Comp. Appl. Math.
DOI 10.1007/s40314-013-0081-z

Improved estimation of distribution algorithm for the problem of single-machine scheduling with deteriorating jobs and different due dates

Hua-Ping Wu · Min Huang

Received: 22 April 2013 / Revised: 21 September 2013 / Accepted: 23 September 2013

Abstract This paper investigates single-machine scheduling problem, which is an NP-hard problem, with deteriorating jobs and different due dates to minimize total tardiness. First, two special polynomially solvable cases of the problem and a mixed-integer programming (MIP) model are proposed. Since the large-scale problem needs a long time when the MIP is solved using the CPLEX, the improved estimation of distribution algorithm (EDA) is proposed to solve the problem with a large size. EDA depends on the probabilistic model, which denotes the distribution of decision variables in the feasible region space. Meanwhile, EDA owns efficient search capability and convergence. To obtain an improved initial population, an efficient initialization scheme based on the feature of two special cases and a heuristic algorithm are adopted in the process of constructing the initial population. The probabilistic model is composited based on elite solutions from each generation. Simultaneously, mutation is embedded to maintain the diversity of the population. Compared with the results, numerical experiments show that the proposed algorithm can obtain good near-optimal solutions within a short period.

Keywords Single machine · Deteriorating jobs · Tardiness · Estimation of distribution algorithm · Mixed integer programming model

Mathematics Subject Classification 00A69

Communicated by José Mario Martínez.

H.-P. Wu (✉) · M. Huang
State Key Laboratory of Synthetical Automation for Process Industries (Northeastern University), College of Information Science and Engineering, Northeastern University, Shenyang 110819, Liaoning, China
e-mail: wuhuaping2010@126.com

M. Huang
e-mail: mhuang@mail.neu.edu.cn

Published online: 06 October 2013

1 Introduction

In classical deterministic scheduling problems, the processing times of jobs are assumed to be constant values. However, in real production processing, a job processed later needs more time than the same job when processed earlier. For example, in steel production, heating ingots is necessary. The main purpose of it is to soften the ingots for rolling and providing a sufficiently high initial temperature so that rolling processing is insured to complete in a required temperature region. But, the temperate of ingots, while waiting to enter the rolling machine, drops below a certain level. At this time, they will require to be reheated, and thus they need a longer processing time before rolling. Scheduling in these setting is known as scheduling deterioration jobs.

Deterioration jobs scheduling problem was first introduced independently by Gupta and Gupta (1988) and by Browne and Yechiali (1990). Since then, related models have been extensively studied from variety of perspectives. For instance, Bachman et al. (2002) proved that minimizing the total weighted completion time is NP hard when the processing time is a linear function. Cheng and Ding (2001) considered a piecewise-linear model where each task has a normal processing time that deteriorates as a step function if its starting time is beyond a given deterioration date. And they mainly proved the flow time problem is NP complete and gave a pseudo-polynomial algorithm for the makespan problem. Cheng et al. (2010) considered a new deterioration model where the actual job processing time of a job is a function of the processing times of the jobs already processed and proved that the problems to minimize the total weighted completion time, maximum lateness, and maximum tardiness are polynomially solvable. Ozturkoglu and Bulfin (2011) gave a unique integer program to solve the problem of makespan and total completion time with deterioration jobs. Lai and Lee (2010) investigated a new deterioration model in which the actual job processing time is a general function of the normal processing times of jobs already processed and its scheduled position. They showed that certain objectives including makespan, total completion time, total weighted completion time, total tardiness and maximum tardiness under the proposed model remain polynomially solvable. Following it, Lai et al. (2011) studied the scheduling problem with deteriorating jobs and past-sequence-dependent setup times. Farahani and Hosseini (2013) addressed the actual processing time of each job as a piecewise linear function of its start time and the objective is to minimize the cycle time. Moreover, in many realistic scheduling situations, since the effects of learning and deterioration co-exist, others considered the scheduling problem with the effects of learning and deterioration simultaneously. For instance, Yin and Xu (2011) proposed a general scheduling model with the effects of learning and deterioration simultaneously, in which job processing times are defined by functions of their start times and positions in the sequence. Based on it, they proved that the single-machine scheduling problems for minimizing makespan, sum of the kth power of completion times, total lateness and sum of earliness penalties (with a common due date) are polynomially solvable. Meanwhile, they also showed that the problem under special conditions for minimizing the total weighted completion time, discounted total weighted completion time, maximum lateness, maximum tardiness, total tardiness and weighted earliness penalties (with a common due date) are polynomially solvable.

Before Du and Leung (1990), the problem of minimizing the total tardiness had been considered in some prior research. But, Du and Leung (1990) firstly proved that it is NP complete problem to minimize the total tardiness problem with different due dates and processing times for all jobs and gave a pseudo-polynomial algorithm for solving the problem. Following it, many literatures consider the related problem from the different aspects. Holsenback and Russell (1992) presented a heuristic algorithm Net Benefit of Relocation (NBR) for solving the problem, which is better than other heuristic algorithms. Later, Russell and Holsenback

(1997) proposed M-NBR through expanding the NBR for the problem. Ben-Daya and Al-Fawzan (1996) proposed a simulated annealing approach for the problem. Later, lots of extended problems are also studied in recent years. Bozejko (2010) used the parallel path relinking method for the single-machine total weighted tardiness problem with sequence-dependent setups. Mazdeh et al. (2010) developed a linear programming formulation and three algorithms for solving the problem of minimizing total weighted tardiness with drop dead dates in single-machine scheduling. Akker et al. (2010) studied the problem of minimizing total weighted tardiness on a single machine with release dates and equal-length jobs and formulate it as a time-indexed integer linear programming. And then they showed several special cases for the problem can be solved in polynomial time. Akturk and Ilhan (2011) proposed a DP-based heuristic to solve the problem of single-computer numerical-controlled machines scheduling with controllable processing times to minimize total weighted tardiness. Kirlik and Oguz (2012) used a variable neighborhood search for minimizing total weighted tardiness with sequence-dependent setup times on a single machine.

In recent years, some papers mentioned minimizing the total tardiness with deteriorating jobs. Toksan and Guner (2009) considered a parallel machine earliness/tardiness scheduling problem with different penalties with learning and deterioration under the condition that all jobs have common due date. They proposed a mathematical model for the problem and algorithm and lower bound procedure to solve larger test problems. Meng et al. (2012) considered the problem of scheduling deteriorating jobs with a common due window on a single machine and propose an optimal algorithm—CDW for solving the problem of minimizing costs for earliness, tardiness, earliest due date assignment and due window size penalties.

However, the above-mentioned studies assume that the due dates of jobs are common or have some special conditions. This paper studies the single-machine deteriorating jobs scheduling problem with different due dates when the objective is minimizing the total tardiness. Since the problem with normal processing times of jobs has been proved to be NP complete by Du and Leung (1990), the problem with deteriorating jobs is still NP complete.

Our contribution is that: we consider the real problem in steel industry as the special problem of single-machine deteriorating jobs with different due date; a mixed integer programming model is given; and an improved estimation of distribution algorithm (EDA) is addressed to find good near optimal solutions for the large size of the problem.

The rest of this paper is organized as follows. In Sect. 2, problem is described and a mixed integer programming model is given. Then, two special cases that can be solved in polynomial time are considered in Sect. 3, namely, the case with common normal processing times and the case with identical due dates. In Sect. 4, an improved EDA is proposed for solving the problem in this paper. Section 5 is the numerical experiments, followed by the conclusion in the last section.

2 Problem statements and a mixed integer programming model

This paper studies the single-machine deteriorating jobs scheduling problem, which is NP complete, with different due dates when the objective is minimizing the total tardiness. This problem can be described in Sect. 2.1. And followed by a mixed integer programming model for the problem proposed in Sect. 2.2.

2.1 Problem statement

Assume that there are n jobs required to be scheduled. The normal processing time and due date for job j ($j \in J = \{1, 2, \ldots, n\}$) is $p_{[j]}$ and d_j, respectively. All jobs have a

common deteriorating rate b. The actual processing time p_j of job j is a function of $p_{[j]}$, b and its starting time s_j, i.e., $p_j = p_{[j]} + bs_j$, which is proposed by Cheng and Ding (1998). The completion time of each job j is denoted as C_j. The objective is to find an optimal schedule to minimize the total tardiness. For convenience, the three-field notation scheme $\alpha\,|\beta\,|\gamma$ introduced by Graham et al. (1979) is used to denote this problem, i.e., $1\left|p_j = p_{[j]} + bs_j\right|\sum_{j=1}^{n} T_j$, where $\sum_{j=1}^{n} T_j$ denotes the total tardiness.

2.2 A mixed integer programming model

The mixed integer programming model of the problem in this paper is as follows.

Objective function

$$\text{Min} \sum_{k=1}^{n} T_{[k]}$$

s.t.

$$z_{jk} = \begin{cases} 1, & \text{if job } j \text{ is scheduled at position } k; \\ 0, & \text{Otherwise.} \end{cases} \quad j = 1, 2, ..., n, k = 1, 2, ..., n. \tag{1}$$

$$\sum_{j=1}^{n} z_{jk} = 1, \quad k = 1, 2, ..., n. \tag{2}$$

$$\sum_{k=1}^{n} z_{jk} = 1, \quad j = 1, 2, ..., n. \tag{3}$$

$$p_{[k]} = \sum_{j=1}^{n} z_{jk} a_j, \quad k = 1, 2, ..., n. \tag{4}$$

$$d_{[k]} = \sum_{j=1}^{n} z_{jk} d_j, \quad k = 1, 2, ..., n. \tag{5}$$

$$s_{[k]} = 0, \quad k = 1. \tag{6}$$

$$C_{[k]} = (1 + b)s_{[k]} + p_{[k]}, \quad k = 1, 2, ..., n. \tag{7}$$

$$s_{[k]} \geq C_{[k-1]}, \quad k = 2, 3, \ldots n. \tag{8}$$

$$T_{[k]} = \begin{cases} C_{[k]} - d_{[k]}, & C_{[k]} > d_{[k]}; \\ 0, & \text{Otherwise.} \end{cases} \quad k = 1, 2, ..., n. \tag{9}$$

The above parameters and variables used in the model are described in the following paragraph, and followed by constraints explanation.

Parameters

j: The job index, $j = 1, 2, ..., n$;
k: The location index, $k = 1, 2, ..., n$;
a_j: The normal processing time of job, $a_j > 0$;
d_j: The due date of job, $d_j \geq = 0$;
b: The deterioration rate for all job, $b > 0$.

Decision variables

Z_{jk}:The value of Z_{jk} is 1 if job j is scheduled at position k to be processed; otherwise 0, $j = 1, 2, ..., n$, $k = 1, 2, ..., n$.

$p_{[k]}$: The normal processing time of job at position k, $k = 1, 2, ..., n$;
$d_{[k]}$: The due date of job at position k, $k = 1, 2, ..., n$;
$s_{[k]}$: The starting time of job at position k, $k = 1, 2, ..., n$;
$C_{[k]}$: The completion time of job at position k, $k = 1, 2, ..., n$;
$T_{[k]}$: The tardiness of job at position k, $k = 1, 2, ..., n$. The value of $T_{[k]}$ is $C_{[k]} - d_{[k]}$ if $C_{[k]} > d_{[k]}$; otherwise 0, $k = 1, 2, ..., n$.

Constraint (1) defines whether job j is scheduled at position k or not. Constraint (2) defines that each job can be scheduled only once. Constraint (3) denotes that only one job can be scheduled at position k. Constraints (4) and (5) describe the normal processing time and the due date of job at position k, respectively. Constraint (6) defines that the starting time of the first position job is equal to 0. Constraint (7) describes the completion time of job at position k. Constraint (8) indicates that the starting time of the k^{th} position job is greater than or equal to the previous jobs' completion time on the machine. Constraint (9) gives the tardiness of job at position k, namely, $T_{[k]}$ is equal to $C_{[k]} - d_{[k]}$ if $C_{[k]} > d_{[k]}$; otherwise 0.

3 Polynomially solvable cases

In this section, we consider two special cases for the problem of single-machine scheduling with deteriorating jobs and different due dates. One of the cases is the problem with common normal processing times, and the other is the problem with identical due dates. Both cases can be solved in polynomial time.

3.1 Common normal processing times

When all jobs have common processing time, the problem reduces to
$1 \left| p_j = p + bs_j \right| \sum_{j=1}^{n} T_j,\ p_{[j]} = p\ (\forall j \in J)$.

Theorem 1 *For the problem* $1 \left| p_j = p + bs_j \right| \sum_{j=1}^{n} T_j,\ p_{[j]} = p\ (\forall j \in J)$ *in which each job with common processing times and the objective of minimizing the total tardiness under the condition of deteriorating phenomenon, there exists an optimal schedule in which the optimal sequence can be obtained by sequencing the jobs in non-decreasing order of due date , i.e., the earliest due date* (*EDD*) *rule.*

Proof Assume that schedule S has two adjacent jobs J_i and J_j with job J_i immediately preceding job J_j, and J_i and J_j are separately in the kth and $(k+1)$th position. A new sequence S' is obtained from interchanging of jobs J_i and J_j. That is, $S = (\pi, J_i, J_j, \pi')$ and $S' = (\pi, J_j, J_i, \pi')$ where $d_i < d_j$, and π and π' are partial sequences. For convenience, assumed that the completion time of the last job in π is t, and Γ is the total tardiness of all jobs in π, it is suffice to show that S' does not dominate S, i.e., $\sum_{u=1}^{k+1} T_u(S) - \sum_{u=1}^{k+1} T_u(S') \leq 0$. For obtaining the result, let C_k be the completion times of previous k jobs, i.e., $C_k = t+p+bt$. Similarly, $C_{k+1} = p+(b+1)(t+p+bt)$, which denotes the completion times of previous $k+1$ jobs. Apparently, the completion time of a job is the completion time of the job immediately preceding it added to the actual processing time of it. Due to the identical processing times and releasing times for all jobs, the completion time of a job is only related to the completion time of the immediately previous job of it. So, for two schedule S and S', they have the same completion times of the jobs in positions k and $k+1$. According to the value of d_i, d_j, C_k and C_{k+1}, which $d_i < d_j$ and $C_k < C_{k+1}$, we separate it into two cases $C_k \leq d_i$ and $d_i < C_k$.

(1) $C_k \leq d_i$.

$C_k \leq d_i$, which implies the completion time of job in the kth position is equal to or less than the due date of job i. And $d_i < d_j$, then $C_k < d_j$. Thus, the job i will be not tardiness under the schedule S. And the tardiness of job j is also 0 under the schedule S'. Furthermore, three cases are considered depending on the values of d_i, d_j, C_k and C_{k+1} when $C_k \leq d_i$. Namely, $C_k < C_{k+1} < d_i < d_j$, $C_k < d_i < C_{k+1} < d_j$ and $C_k < d_i < d_j < C_{k+1}$.

i. $C_k < C_{k+1} < d_i < d_j$.

This situation denotes the smallest due date of two jobs i and j that is larger than the completion time of job in the $(k+1)$th position. Thus, the tardiness of the two jobs is 0 whether job i is in front of job j. The total tardiness under schedule S is the same as that of schedule S':

$$\sum_{u=1}^{k+1} T_u(S) = \sum_{u=1}^{k+1} T_u(S') = \Gamma.$$

So,

$$\sum_{u=1}^{k+1} T_u(S) - \sum_{u=1}^{k+1} T_u(S') = 0.$$

ii. $C_k < d_i < C_{k+1} < d_j$.

In a schedule S, the tardiness for job j is equal to 0 because the due date of job j is larger than the completion time of job in the position $k+1$. While the due date d_i is less than C_{k+1}, thus, $\sum_{u=1}^{k+1} T_u(S) = \Gamma$ and $\sum_{u=1}^{k+1} T_u(S') = \Gamma + C_{k+1} - d_i$. So, $\sum_{u=1}^{k+1} T_u(S) - \sum_{u=1}^{k+1} T_u(S') < 0$.

iii. $C_k < d_i < d_j < C_{k+1}$.

The completion time of job in the position $k+1$ is later than the due dates d_i and d_j. Thus, the job i in the schedule S and the job j in the schedule S' are tardiness, i.e.,

$$\sum_{u=1}^{k+1} T_u(S) = \Gamma + C_{k+1} - d_j,$$

$$\sum_{u=1}^{k+1} T_u(S') = \Gamma + C_{k+1} - d_i.$$

Since $d_i < d_j$, then $\sum_{u=1}^{k+1} T_u(S) - \sum_{u=1}^{k+1} T_u(S') < 0$.

(2) $d_i < C_k$.

In contrast with the first case, this represents that the completion time of job in the position k is later than the due date d_i. And $C_k < C_{k+1}$, then, the completion time of job in the position $k+1$ is also later than the due date d_i. So, the job i will be always tardiness whether it is scheduled in S or not. Moreover, it is also partitioned into three cases similar to the first one, i.e., $d_i < d_j < C_k < C_{k+1}$, $d_i < C_k < d_j < C_{k+1}$, and $d_i < C_k < C_{k+1} < d_j$.

i. $d_i < d_j < C_k < C_{k+1}$.

In this situation, in scheduling S and S', both jobs i and j are tardiness because their due dates are earlier than their completion time,

$$\sum_{u=1}^{k+1} T_u(S) = \Gamma + (C_k - d_i) + (C_{k+1} - d_j),$$

$$\sum_{u=1}^{k+1} T_u(S') = \Gamma + (C_k - d_j) + (C_{k+1} - d_i).$$

So, $\sum_{u=1}^{k+1} T_u(S) - \sum_{u=1}^{k+1} T_u(S') = 0$.
ii. $d_i < C_k < d_j < C_{k+1}$.

Apparently, both jobs i and j are tardiness in schedule S. While in schedule S' the only job i is tardiness. Namely,

$$\sum_{u=1}^{k+1} T_u(S) = \Gamma + (C_k - d_j) + (C_{k+1} - d_i),$$
$$\sum_{u=1}^{k+1} T_u(S') = \Gamma + (C_{k+1} - d_i).$$

Since $C_k < d_j$, then $\sum_{u=1}^{k+1} T_u(S) - \sum_{u=1}^{k+1} T_u(S') < 0$.
iii. $d_i < C_k < C_{k+1} < d_j$.
In the last situation, in schedule S and S', the job i is the tardiness. Thus,

$$\sum_{u=1}^{k+1} T_u(S) = \Gamma + (C_k - d_i),$$
$$\sum_{u=1}^{k+1} T_u(S') = \Gamma + (C_{k+1} - d_i).$$

Since $C_k < C_{k+1}$, then $\sum_{u=1}^{k+1} T_u(S) - \sum_{u=1}^{k+1} T_u(S') < 0$.
From the above analysis, we obtain S dominates S'.
Therefore, S dominates S' when $d_i < d_j$ and $p_i = p_j = p$. □

Theorem 1 shows that the single-machine deteriorating jobs scheduling problem with different due dates can be solved in polynomial time using EDD rule when each job has common processing time. Next, we consider the problem under the condition that all jobs own identical due dates.

3.2 Common due dates

If all jobs own identical due dates, the problem reduces to
$1 \left| p_j = p_{[j]} + bs_j \right| \sum_{j=1}^{n} T_j, d_j = d, \forall j \in J.$

Theorem 2 *For the problem* $1 \left| p_j = p_{[j]} + bs_j \right| \sum_{j=1}^{n} T_j$, $d_j = d, \forall j \in J$ *in which each job has different processing time, common due date d, and the objective of minimizing the total tardiness, there exists an optimal schedule in which the optimal sequence of jobs can be obtained by sequencing the jobs in non-decreasing order of processing time* p_j*, i.e., the shortest processing time* (*SPT*) *rule.*

Proof Here, the assumption is similar to the Theorem 1. There are two schedules $S = (\pi, J_i, J_j, \pi')$ and $S' = (\pi, J_j, J_i, \pi')$ where $p_{[i]} < p_{[j]}$, $d_i = d_j = d$, and π and π' are also partial sequences. To show that S dominates S', i.e., $\sum_{u=1}^{k+1} T_u(S) - \sum_{u=1}^{k+1} T_u(S') \leq 0$. Likewise, assumed that the position of the last job in π is $k-1$ and the completion time of

it in π is $t\,(t \geq 0)$, and Γ is the total tardiness of all jobs in π. Due to the starting time of the job in the k^{th} position being related to the completion time of the last job in π, the starting time of job J_i in schedule S is equal to t. Thus, the completion time of job J_i in S is:

$C_i(S) = p_{[i]} + b(t+1)$. Followed by J_i , the completion time of job J_j in S is: $C_j(S) = p_{[j]} + (b+1)p_{[i]} + t(b+1)^2$. Likewise, the completion time of job J_j in S' is: $C_j(S') = p_{[j]} + b(t+1)$. And the completion time of job J_i in S' is: $C_j(S') = p_{[i]} + (b+1)p_{[j]} + t(b+1)^2$.

Since $p_{[i]} < p_{[j]}$, then $C_i(S) < C_j(S)$, $C_i(S) < C_j(S')$, $C_i(S) < C_i(S')$, $C_j(S') < C_i(S')$ and $C_j(S) < C_i(S')$.

Based on the values of d, $C_i(S)$, $C_j(S)$, $C_j(S')$ and $C_i(S')$, we consider the following six cases. Namely, $d < C_i(S)$, $d > C_i(S')$, $C_i(S) < d < \min(C_j(S), C_j(S'))$, $\max(C_j(S), C_j(S')) < d < C_i(S')$, $C_j(S) < d < C_j(S')$ and $C_j(S') < d < C_j(S)$.

(1) $d < C_i(S)$.

When the due date d is less than the completion time of the job i in schedule S, it implies that the due date d is also less than $C_j(S)$, $C_j(S')$ and $C_i(S')$. In other words, two jobs i and j are tardiness in schedule S and S', thus,

$$\begin{aligned}\sum_{u=1}^{k+1} T_u(S) &= \Gamma + [C_i(S) - d] + [C_j(S) - d],\\ \sum_{u=1}^{k+1} T_u(S') &= \Gamma + [C_j(S') - d] + [C_i(S') - d].\end{aligned}$$

Due to $C_i(S) < C_j(S')$ and $C_j(S) < C_i(S')$, clearly, $\sum_{u=1}^{k+1} T_u(S) - \sum_{u=1}^{k+1} T_u(S') < 0$.

(2) $d > C_i(S')$.

If $d > C_i(S')$, It represents the due date d is always later than the completion times of the two jobs i and j in schedule S and S'. Thus, the two jobs will not be tardiness no matter how to schedule. Namely,

$$\begin{aligned}\sum_{u=1}^{k+1} T_u(S) &= 0,\\ \sum_{u=1}^{k+1} T_u(S') &= 0.\end{aligned}$$

Obviously, $\sum_{u=1}^{k+1} T_u(S) - \sum_{u=1}^{k+1} T_u(S') = 0$.

(3) $C_i(S) < d < \min(C_j(S), C_j(S'))$.

When $C_i(S) < d < \min(C_j(S), C_j(S'))$, the due date d is only larger than $C_i(S)$. For two jobs i and j, only the job i in the schedule S is not tardiness. Thus,

$$\begin{aligned}\sum_{u=1}^{k+1} T_u(S) &= \Gamma + [C_j(S) - d],\\ \sum_{u=1}^{k+1} T_u(S') &= \Gamma + [C_j(S') - d] + [C_i(S') - d].\end{aligned}$$

Since $C_j(S) < C_i(S')$ and $C_j(S') > d$, $\sum_{u=1}^{k+1} T_u(S) - \sum_{u=1}^{k+1} T_u(S') < 0$.

(4) $\max(C_j(S), C_j(S')) < d < C_i(S')$.

When the due date d is only less than $C_i(S')$, the two jobs i and j in schedule S and S' are not tardiness except for the job j in schedule S'. Thus,

$$\sum_{u=1}^{k+1} T_u(S) = 0,$$
$$\sum_{u=1}^{k+1} T_u(S') = \Gamma + [C_i(S') - d].$$

Apparently, $\sum_{u=1}^{k+1} T_u(S) - \sum_{u=1}^{k+1} T_u(S') < 0$.

(5) $C_j(S) < d < C_j(S')$.

In this case, the two jobs i and j in schedule S are not tardiness, while in schedule S' they are all tardiness. Thus,

$$\sum_{u=1}^{k+1} T_u(S) = 0,$$
$$\sum_{u=1}^{k+1} T_u(S') = \Gamma + [C_j(S') - d] + [C_i(S') - d].$$

Easily, we obtain $\sum_{u=1}^{k+1} T_u(S) - \sum_{u=1}^{k+1} T_u(S') < 0$.

(6) $C_j(S') < d < C_j(S)$.

In this case, the job i in schedule S and the job j in schedule S' are not tardiness. Namely,

$$\sum_{u=1}^{k+1} T_u(S) = \Gamma + [C_j(S) - d],$$
$$\sum_{u=1}^{k+1} T_u(S') = \Gamma + [C_i(S') - d].$$

Clearly, $\sum_{u=1}^{k+1} T_u(S) - \sum_{u=1}^{k+1} T_u(S') < 0$ because of $C_j(S) < C_i(S')$.

Summarily, S dominates S' when $p_{[i]} < p_{[j]}$ and $d_i = d_j = d$. □

Theorem 2 shows that the single-machine deteriorating jobs scheduling problem with different processing times can be solved in polynomial time using SPT rule when each job has common due date.

4 Improved estimation of distribution algorithm

EDA is firstly introduced by Mühlenbein and Paaß (1996), which follows the principles of natural evolution the same as genetic algorithm (GA). Differing from GA, EDA includes neither crossover nor mutation operators. Instead of them, it only owns a probability model, which is estimated from a group of elite individuals selected from previous generation, to produce new individuals at each generation. At the beginning of the algorithm, sample randomly M individuals from the search space to obtain an initial population P_0. Choose N $(N < M)$ elite individuals from P_0 through select operator to determine the joint probability model according to their distribution. And then M individuals are sampled depending on the probability

model and replace some individuals in the current generation to construct the new population at next generation. The above steps are repeated until the stopping criterion is satisfied.

In brief, the EDA is a population-based search algorithm based on probabilistic modelling of promising solutions in combination with the simulation of the induced models to guide their search. It has been successfully applied to multi-objective knapsack (Shah and Reed 2011), flowshop scheduling problems (Jarboui et al. 2009; Pan and Ruiz 2012), nurse rostering (Aickehin et al. 2007; Aickehin and Li 2007), resource-constrained project scheduling problem (Wang and Fang 2012) and others. It is important to stress that in most of these applications no other technique was shown to be capable of achieving better performance than EDA or solving problems of comparable size and complexity.

To obtain better solutions, the improved EDA is proposed in the following paragraphs.

4.1 Encoding scheme and fitness function

In EDA approach, solution representation and fitness function are necessarily the same as other evolution algorithms. Here, a chromosome (solution) consists of permutation of jobs orderly processed on single machine. For example, there are five jobs ($j = 1, 2, 3, 4, 5$) processed by single machine. Each feasible solution (chromosome) is represented as the permutation of $\{1, 2, 3, 4, 5\}$.

To converge on an appropriate solution, the fitness function is designed as one correlating closely with the objective. In this paper, the objective function is directly adopted as fitness function which is not only denoted as the goal itself also computed easily and quickly.

4.2 Population initialization

At the beginning of the algorithm, the initial population is composed of M individuals sampled from the global region. In order to make sure the quality of individuals and quickly converge to good solutions, two theorems from Sect. 3 and the NBR heuristic algorithm proposed by Holsenback and Russell (1992) are applied for sampling three individuals. For convenience, its steps are given in the end of this paper. Other $M - 3$ individuals of population P_0 will be sampled randomly from the search space.

4.3 Selection operator and probability model estimation

Selection operator generally includes sorting, roulette-wheel selection, random ergodic sampling, tournament selection and so on, in which roulette-wheel selection and tournament selection are usually used as selection operators in evolution algorithms. In this paper, sorting algorithm is adopted as selection operator, which orders the total tardiness of each solution in non-decreasing and then estimates probability model.

How to design probability model is the key procedure and has a great effect on the performance of EDA. Probability model pr_g $(g = 1, 2, \ldots)$ is derived from N elite individuals (first N individuals) chosen from the immediately previous population P_{g-1} according to selection operator, where g denotes the gth generation and N_{g-1} the number of individuals selected from P_{g-1}.

In this paper, probability model $pr_g(\mathbf{x})$ is given as follows.

$$pr_g(\mathbf{x}) = pr_g(\mathbf{x}|N_{g-1}) = \prod_{j=1}^{n} pr_g(x_j). \tag{10}$$

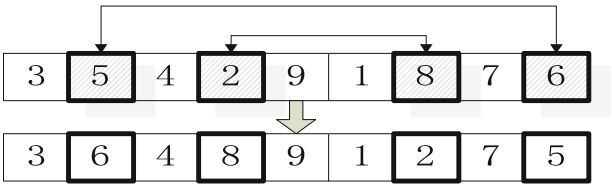

Fig. 1 The procedure of mutate operator

Each job marginal distribution can be estimated from marginal frequencies:

$$pr_g(x_{ji}) = \frac{\sum_{j=1}^{n}\sum_{i=1}^{n}\theta_{ji}(X_{ji} = x_{ji}\,|N_{g-1})}{N_{g-1}}. \tag{11}$$

where

$$\theta_{ji}(X_{ji} = x_{ji}\,|N_{g-1}) = \begin{cases} 1, & \text{if job } j \text{ processed in the } i\text{th position, } X_{ji} = x_{ji}; \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

Obviously, $pr_g(\mathbf{x})$ is $n \times n$ matrix, which is composed of the probability of each job j processed in the ith position.

4.4 Mutate operator

Mutate operator is inserted in the processing of producing the new population to make population diversity. In the immediately previous population, an individual is picked at random, and then draw two pairs of position from it and exchange them. Its procedure can be described as Fig. 1.

4.5 The new population update

Excepting the initial population, other populations consist of a mutated individual, a large part of individuals sampled according to estimation probability model, and a small part of individuals obtained randomly in the search space.

4.6 Stopping criterion

There are two stopping criterions.

(1) Set an integer number τ, the algorithm will be stopped when the times of obtaining the common solutions is equal to τ.
(2) The algorithm will be stopped if the time of iteration satisfies the requirement.

In this paper, the case (2) of stopping criterion is adopted.

4.7 Solution improved

To improve the solution obtained from the above algorithm, here, the pair-wise interchange technique is used. Namely, a new sequence is derived by interchanging jobs from the first job to the last job. If it is prior to the original one, then replace the original sequence.

Summarizing the above parts, the improved EDA follows the below steps.

4.8 The procedure of the improved EDA

Step 1: Set $g = 0$ and initialize all parameters, go to Step 2;

Step 2: Sample M individuals to construct initial population P_0, go to Step 3;

Step 3: Compute the total tardiness of each individual and obtain the current best solution S^*, go to Step 4;

Step 4: Set $g = g + 1$, select the first N individuals and estimate probability model pr_g, go to Step 5;

Step 5: Draw at random an individual and perform mutating to it, go to Step 6;

Step 6: Sample N $(N < M)$ individuals through pr_{g-1}, and choose randomly $M - N - 1$ solutions from the search space. Simultaneously, update the new population P_g, go to Step 7;

Step 7: Calculate the objective value of each individual and replace S^* with the current best solution, go to Step 8;

Step 8: Check times of iteration, if it satisfies the requirement, namely, $g = G$ (G is the maximum iteration time), go to Step 9. Otherwise, return to Step 4.

Step 9: Calculate the objective value of each individual and obtain a schedule S^* with the best objective value. At this time, a good solution is obtained.

5 Numerical experiments

Since the same problem is not seen in the previous researches, the parts of experimental design follow framework used by Yin et al. (2012). The job normal processing times are generated from a uniform distribution over the integers between 1 and 100. The due dates are generated from a uniform distribution on $\mathrm{Z}(1 - \lambda - R/2), \mathrm{Z}(1 - \lambda + R/2)$ where $\mathrm{Z} = \sum_{j=1}^{n} p_{[j]}$, $\lambda (0 < \lambda \leq 0.5)$ is the factor of the tardiness and $R (0 < R \leq 1)$ is a control variable, which decides the scatter range of the due dates. The size of population $M = 100$. The maximum iteration time $G = 100$. The size of elite population $N = 50$. For designing deteriorating rate b, we follow the criterion used by Lee et al. (2008). All programming are operated on the same personal computer with an Intel (R) Core (TM) 2 processors, where CPU is 2.40 GHz and RAM is 2.93 GB. And the software tools are the Microsoft Visual Studio 2008 and the CPLEX Optimization Studio 12.4. The following experiments are tested according to the above designing rules.

Here, we conveniently explain why NBR is selected in this paper, M-NBR not. For IEDA, MNBR-EDA, where MNBR-EDA denotes that M-NBR is used for generating an initial solution in the processing of applying EDA, NBR and M-NBR are only used to generate an initial solution, the solution of IEDA or MNBR-EDA algorithm towards a stable good near-optimal solution when the iteration time is equal to 100. On the other hand, the CPU time of IEDA is almost equal to that of MNBR-EDA. It is the reason that the time complexity of NBR and that of M-NBR is identical. Therefore, in this paper, we retained the original method—NBR used.

Firstly, test the time and solution performance of algorithms. We set $n = 10, 11$, $\lambda = 0.1, 0.3, 0.5$, $b = 0.1, 0.3, 0.5$, $R = 0.1, 0.5, 0.9$. Thus, there is 54 experimental situations, each of which is randomly generated a set of 100 instances. For all the algorithms, the average error rates (AER) of improved EDA (IEDA) and NBR heuristic (NBR) are recorded in Table 1. The error rate (ER) is represented as $ER = (H - H^*)/H^* \times 100\,\%$, where H is the solution from the IEDA and NBR heuristic and H^* is the optimal solution from CPLEX solving a mixed integer programming model. Meanwhile, since the CPU time of NBR is very

Table 1 Performance of the algorithms

λ	b	R	$n = 10$				$n = 11$			
			AER		CPU Time (s)		AER		CPU Time (s)	
			IEDA	NBR	IEDA	CPLEX	IEDA	NBR	IEDA	CPLEX
0.1	0.1	0.1	0.00	0.00	4.14	68.38	0.00	4.24	5.32	172.52
		0.5	0.00	12.45	4.59	366.25	2.06	8.39	5.65	564.14
		0.9	0.00	4.12	4.54	12.22	4.09	6.16	5.68	1,844.28
	0.3	0.1	0.26	8.28	5.67	89.87	0.00	6.87	5.36	4,024.13
		0.5	0.00	1.40	6.04	62.78	0.00	4.25	4.76	434.63
		0.9	0.00	7.39	4.06	1,007.19	0.00	4.41	4.28	142.98
	0.5	0.1	0.00	5.44	4.43	209.30	0.00	2.67	5.35	10,086.81
		0.5	0.00	3.88	4.06	609.76	0.00	5.55	4.86	4,519.09
		0.9	0.00	6.81	4.00	91.34	4.85	8.85	4.18	83.92
0.3	0.1	0.1	2.33	8.30	3.96	46.72	0.00	7.74	4.53	2,455.61
		0.5	1.01	2.09	4.18	16.03	3.91	11.40	6.71	499.27
		0.9	3.03	6.85	4.79	33.91	5.40	9.78	4.31	26.23
	0.3	0.1	0.00	4.44	4.68	474.41	0.00	3.66	8.85	199.14
		0.5	0.71	6.48	4.75	349.63	0.00	7.30	4.39	71.64
		0.9	0.00	4.11	4.82	630.58	0.00	18.75	4.36	623.30
	0.5	0.1	0.00	6.82	4.79	154.14	0.00	4.42	6.14	430.42
		0.5	0.00	2.37	4.67	1,256.11	0.00	10.47	5.79	1,643.86
		0.9	0.00	0.98	4.50	243.94	0.00	9.97	6.87	6,451.34
0.5	0.1	0.1	0.54	8.94	4.56	47.25	0.95	5.98	6.26	653.38
		0.5	0.00	3.88	3.82	95.30	0.26	7.81	7.04	18,024.72
		0.9	4.16	4.49	4.71	11.69	4.78	6.72	6.46	442.30
	0.3	0.1	0.00	5.03	5.15	249.31	0.00	2.82	9.64	2,682.92
		0.5	0.00	1.04	4.34	280.06	0.00	3.54	8.75	461.89
		0.9	0.00	7.59	4.31	45.63	0.71	10.17	6.95	2,759.47
	0.5	0.1	0.00	14.74	4.48	81.53	0.00	7.17	4.48	19,095.72
		0.5	0.00	8.58	4.31	260.94	0.00	4.29	7.34	2,650.26
		0.9	0.00	6.07	3.90	1,618.86	0.00	14.10	9.50	1, 892.49

short, the mean of CPU time (CPU Time) of IEDA and CPLEX is only noted, where the unit of time is second (s).

From the Table 1, it can be seen that the mean error percentages for the IEDA is very small and acceptable. The solutions obtained from it are all not the optimal ones, but they are better near-optimal ones. For NBR heuristic, most of situations cannot obtain better solutions than ones from IEDA.

About the CPU time, when the number of jobs becomes 11, the CPU time of CPLEX costs lots of times so that generally it is not acceptable. By contrast, that of the IEDA hardly changes and it always keeps a very short time.

Moreover, it shows that the CPU time of the IEDA and NBR is rather short and stable for the certain size of jobs. And it is hardly dependent on λ, R and b.

Table 2 Performance of the algorithms for the large scale of problem

λ	b	R	The average error rate				
			$n = 10$	$n = 30$	$n = 50$	$n = 70$	$n = 90$
0.1	0.1	0.1	0.00	6.96	26.71	21.07	13.85
		0.5	12.45	16.2	20.73	21.63	27.36
		0.9	4.12	21.78	20.52	20.24	23.52
	0.3	0.1	8.02	4.9	31.95	23.6	29.69
		0.5	1.4	18.55	36.46	49.53	24.68
		0.9	7.39	8.06	27.76	19.92	21.23
	0.5	0.1	5.44	22.57	34.21	69.62	27.96
		0.5	3.88	17.82	30.35	8.79	35.35
		0.9	6.81	6.75	12.45	24.92	35.15
0.3	0.1	0.1	5.97	3.36	17.56	19.41	22.52
		0.5	1.08	10.94	19.7	27.27	29.08
		0.9	3.82	37.06	31.6	24.64	21.53
	0.3	0.1	4.44	15.89	14.49	30.56	35.97
		0.5	5.77	16.48	9.08	24.18	29.87
		0.9	4.11	34.18	31.7	32.77	26.6
	0.5	0.1	6.82	16.81	26.54	22.33	33.69
		0.5	2.37	43.47	10.87	7.67	40.36
		0.9	0.98	22.02	29.17	25.85	18.17
0.5	0.1	0.1	8.4	13.71	9.46	12.96	28.15
		0.5	3.88	7.05	17.55	20.5	22.9
		0.9	0.33	4.45	27.48	10.82	18.63
	0.3	0.1	5.03	14.38	24.77	24.49	27.36
		0.5	1.04	15.68	19.78	19.82	25.72
		0.9	7.59	17.51	15.22	24.48	28.08
	0.5	0.1	14.74	11.03	15.15	28.85	22.68
		0.5	8.58	2.76	56.82	18.2	26.27
		0.9	6.07	11.61	25.39	64.63	24.13
Average			5.20	15.63	23.83	25.87	26.69

Secondly, since the same problem is not seen in the previous researches and the CPLEX needs a long time for the large scale of jobs, here, and we only further test the performance of IEDA and NBR algorithms for the large scale of jobs. Set $n = 10, 30, 50, 70, 90$, the other parameters are the same as the first experiment. As a result, there are 108 experiments, each of which is randomly generated a set of 100 instances. For all the algorithms, Table 2 provides the average error rate which is the solution of IEDA relative to the one of NBR, i.e., the error rate is represented as $\text{var} = (H_{\text{NBR}} - H_{\text{IEDA}})/H_{\text{NBR}} \times 100\,\%$, where H_{IEDA} is a solution from IEDA and H_{NBR} from NBR. If var is positive, it denotes that the solution from IEDA is prior to the one from NBR, the larger it is, the better the solution from IEDA is, and vice versa.

From Table 2, it can be seen that the solutions from IEDA are always far prior to ones from NBR for the large scale, in which the average difference value percent will gradually increase with the size of jobs.

As a result, considering the above performances, the improved EDA is a good choice for decision maker.

6 Conclusion

This paper focuses on the NP hard problem of single-machine scheduling with deteriorating job and different due dates, in which the objective is to find an optimal schedule for minimizing tardiness. According to the characteristics of the problem, two special cases of the problem are given. Then, a mixed integer programming model and IEDA are proposed. Numerical experiments suggest that the proposed algorithm is effective and efficient. In the future, a single-machine scheduling problem with different release times and due dates will be considered.

Acknowledgments This work is supported by the National Natural Science Foundation of China under Grant Nos. 71071028, 71021061 and 70931001, the National Science Foundation for Distinguished Young Scholars of China under Grant No. 61225012; the Specialized Research Fund of the Doctoral Program of Higher Education for the Priority Development Areas under Grant No. 20120042130003; Specialized Research Fund for the Doctoral Program of Higher Education under Grant No. 20110042110024; the Specialized Development Fund for the Internet of Things from the ministry of industry and information technology of the P.R. China; the Fundamental Research Funds for the Central Universities under Grant Nos. N110204003 and N120104001.

Appendix

The steps of the NBR heuristic proposed by Holsenback and Russell (1992) are as follows.

Step 1: Label all jobs and adjust due dates according to due date modification rule. Then, order the set by EDD rule. Renumber the jobs so that this sequence is (J_1,..., J_N). And calculate the slack or tardiness of each job in the sequence.

Step 2: Beginning with the job J_N in the last position, and incrementally proceeding toward J_1, identify the first job J_k with $T_k > P_k$. If none exists, proceed to Step 7.

Step 3: Beginning with J_{k-1} and continuing toward J_1, identify the first preceding job J_j with the property $p_j > p_k$. If none exists, proceed to Step 6. Beginning with J_{j-1} and continuing toward J_1, identify the first preceding job J_{j-m} with property $p_{j-m} > p_j$. Continue in this manner until J_1 has been considered.

Step 4: Compute the $\mathrm{NBR}_j = \sum_{i=j+1}^{k} \min(p_j, T_i) - \max(0, \sum_{i=j+1}^{k} p_i - S_j)$ for each job J_i, where $S_j = \max(d_j - C_j)$, identified in Step 3, considering that it will be relocated to a position immediately following J_k. Relocate that job J_i, yielding the greatest NBR_i, subject to the constraint $\mathrm{NBR}_i > 0$. If no $\mathrm{NBR}_i > 0$, proceed to Step 6. If two or more jobs yield the same NBR_i, relocate that job with the greatest processing time. This selection method is purely arbitrary and is chosen to reduce the number of computations necessary for a final ordering.

Step 5: Renumber the current sequences as (J_1,..., J_N) and recompute the corresponding tardiness and slack.

Step 6: Begin with the job in position $k - 1$ and repeat Steps 2–4, substituting $k - 1$ for k. Continue in this fashion until the job in position 1 has been considered.

Step 7: Final ordering is complete. Apply the original due dates if any were modified and compute the tardiness of the schedule.

References

Aickehin U, Burke EK, Li J (2007) An estimation of distribution algorithm with intelligent local search for rule-based nurse rostering. J Oper Res Soc 58:1574–1585

Aickehin U, Li J (2007) An estimation of distribution algorithm for nurse scheduling. Ann Oper Res 155:289–309

Akker JM, Diepen G, Hoogeveen JA (2010) Minimizing total weighted tardiness on a single machine with release dates and equal-length jobs. J Sched 13:561–576

Akturk MS, Ilhan T (2011) Single CNC machine scheduling with controllable processing times to minimize total weighted tardiness. Comput Oper Res 38:771–781

Bachman A, Cheng TCE, Janiak A, Ng CT (2002) Scheduling start time dependent jobs to minimize the total weighted completion time. J Oper Res Soc 53:688–693

Ben-Daya M, Al-Fawzan M (1996) A simulated annealing approach for the one-machine mean tardiness scheduling problem. Eur J Oper Res 93:61–67

Bozejko W (2010) Parallel path relinking method for the single machine total weighted tardiness problem with sequence-dependent setups. J Intell Manuf 21:777–785

Browne S, Yechiali U (1990) Scheduling deteriorating jobs on a single processor. Oper Res 38:495–498

Cheng TCE, Ding Q (1998) The complexity of single machine scheduling with release times. Inf Process Lett 65:75–79

Cheng TCE, Ding Q (2001) Single machine scheduling with step-deteriorating processing times. Eur J Oper Res 134:623–630

Cheng TCE, Lee WC, Wu CC (2010) Single-machine scheduling with deteriorating functions for job processing times. Appl Math Model 34:4171–4178

Du J, Leung JYT (1990) Minimizing total tardiness on one machine is NP-hard. Math Oper Res 15:483–495

Farahani MH, Hosseini L (2013) Minimizing cycle time in single machine scheduling with start time-dependent processing times. Int J Adv Manuf Technol 64:1479–1486

Graham RL, Lawler EL, Lenstra JK (1979) Optimization and approximation in the deterministic sequencing and scheduling: a survey. Ann Discret Math 5:287–326

Gupta JND, Gupta SK (1988) Single facility scheduling with integer processing times. Comput Ind Eng 14:387–393

Holsenback JE, Russell RM (1992) A heuristic algorithm for sequencing on one machine to minimize total tardiness. J Oper Res Soc 43:53–62

Jarboui B, Eddaly M, Siarry P (2009) An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problem. Comput Oper Res 36:2638–2646

Kirlik G, Oguz C (2012) A variable neighborhood search for minimizing total weighted tardiness with sequence dependent setup times on a single machine. Comput Oper Res 39:1506–1520

Lai PJ, Lee WC (2010) Single-machine scheduling with a integer deterioration function. Inf Process Lett 110:455–459

Lai PJ, Lee WC, Chen HH (2011) Scheduling with deteriorating jobs and past-sequence-dependent setup times. Int J Adv Manuf Technol 54:737–741

Lee WC, Wu CC, Chung YH (2008) Scheduling deteriorating jobs on a single machine with release times. Comput Ind Eng 54:441–452

Mazdeh MM, Nakhjavani AK, Zareei A (2010) Minimizing total weighted tardiness with drop dead dates in single machine scheduling problem. Int J Ind Eng Prod Res 21:89–95

Meng J, Yu J, Lu X (2012) Scheduling deteriorating jobs with a common due window on a single machine. Inf Technol J 11:392–395

Mühlenbein H, Paaß G (1996) From recombination of genes to the estimation of distributions I. Binary parameters. In: Lecture notes in computer science: parallel problem solving from nature-PPSN IV, vol 1411. Springer, Berlin, pp 178–187

Ozturkoglu Y, Bulfin RL (2011) A unique integer mathematical model for scheduling deteriorating jobs with rate-modifying activities on a single machine. Int J Adv Manuf Technol 57:753–762

Pan Q, Ruiz R (2012) An estimation of distribution algorithm for lot-streaming flow shop problems with setup times. Omega 40:166–180

Russell RM, Holsenback JE (1997) Evaluation of greedy, myopic and less-greedy heuristics for the single machine, total tardiness problem. J Oper Res Soc 48:640–646

Shah R, Reed P (2011) Comparative analysis of multiobjective evolutionary algorithms for random and correlated instances of multiobjective d-dimensional knapsack problems. Eur J Oper Res 211:466–479

Toksan DM, Guner E (2009) Parallel machine earliness/tardiness scheduling problem under the effects of position based learning and linear/integer deterioration. Comput Oper Res 36:2394–2417

Wang L, Fang C (2012) An effective estimation of distribution algorithm for the multi-mode resource-constrained project scheduling problem. Comput Oper Res 39:449–460

Yin Y, Cheng TCE, Xu D, Wu CC (2012) Common due date assignment and scheduling with a rate-modifying activity to minimize the due date, earliness, tardiness, holding, and batch delivery cost. Comput Ind Eng 63:223–234

Yin Y, Xu D (2011) Some single-machine scheduling problems with general effects of learning and deterioration. Comput Math Appl 61:100–108