( d_{\text{min}} = \text{MAX}(0, P(\tau - \rho/2)) \) and \( P = \sum_{j=1}^{n} p_j \). The two parameters \( \rho \) and \( \tau \) 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 (\( a_j \)) and tardiness (\( b_j \)) 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
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>
<thead>
<tr>
<th>Table 1</th>
<th>properties of optimal schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property number</td>
<td>Processing times and due dates</td>
</tr>
<tr>
<td>1</td>
<td>$d_i &gt; (A+p_i)$, $d_j &gt; (A+p_j)$; $d_i &gt; (A+p_j)$</td>
</tr>
<tr>
<td>2</td>
<td>$d_i &gt; (A+p_i)$, $d_j &gt; (A+p_j)$, $d_i \leq (A+p_i)$; $d_j \leq (A+p_j)$</td>
</tr>
<tr>
<td>3</td>
<td>$d_i &lt; (A+p_i)$, $d_j &lt; (A+p_j)$</td>
</tr>
<tr>
<td>4</td>
<td>$d_i &lt; (A+p_i)$, $d_j &lt; (A+p_j)$, $d_i \leq (A+p_i)$; $d_j \leq (A+p_j)$</td>
</tr>
<tr>
<td>5</td>
<td>$d_i &gt; (A+p_i)$, $d_j &lt; (A+p_j)$</td>
</tr>
<tr>
<td>6</td>
<td>$d_i &gt; (A+p_i)$, $d_j &lt; (A+p_j)$, $d_i \leq (A+p_i)$; $d_j \leq (A+p_j)$</td>
</tr>
<tr>
<td>7</td>
<td>$d_i &gt; (A+p_i)$, $d_j &lt; (A+p_j)$</td>
</tr>
<tr>
<td>8</td>
<td>$d_i &gt; (A+p_i)$, $d_j &lt; (A+p_j)$</td>
</tr>
<tr>
<td>9</td>
<td>$d_i \leq (A+p_i)$, $d_j &gt; (A+p_j)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Average CPU seconds of different algorithms on each instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job sets</td>
<td>DP</td>
</tr>
<tr>
<td>20</td>
<td>0.021073</td>
</tr>
<tr>
<td>30</td>
<td>0.084101</td>
</tr>
<tr>
<td>40</td>
<td>0.284798</td>
</tr>
<tr>
<td>50</td>
<td>0.682626</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 3</th>
<th>Parameter values used in SGA and GADP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crossover operator</td>
<td>Two-point crossover</td>
</tr>
<tr>
<td>Crossover rate</td>
<td>0.8</td>
</tr>
<tr>
<td>Mutation operator</td>
<td>Swap mutation</td>
</tr>
<tr>
<td>Mutation rate</td>
<td>0.3</td>
</tr>
<tr>
<td>Population size</td>
<td>100</td>
</tr>
<tr>
<td>Generations</td>
<td>1000</td>
</tr>
<tr>
<td>Clone strategy</td>
<td>Swap mutation</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Instance</th>
<th>DP</th>
<th>SGA objective value</th>
<th>GADP objective value</th>
<th>B&amp;B</th>
<th>CPU time (Avg)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>Mean</td>
<td>Max</td>
<td>Min</td>
<td>Mean</td>
</tr>
<tr>
<td>sks222a</td>
<td>5298</td>
<td>5362</td>
<td>5393</td>
<td>5286</td>
<td>5291</td>
</tr>
<tr>
<td>sks223a</td>
<td>4546</td>
<td>4546</td>
<td>4546</td>
<td>4442</td>
<td>4442</td>
</tr>
<tr>
<td>sks224a</td>
<td>3840</td>
<td>3840</td>
<td>3840</td>
<td>3840</td>
<td>3840</td>
</tr>
<tr>
<td>sks225a</td>
<td>3989</td>
<td>4089.5</td>
<td>4113</td>
<td>3958</td>
<td>3958.6</td>
</tr>
<tr>
<td>sks226a</td>
<td>3053</td>
<td>3053</td>
<td>3053</td>
<td>3020</td>
<td>3020</td>
</tr>
<tr>
<td>sks227a</td>
<td>2048</td>
<td>2048</td>
<td>2048</td>
<td>2001</td>
<td>2030.8</td>
</tr>
<tr>
<td>sks228a</td>
<td>2111</td>
<td>2193.9</td>
<td>2199</td>
<td>2085</td>
<td>2085</td>
</tr>
<tr>
<td>sks232a</td>
<td>4357</td>
<td>4357</td>
<td>4357</td>
<td>4319</td>
<td>4320.4</td>
</tr>
<tr>
<td>sks233a</td>
<td>4411</td>
<td>4578.2</td>
<td>4628</td>
<td>4411</td>
<td>4411</td>
</tr>
<tr>
<td>sks234a</td>
<td>5076</td>
<td>5076</td>
<td>5076</td>
<td>5060</td>
<td>5062.8</td>
</tr>
<tr>
<td>sks235a</td>
<td>3183</td>
<td>3217.1</td>
<td>3219</td>
<td>3118</td>
<td>3118</td>
</tr>
<tr>
<td>sks236a</td>
<td>2887</td>
<td>2927.8</td>
<td>2932</td>
<td>2806</td>
<td>2801.7</td>
</tr>
<tr>
<td>sks237a</td>
<td>1873</td>
<td>1873</td>
<td>1873</td>
<td>1809</td>
<td>1809</td>
</tr>
<tr>
<td>sks238a</td>
<td>1915</td>
<td>1968.3</td>
<td>1972</td>
<td>1872</td>
<td>1872</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Job</th>
<th>Instance</th>
<th>SGA objective value</th>
<th>GADP objective value</th>
<th>DP</th>
<th>SGA</th>
<th>GADP</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>sks272a</td>
<td>6420</td>
<td>6663.3</td>
<td>7556</td>
<td>6420</td>
<td>6445.1</td>
</tr>
<tr>
<td></td>
<td>sks273a</td>
<td>8007</td>
<td>8488</td>
<td>9277</td>
<td>8007</td>
<td>8074.7</td>
</tr>
<tr>
<td></td>
<td>sks274a</td>
<td>6030</td>
<td>6505.8</td>
<td>6948</td>
<td>6030</td>
<td>6095.4</td>
</tr>
<tr>
<td></td>
<td>sks275a</td>
<td>9513</td>
<td>9721</td>
<td>10437</td>
<td>9513</td>
<td>9513</td>
</tr>
<tr>
<td></td>
<td>sks276a</td>
<td>3526</td>
<td>3975.6</td>
<td>5014</td>
<td>3526</td>
<td>3612.8</td>
</tr>
<tr>
<td></td>
<td>sks277a</td>
<td>7066</td>
<td>7155</td>
<td>8785</td>
<td>7066</td>
<td>6135.1</td>
</tr>
<tr>
<td></td>
<td>sks278a</td>
<td>7700</td>
<td>8236.2</td>
<td>9765</td>
<td>7700</td>
<td>7701.3</td>
</tr>
<tr>
<td>30</td>
<td>sks372a</td>
<td>20,917</td>
<td>22,860</td>
<td>24,935</td>
<td>20,818</td>
<td>21,066</td>
</tr>
<tr>
<td></td>
<td>sks373a</td>
<td>13,557</td>
<td>14,491</td>
<td>16,558</td>
<td>13,465</td>
<td>13,535</td>
</tr>
<tr>
<td></td>
<td>sks374a</td>
<td>12,171</td>
<td>14,371</td>
<td>18,357</td>
<td>12,171</td>
<td>12,610</td>
</tr>
<tr>
<td></td>
<td>sks375a</td>
<td>13,686</td>
<td>14,503</td>
<td>16,599</td>
<td>13,662</td>
<td>13,825</td>
</tr>
<tr>
<td></td>
<td>sks376a</td>
<td>12,718</td>
<td>13,917</td>
<td>16,381</td>
<td>12,594</td>
<td>12,834</td>
</tr>
<tr>
<td></td>
<td>sks377a</td>
<td>10,300</td>
<td>11,394</td>
<td>13,380</td>
<td>10,206</td>
<td>10,236</td>
</tr>
<tr>
<td></td>
<td>sks378a</td>
<td>14,213</td>
<td>14,812</td>
<td>15,910</td>
<td>13,988</td>
<td>13,996</td>
</tr>
<tr>
<td>40</td>
<td>sks472a</td>
<td>31,841</td>
<td>34,300</td>
<td>40,461</td>
<td>31,426</td>
<td>31,692</td>
</tr>
<tr>
<td></td>
<td>sks473a</td>
<td>31,832</td>
<td>33,742</td>
<td>36,054</td>
<td>31,474</td>
<td>31,675</td>
</tr>
<tr>
<td></td>
<td>sks474a</td>
<td>29,487</td>
<td>31,866</td>
<td>35,967</td>
<td>28,688</td>
<td>28,761</td>
</tr>
<tr>
<td></td>
<td>sks475a</td>
<td>27,433</td>
<td>31,348</td>
<td>37,175</td>
<td>24,904</td>
<td>25,566</td>
</tr>
<tr>
<td></td>
<td>sks476a</td>
<td>22,907</td>
<td>25,399</td>
<td>30,520</td>
<td>21,983</td>
<td>22,136</td>
</tr>
<tr>
<td></td>
<td>sks477a</td>
<td>26,884</td>
<td>29,945</td>
<td>34,366</td>
<td>26,548</td>
<td>26,635</td>
</tr>
<tr>
<td></td>
<td>sks478a</td>
<td>19,111</td>
<td>22,625</td>
<td>31,300</td>
<td>18,387</td>
<td>18,502</td>
</tr>
<tr>
<td>50</td>
<td>sks572a</td>
<td>50,929</td>
<td>54,373</td>
<td>57,042</td>
<td>49,387</td>
<td>49,705</td>
</tr>
<tr>
<td></td>
<td>sks573a</td>
<td>45,007</td>
<td>47,290</td>
<td>50,441</td>
<td>44,664</td>
<td>44,889</td>
</tr>
<tr>
<td></td>
<td>sks574a</td>
<td>42,715</td>
<td>47,354</td>
<td>53,077</td>
<td>40,998</td>
<td>41,371</td>
</tr>
<tr>
<td></td>
<td>sks575a</td>
<td>28,902</td>
<td>32,658</td>
<td>46,119</td>
<td>25,476</td>
<td>25,585</td>
</tr>
<tr>
<td></td>
<td>sks576a</td>
<td>34,389</td>
<td>37,516</td>
<td>41,303</td>
<td>32,565</td>
<td>33,151</td>
</tr>
<tr>
<td></td>
<td>sks577a</td>
<td>30,546</td>
<td>37,738</td>
<td>45,969</td>
<td>28,646</td>
<td>28,891</td>
</tr>
<tr>
<td></td>
<td>sks578a</td>
<td>49,851</td>
<td>54,063</td>
<td>59,652</td>
<td>49,087</td>
<td>49,207</td>
</tr>
</tbody>
</table>
value of \((n, \tau, p)\) 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 \(d_j < P\) for all the jobs, where \(P = \sum_{i=1}^{n} p_i\). When \(d_j \geq \sum_{i=1}^{n} p_i\) 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

| 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 \( z_j \), and tardiness penalty \( \beta_j \). 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 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 \( \Pi_x \) 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 \( \Pi_x \) is better than the schedule \( \Pi_y \) only when \( d_j > (A + \frac{p_i(z_i + \beta_i - \beta_j) + \beta_i p_j}{(A + \beta_i)}). \)

**Conjecture.** Now, we discuss a special case when \( d_j = (A + p_j) \). Here in the schedule \( \Pi_x \) jobs \( i \) and \( j \) (in position \( k \) and \( k + 1 \)) are tardy jobs. After interchange, in schedule \( \Pi_y \), 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 \( \frac{p_i}{\beta_i} \leq \frac{p_j}{\beta_j} \).

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 \( d_i < (A + p_i), d_i > (A + p_j), d_i < (A + p_i + p_j) \), 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., \( \frac{p_i}{\beta_i} \leq \frac{p_j}{\beta_j} \).

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

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

$$
Z(\Pi_x) = G_1 + G_2 + x_i(d_i - A - p_i) + \beta_j(A + p_i + p_j - d_i),
$$

$$
Z(\Pi_y) = G_1 + G_2 + x_j(d_j - A - p_j) + \beta_i(A + p_j + p_i - d_j).
$$

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

$$
X = (x_j + \beta_j)(d_j - p_j - A) + \beta_p - (x_i + \beta_i)(d_i - p_i - A) - \beta_p.
$$

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

$$(x_j + \beta_j)(d_j - p_j - A) + \beta_p - (x_i + \beta_i)(d_i - p_i - A) - \beta_p.$$ 

From the above condition, we see that if $X > 0$, then the schedule $\Pi_x$ is better than the schedule $\Pi_y$; 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 $i$ only when $(x_j + \beta_j)(d_j - p_j - A) + \beta_p - (x_i + \beta_i)(d_i - p_i - A) - \beta_p$. Based on this analysis, we state the following property.

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

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

**Possibility (ii).** Here, in schedule $\Pi_x$ job $i$ (in position $k$) is an early job and job $j$ (in position $k+1$) is a tardy job. After interchange, in schedule $\Pi_y$ 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 $\Pi_x$ and $\Pi_y$ are:

$$
Z(\Pi_x) = G_1 + G_2 + x_i(d_i - A - p_i) + \beta_j(A + p_i + p_j - d_i),
$$

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

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

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

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

$$
d_i \leq A + \frac{p_j(x_i + \beta_i - \beta_j)}{(x_i + \beta_i)}. $$

From the above condition, we see that if $X > 0$, then schedule $\Pi_x$ is better than schedule $\Pi_y$; 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 $d_i \leq A + \frac{p_j(x_i + \beta_i - \beta_j)}{(x_i + \beta_i)}$. Based on this analysis, we state the following property.
Property 6. In schedule $P_x$, for two adjacent jobs $i$ (in position $k$) an early job, and $j$ (in position $k+1$) a tardy job, and if $d_i < (A + p_i)$ and $d_i < (A + p_i + p_j)$, then schedule $P_x$ is better than schedule $Z(P_i)$ only when $d_i \leq \{ A + \frac{p_j(x_j + \beta_j - \beta_i)}{(x_j + \beta_j)} \}$.

Conjecture. Now, we discuss a special case when $d_j = (A + p_j)$. Here in the schedule $P_x$ jobs $i$ (in position $k$) an early job and job $j$ (in position $k+1$) is a tardy job. After interchange, in schedule $P_x$, the job $j$ (in position $k$) is an on time job, and job $i$ (in position $k+1$) is a tardy job. This means that $d_i > (A + p_i)$, $d_j < (A + p_j)$, and $d_j < (A + p_i + p_j)$. Here also, job $i$ will come before job $j$ only when $d_i \leq \{ A + \frac{p_j(x_j + \beta_j - \beta_i)}{(x_j + \beta_j)} \}$. 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 $d_i > (A + p_i)$, $d_j < (A + p_j)$, and $d_j < (A + p_i + p_j)$ 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 $P_x$ jobs $i$ (in position $k$) is an early job and job $j$ (in position $k+1$) is a tardy job. After interchange, in schedule $P_i$, both jobs $j$ (in position $k$) and $i$ (in position $k+1$) are early jobs. This means that $d_i > (A + p_j)$, $d_j < (A + p_i + p_j)$ and $d_j < (A + p_i + p_j)$. The total absolute deviation $Z(P_x)$, $Z(P_y)$ for schedules $P_x, P_y$ are

$$Z(P_x) = G_1 + G_2 + x_i(d_i - A - p_i) + \beta_j(A + p_i + p_j - d_i),$$
$$Z(P_y) = G_1 + G_2 + x_j(d_j - A - p_j) + \alpha_i(d_i - A - p_i - p_j).$$

We now derive the condition under which $Z(P_x) \leq Z(P_y)$. For this purpose, we obtain the value of $Z(P_x) - Z(P_y)$ and is given by

$$X = (x_i + \beta_j)(d_j - A) - p_j(x_i + \beta_i + x_j) - \beta_i p_j.$$  

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

$$d_j \geq \left\{ A + \frac{p_j(x_j + \beta_j + x_i) + \beta_j p_i}{(x_j + \beta_j)} \right\}.$$ 

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

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

Conjecture. Now, we discuss a special case when $d_i = (A + p_i + p_j)$. Here in the schedule $P_x$ 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 $P_y$, 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 $d_i > (A + p_i)$, $d_j > (A + p_j)$, $d_j = (A + p_i + p_j)$ and $d_j < (A + p_i + p_j)$. Here also, job $i$ will come before job $j$ only when $d_j \geq \{ A + \frac{p_j(x_j + \beta_j + x_i) + \beta_j p_i}{(x_j + \beta_j)} \}$. 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 $d_i > (A + p_i)$, $d_i > (A + p_j)$, $d_i = (A + p_i + p_j)$ 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 $P_x$ job $i$ (in position $k$) is an early job and job $j$ (in position $k+1$) is a tardy job. After interchange, in schedule $P_y$ 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 \( \Pi_x, \Pi_y \) are

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

We now derive the condition under which \( Z(\Pi_y) \leq Z(\Pi_x) \). 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

\[
X = -p_j\beta_i - p_i\alpha_i.
\]

From the above expression, we see that \( X = -p_j\beta_i - p_i\alpha_i \), which implies that schedule \( \Pi_y \) is always better than schedule \( \Pi_x \). Based on this analysis, we state the following property.

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

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 \( R_y \), in which this property is true, is shown in Fig. 1.

**Status 4:** Here, in \( \Pi_y \) job \( i \) (in position \( k \)) is a tardy job and job \( j \) (in position \( k + 1 \)) is an early job. After interchange, in \( \Pi_x \) 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 \( \Pi_x, \Pi_y \) are

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

We now derive the condition under which \( Z(\Pi_y) \leq Z(\Pi_x) \). 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 \( \Pi_x \) is always better than schedule \( \Pi_y \). Based on this analysis, we state the following property.

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

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 \( R_x \), 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 \( k + 1 \)) an on time job, in schedule \( \Pi_x \). 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 \( \Pi_x \) job (in position \( k \)) is an early job, and \( j \) (in position \( k + 1 \)) is an on time job. After interchange, in the schedule \( \Pi_y \), 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 \( \Pi_y \), 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 \( \Pi_x \) is better than schedule \( \Pi_y \) only when \( \frac{p_i}{\alpha_i} \geq \frac{p_j}{\beta_j} \).
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 \( d_j = (A + p_j + p_i). \) In Fig. 1, it is a point on the boundary between region \( R_1 \) and region \( R_7. \)

Possibility (ii). Here in the schedule \( \Pi_x, \) 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 \( \Pi_y, \) 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 \( d_i = (A + p_i + p_j). \) Hence, we obtain the same property as in Property 2, for this possibility also.

**Property 11.** In schedule \( \Pi_x, \) 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 the schedule \( \Pi_x \) is better than the schedule \( \Pi_y \) only when \( \ldots \)