Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 768148, 12 pages
http://dx.doi.org/10.1155/2015/768148

Research Article

Sum-of-Processing-Times-Based Two-Agent Single-Machine Scheduling with Aging Effects and Tardiness

Do Gyun Kim and Jin Young Choi

Department of Industrial Engineering, Ajou University, Suwon 443-749, Republic of Korea

Correspondence should be addressed to Jin Young Choi; choijy@ajou.ac.kr

Received 21 January 2015; Accepted 18 March 2015

Academic Editor: Yunqiang Yin

We consider a two-agent single-machine scheduling problem that minimizes the total weighted tardiness of one agent under the restriction that the second agent is prohibited from having tardy jobs. The actual processing times of all jobs are affected by a sum-of-processing-times-based aging effect. After showing the NP-hardness of the problem, we design a branch-and-bound (B&B) algorithm to find an optimal solution by developing dominance properties and a lower bound for the total weighted tardiness to increase search efficiency. Because B&B takes a long time to find an optimal solution, we propose a genetic algorithm as an efficient, near optimal solution approach. Four methods for generating initial populations are considered, and edge recombination crossover is adopted as a genetic operator. Through numerical experiments, we verify the outstanding performance of the proposed genetic algorithm.

1. Introduction

Over the past few decades, much research has focused on scheduling problems that consider diverse conditions resulting from production method and working environment changes. In particular, in recent years, a multiagent scheduling problem has received more attention. The multiagent scheduling problem determines the processing order of jobs that belong to different agents with different objectives. A proper scheduling method effectively utilizes limited resources. In this paper, we consider a special case of the multiagent scheduling problem—a two-agent single-machine scheduling problem, where each agent or decision maker competes to use a single machine while optimizing their own objective.

Multiagent scheduling is applicable to a variety of industries. For example, runway scheduling for airplanes is competitive among airlines that want their flights to occupy the limited runway space first [1]. In this situation, each flight is a job to be processed, each airline is an agent, and runways are limited resources. A production line that processes a combination of two product families or the task of creating a schedule for production and maintenance departments that commonly use the same project resources are two examples of two-agent problems [2]. In these cases, the two product families or two departments play the role of agents, and the tasks belonging to each product family or project schedule fulfilled by each department are the jobs to be processed by the agents.

To solve the two-agent scheduling problem, we can consider a traditional, bicriteria optimization approach. Specifically, because two agents have two different objective functions, (i) an optimal schedule for one objective function can be found and then subsequently used as a new constraint to optimize the second objective function, or (ii) a new objective function can be constructed as a weighted linear combination of the two objective functions. However, the first method is not valid because all jobs affect both objective functions at the same time. The second method is also not reasonable if the units of measure of each objective function are different. Therefore, a new efficient and exceptional scheduling framework that reflects the multiagent concept is required.

The concept of multiagent scheduling was suggested by Cres and Moulin [3] and Baker and Smith [4]. Agnetis et al. [5] introduced new terminology, namely, multiagent scheduling (MAS). Agnetis et al. [5] studied two-agent scheduling problems with a single machine or two-machine open shop environment and proved their complexity. Later, Agnetis et al.

[6] and Cheng et al. [7] proposed scheduling problems that contain more than two agents. Leung et al. [8] and Ng et al. [9] analyzed problems with unknown computational complexity or additional constraints such as precedence. Balasubramanian et al. [10] and Tuong and Soukhal [11] applied the concept of MAS to various machine environments such as parallel machines and three-machine flow shop. Lee et al. [12] constructed a solution framework for the two-agent scheduling problem with NP-hard complexity by applying a branch-and-bound (B&B) algorithm and simulated annealing. Yin et al. [13] proposed polynomial or pseudopolynomial solutions for two NP-hard problems to decide the due dates of one agent while not exceeding the maximum regular function value or the number of tardy jobs of the other agent.

Workers in real working environments gain job experience by doing the same tasks repeatedly, which may result in jobs being completed faster than the normal processing time. However, the actual execution time of completing a job may deteriorate if repeated use of machinery causes mechanical errors to occur or if a delayed start time causes the expected processing time of a job to increase. The decrease in the processing time in the former case is called the learning effect, and the increase in the processing time in the latter case is called the aging effect. These effects can be represented using mathematical expressions, where the actual processing times can be represented using position-based linear or exponential functions, or the sum-of-processing-times-based functions.

A lot of research has been devoted to the learning and aging effects in scheduling problems. Linear learning and aging effects were introduced into the scheduling problem by Cheng and Wang [14], J. N. D. Gupta and S. K. Gupta [15], and Browne and Yechiali [16]. Cheng and Wang [14] considered a scheduling problem where processing times were affected by a simple linear learning effect, which revealed the complexity of the problem. J. N. D. Gupta and S. K. Gupta [15] and Browne and Yechiali [16] studied scheduling problems with linear and polynomial aging effects and proposed a heuristic algorithm to find a near optimal solution. Bachman and Janiak [17] and Wang and Xia [18], motivated by Cheng and Wang [14], proposed more simple and generalized learning functions than those of Cheng and Wang [14]. Moreover, Mosheiov [19], Sundararaghavan and Kunnathur [20], and Bachman et al. [21] studied the complexity of scheduling problems under linear aging effects and suggested solution algorithms for polynomial solvable cases. Alidaee and Womer [22] carried out a review of various scheduling problems with linear aging effects noting their complexity and aging function form. Yin et al. [23] suggested the single-machine scheduling problem with linear aging effect under two-agent consideration and provided the complexity and solvability information for various combinations of the two agents' objective functions.

Scheduling problems with exponential learning and aging effects were considered by Biskup [24]. Biskup [24] investigated their complexity and proposed a solution algorithm to scheduling problems with exponential aging effects. Later, Mosheiov [25, 26] and Mosheiov and Sidney [27] enriched the contribution of Biskup [24] by considering several machine environments and objective functions. Lee and Wu [28] explored a two-machine flow shop scheduling problem where each machine had its own learning effect. Cheng et al. [29] presented a concise survey of scheduling problems affected by linear, piecewise linear, and nonlinear aging effects.

It was noted, however, that learning and aging effects represented by linear or exponential functions cannot reflect the amount of jobs already processed, which adversely causes large variations in actual processing time values [30]. To overcome this limitation, a new learning and aging function was introduced based on the sum-of-processing-times of single agent. Yin et al. [31] studied the single-machine and flowshop scheduling problems with sum-of-processing-times-based as well as position-dependent learning effect. Yin et al. [32] developed some approximation algorithms for single-machine scheduling problem considering sum-of-processing-times-based aging effect. In case of two agents, Liu et al. [33] proposed a polynomial time algorithm under a sum-of-processing-times-based aging effect and considered the total completion time and makespan as the objective functions of the two agents. Wu et al. [34] and Wu [35] investigated a scheduling problem where one agent had a learning effect and the second agent had an aging effect. The total weighted completion time and maximum lateness were the objective functions of the two agents, and they solved the problem using ant colony optimization and simulated annealing, respectively.

Note that of the performance measures considered in these previous works, tardiness is not considered for two-agent single-machine scheduling based on the sum-of-processing-times, which is closely related to the due date. Specifically, when a due date is not met, additional penalties are accrued by tardiness; therefore, the due date is a crucial index for a firm to maintain a solid relationship with a client. As a special case of the tardiness restriction, we solve a scheduling problem where the total weighted tardiness has a very high computational complexity as shown in Figure 1 [36]. Here, w_j, C_j, T_j, and U_j are the weight, completion time, tardiness, and tardiness indicator of job j, respectively. $C_{\max}$ and $L_{\max}$ are the makespan and maximum lateness, respectively. Furthermore, the direction of arrows represents the difficulty of the objective functions; that is, the objective function in the arrow head has higher complexity than that in the arrow end.

Motivated by these remarks, we consider a two-agent single-machine scheduling problem that minimizes the total weighted tardiness for the first agent under the restriction that no tardy job is allowed for the second agent. Moreover, the actual processing times of all jobs are affected by the sum-of-processing-times-based aging effect. We prove the NP-hardness of the problem and design a B&B algorithm to find an optimal solution by developing dominance properties and a lower bound for the total weighted tardiness that makes the search procedure efficient. Because B&B takes a long time to find an optimal solution, we propose a genetic algorithm as an efficient, near optimal solution approach. Four methods to generate initial populations are considered, and edge recombination crossover is adopted as a genetic operator, which was suggested by Whitley et al. [37] and specialized

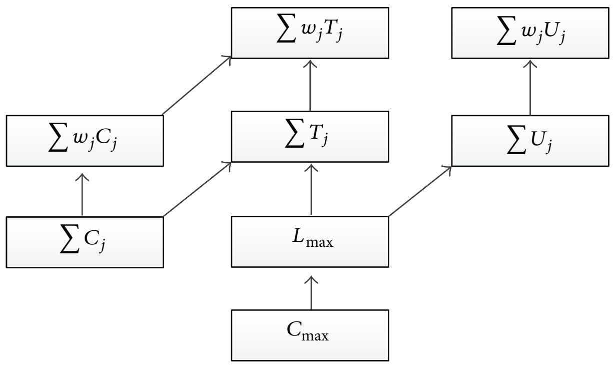

FIGURE 1: Complexity hierarchies of objective functions in scheduling problems.

for combinatorial optimization problems. A numerical experiment is designed and executed to evaluate the performance of the proposed B&B and genetic algorithm.

The remainder of this paper is organized as follows. Section 2 defines the problem and suggests a B&B algorithm, including dominance properties and a lower bound. Section 3 develops an efficient genetic algorithm to obtain near optimal solutions, and Section 4 evaluates the performance of the suggested algorithms using a numerical experiment. Finally, in Section 5, we summarize our results and discuss future work.

2. Problem Definition and a Branch-and-Bound Algorithm

2.1. Problem Definition. First, we define the notation used throughout this paper and formally define the problem. The notation is as follows:

n: total number of jobs to be processed;

n_X: the number of jobs to be processed by agent X, where $X \in \{A, B\}$, $n_A + n_B = n$;

J_i^X: the ith job of agent X, where $X \in \{A, B\}$;

$J^A = \{J_1^A, J_2^A, \ldots, J_{n_A}^A\}$: set of jobs to be processed by agent A;

$J^B = \{J_1^B, J_2^B, \ldots, J_{n_B}^B\}$: set of jobs to be processed by agent B;

p_i^X: normal processing time of job J_i^X, where $X \in \{A, B\}$;

w_i^X: weight of job J_i^X to be processed by agent $X \in \{A, B\}$;

$p_{[l]}^X$: normal processing time of any job assigned to position l in a sequence, where $X \in \{A, B\}$;

p_{ir}^X: actual processing time of job J_i^X assigned to position r in a sequence, where $X \in \{A, B\}$;

α: aging index $(0 < \alpha < 1)$;

M: large constant positive value.

The scheduling problem under consideration is a single-machine scheduling problem with two agents, A and B. They have jobs affected by a sum-of-processing-times-based aging effect. The objective function to minimize is the total weighted tardiness of agent A under the restriction that no tardy jobs are allowed for agent B. In particular, we focus on the actual processing time p_{ir}^X of job J_i^X assigned to rth position in a sequence; that is,

$$p_{ir}^X = p_i^X \left(1 + \sum_{l=1}^{r-1} p_{[l]}^X\right)^{\alpha}. \quad (1)$$

We formulate a mixed integer programming (MIP) model for the problem as follows. First, we arrange the jobs of the two agents:

$$J_i = \begin{cases} J_i^A, & \text{if } 1 \le i \le n_A, \\ J_{i-n_A}^B, & \text{if } n_A + 1 \le i \le n_A + n_B. \end{cases} \quad (2)$$

By defining a decision variable x_{ij}, $\forall i < j$, $i = 1, 2, \ldots, n-1$ as

$$x_{ij} = \begin{cases} 1, & \text{if } J_i \text{ precedes } J_j, \\ 0, & \text{otherwise,} \end{cases} \quad (3)$$

we can construct the following MIP model:

$$\text{Min} \quad Z = \sum_{i=1}^{n_A} w_i^A T_i \quad (4)$$

$$\text{subject to} \quad S_i + p_i \cdot \left[1 + \sum_{j=1}^{i-1} \left(p_j x_{ji}\right) + \sum_{j=i+1}^{n} p_j \left(1 - x_{ij}\right)\right]^{\alpha} = C_i, \quad \forall i = 1, 2, \ldots, n \quad (5)$$

$$C_i - d_i \le T_i \quad \forall i = 1, 2, \ldots, n_A \quad (6)$$

$$C_i - d_i \le 0, \quad \forall i = n_A + 1, n_A + 2, \ldots, n_A + n_B \quad (7)$$

$$C_i \le S_j + M\left(1 - x_{ij}\right), \quad \forall 1 \le i < j \le n \quad (8)$$

$$C_j \le S_i + M x_{ij}, \quad \forall 1 \le i < j \le n \quad (9)$$

$$C_i \ge 0, \quad S_i \ge 0, \quad T_i \ge 0, \quad \forall i = 1, 2, \ldots, n \quad (10)$$

$$x_{ij} \in \{0, 1\}, \quad \forall i < j, \ i = 1, 2, \ldots, n-1, \quad (11)$$

where C_i, d_i, p_i, S_i, and T_i are the completion time, due date, normal processing time, starting time, and tardiness of job J_i, respectively. Equation (4) is the objective function for minimizing the total weighted tardiness of jobs for agent A. Equation (5) represents the completion time of job J_i, which considers the sum-of-processing-times-based aging effect.

Specifically, the sum-of-processing-times of jobs already processed are denoted by $\sum_{j=1}^{i-1}(p_j x_{ji})$ and $\sum_{j=i+1}^{n} p_j(1 - x_{ij})$ when $j < i$ and $j > i$, respectively. Equation (6) computes the total tardiness of job J_i for agent A, and (7) prevents jobs for agent B from being tardy. Equations (8) and (9) ensure that the completion times of preceding jobs are less than or equal to the starting times of subsequent jobs. Equation (10) is the nonnegativity constraint, and (11) is the binary constraint of x_{ij}.

To facilitate our discussion, we define the following notations.

d_i^X: Due date of job J_i^X to be processed by agent $X \in \{A, B\}$.

T_i^A: Tardiness of job J_i^A to be processed by agent A, where $T_i^A = \max\{C_i^A - d_i^A, 0\}$.

C_i^X: Completion time of job J_i^X to be processed by agent $X \in \{A, B\}$.

U_j^B: Tardiness index of job J_j^B to be processed by agent B, where

$$U_j^B = \begin{cases} 1, & \text{if } T_j^B > 0 \\ 0, & \text{if } T_j^B = 0. \end{cases} \tag{12}$$

Graham et al. [38] proposed a classification scheme consisting of three fields, $\Psi_1 \mid \Psi_2 \mid \Psi_3$, for various scheduling problems. In this notation, Ψ_1 is the machine environment such as single-machine, flow shop, or job shop, Ψ_2 is the detailed information about the characteristics of jobs (e.g., precedence, preemption, and learning/aging effect), and the last field Ψ_3 represents the objective function to be optimized. Furthermore, Agnetis et al. [5] extended this notation into a new scheme, $\Psi_1 \mid \Psi_2 \mid \Psi_3^A : \Psi_3^B$, to denote two objective functions for two-agent scheduling problems. Based on this notation, the scheduling problem considered in this paper can be expressed as

$$1 \mid p_{ir}^X = p_i^X \left(1 + \sum_{l=1}^{r-1} p_{[l]}^X\right)^\alpha \mid \sum_{i=1}^{n_A} w_i^A T_i^A : \sum_{j=1}^{n_B} U_j^B \le 0. \tag{13}$$

We now discuss the complexity of this problem. Leung et al. [8] proved the NP-hardness of the problem $1 \parallel \sum_{i=1}^{n_A} C_i^A : \sum_{j=1}^{n_B} U_j^B \le 0$, which is a special case of the problem $1 \parallel \sum_{i=1}^{n_A} T_i^A : \sum_{j=1}^{n_B} U_j^B \le 0$ subject to the condition that all jobs of agent A have the same due date, namely, 0. Moreover, $1 \parallel \sum_{i=1}^{n_A} T_i^A : \sum_{j=1}^{n_B} U_j^B \le 0$ is a special case of $1 \parallel \sum_{i=1}^{n_A} w_i^A T_i^A : \sum_{j=1}^{n_B} U_j^B \le 0$ subject to the condition that all jobs of agent A have the same weight equal to 1. Thus, from the NP-hard complexity of $1 \parallel \sum_{i=1}^{n_A} C_i^A : \sum_{j=1}^{n_B} U_j^B \le 0$, we can infer the NP-hardness of $1 \parallel \sum_{i=1}^{n_A} T_i^A : \sum_{j=1}^{n_B} U_j^B \le 0$ and $1 \parallel \sum_{i=1}^{n_A} w_i^A T_i^A : \sum_{j=1}^{n_B} U_j^B \le 0$, indicating that the problem expressed in (13) also has NP-hard complexity.

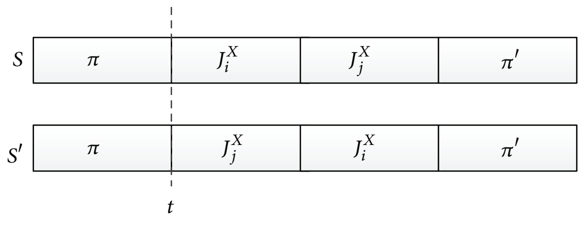

Figure 2: Two schedules S and S' that include two adjacent jobs J_i^X and J_j^X.

2.2. A Branch-and-Bound Algorithm. Because the complexity of the scheduling problem under consideration is NP-hard, we first develop a B&B algorithm to find an optimal solution. This B&B algorithm has the advantage of being able to obtain the global optimal solution despite high complexity and despite its poor computational efficiency resulting from a full enumeration-based search process. Therefore, additional criteria such as dominance properties and a lower bound must be established to efficiently obtain an optimal solution. Baker and Trietsch [39] defined dominance properties for combinatorial optimization problems as standards for excluding a redundant subset suspected of not containing the optimal solution; dominance properties can distinguish only a valuable subset containing the optimal solution from all subsets in the solution space. Jouglet and Carlier [40] carried out a literature review of a variety of dominance properties, categorized by characteristics and organized them according to their utility in combinatorial optimization problems.

Motivated by these observations, in this section, we propose two kinds of dominance properties based on optimality and feasibility conditions to increase the speed of the search process of the B&B algorithm. By comparing two schedules, the optimality-based dominance property can provide a standard to identify a superior schedule whose total weighted tardiness is smaller than that of the other schedule. The feasibility-based dominance property can be used to discard infeasible schedules that contain tardy jobs for agent B. As a result, through these optimality- and feasibility-based dominance properties, an optimal solution can be acquired without generating all possible branches and nodes.

In this paper, dominance properties are derived using a pairwise interchange method applied to two adjacent jobs. We assume that schedule S includes two jobs, J_i^X and J_j^X, that are scheduled consecutively in the rth and $(r+1)$th positions as shown in Figure 2. Furthermore, we consider another schedule S' generated by interchanging jobs J_i^X and J_j^X in schedule S. Partial sequences π and π' are the scheduled and unscheduled jobs, respectively. The starting times of J_i^X in schedule S and the starting times of J_j^X in schedule S' are set to be equal to t.

To derive the dominance conditions between schedules S and S', we can compute the following:

$$C_i(S) = t + p_i^X \left(1 + \sum_{l=1}^{r-1} p_{[l]}^X\right)^\alpha,$$

$$C_j(S) = t + p_i^X\left(1 + \sum_{l=1}^{r-1} p_{[l]}^X\right)^\alpha + p_j^X\left(1 + \sum_{l=1}^{r-1} p_{[l]}^X + p_i^X\right)^\alpha,$$
$$C_j(S') = t + p_j^X\left(1 + \sum_{l=1}^{r-1} p_{[l]}^X\right)^\alpha,$$
$$C_i(S') = t + p_j^X\left(1 + \sum_{l=1}^{r-1} p_{[l]}^X\right)^\alpha + p_i^X\left(1 + \sum_{l=1}^{r-1} p_{[l]}^X + p_j^X\right)^\alpha. \quad (14)$$

Then, we establish the following three conditions that are necessary to support the dominance of S over S':

$$\text{(a)}\ C_j(S) < C_i(S') \quad (15)$$
$$\text{(b)}\ \sum_{x\in\{i,j\}:J_x^A\in J^A} w_x^A T_x^A(S) \le \sum_{x\in\{i,j\}:J_x^A\in J^A} w_x^A T_x^A(S') \quad (16)$$
$$\text{(c)}\ C_x^B(S) \le d_x^B, \quad \text{for } x \in \{i,j\} : J_x^B \in J^B. \quad (17)$$

Condition (a) guarantees that the sum of weighted tardiness of jobs in π' for schedule S is smaller than that of schedule S'. Condition (b) represents the dominance of schedule S over schedule S' with respect to the sum of weighted tardiness of jobs J_i^X and J_j^X in J^A. Condition (c) is the feasibility of the due dates for jobs J_i^X and J_j^X in schedule S for agent B.

Rudek [41] showed that the problem $1 \mid p_{ir} = p_i[1 + \sum_{l=1}^{r-1} f(p_{[l]})]^\alpha \mid C_{\max}$ is optimally solvable by scheduling jobs in nonincreasing order of p_i (i.e., longest processing time (LPT) rule) if $0 < \alpha < 1$ and $d^2 f/dp_{[l]}{}^2 \le 0$. For the scheduling problem under consideration, $f(p_{[l]}) = p_{[l]}$ and $d^2 f/dp_{[l]}{}^2 = 0$, which means that $C_{\max}$ can be minimized by arranging jobs according to the LPT rule; that is, (15) holds if the processing times of jobs J_i^X and J_j^X satisfy $p_i^X > p_j^X$. Based on these arguments, the following dominance properties, based on optimality and feasibility, hold.

Property 1. For schedule S with two tardy jobs J_i^A and J_j^A scheduled in the rth and $(r+1)$th positions, S dominates S' if

(i) $p_i^A > p_j^A$,

(ii) $w_i^A p_i^A/((1 + \sum_{l=1}^{r-1} p_{[l]}^X + p_i^X)^\alpha - (1 + \sum_{l=1}^{r-1} p_{[l]}^X)^\alpha) > w_j^A p_j^A/((1 + \sum_{l=1}^{r-1} p_{[l]}^X + p_j^X)^\alpha - (1 + \sum_{l=1}^{r-1} p_{[l]}^X)^\alpha)$,

(iii) $w_i^A/p_i^A > w_j^A/p_j^A$.

Proof. Taken together, (i) of Property 1 and the results of Rudek [41] imply $C_j(S) < C_i(S')$. Moreover,

$$\begin{aligned} &w_j^A T_j^A(S') + w_i^A T_i^A(S') - w_i^A T_i^A(S) - w_j^A T_j^A(S) \\ &\quad = w_i^A p_j^A\left(1 + \sum_{l=1}^{r-1} p_{[l]}^X\right)^\alpha - w_j^A p_i^A\left(1 + \sum_{l=1}^{r-1} p_{[l]}^X\right)^\alpha \\ &\qquad + w_i^A p_i^A\left[\left(1 + \sum_{l=1}^{r-1} p_{[l]}^X + p_j^A\right)^\alpha - \left(1 + \sum_{l=1}^{r-1} p_{[l]}^X\right)^\alpha\right] \\ &\qquad - w_j^A p_j^A\left[\left(1 + \sum_{l=1}^{r-1} p_{[l]}^X + p_i^A\right)^\alpha - \left(1 + \sum_{l=1}^{r-1} p_{[l]}^X\right)^\alpha\right] \ge 0 \end{aligned} \quad (18)$$

because

$$\begin{aligned} &\frac{w_i^A p_i^A}{\left(1 + \sum_{l=1}^{r-1} p_{[l]}^X + p_i^X\right)^\alpha - \left(1 + \sum_{l=1}^{r-1} p_{[l]}^X\right)^\alpha} \\ &\quad > \frac{w_j^A p_j^A}{\left(1 + \sum_{l=1}^{r-1} p_{[l]}^X + p_j^X\right)^\alpha - \left(1 + \sum_{l=1}^{r-1} p_{[l]}^X\right)^\alpha}, \\ &\quad \frac{w_i^A}{p_i^A} > \frac{w_j^A}{p_j^A}. \end{aligned} \quad (19)$$

We do not consider (17) because both J_i^A and J_j^A are jobs of agent A. Therefore, S dominates S'. □

Property 2. For schedule S with two jobs J_i^A and J_j^B scheduled in the rth and $(r+1)$th positions, S dominates S' if

(i) $p_i^A > p_j^B$,

(ii) $t + p_i^A(1 + \sum_{l=1}^{r-1} p_{[l]}^X)^\alpha + p_j^B(1 + \sum_{l=1}^{r-1} p_{[l]}^X + p_i^A)^\alpha < d_j^B$.

Proof. It follows from (i) of Property 2 that $C_j(S) < C_i(S')$. Moreover,

$$\begin{aligned} &w_i^A T_i^A(S') - w_i^A T_i^A(S) \\ &\quad = w_i^A \max\left[t + p_j^B\left(1 + \sum_{l=1}^{r-1} p_{[l]}^X\right)^\alpha \right. \\ &\qquad \left. + p_i^A\left(1 + \sum_{l=1}^{r-1} p_{[l]}^X + p_j^B\right)^\alpha - d_i^A, 0\right] \\ &\qquad - w_i^A \max\left[t + p_i^A\left(1 + \sum_{l=1}^{r-1} p_{[l]}^X\right)^\alpha - d_i^A, 0\right] \ge 0. \end{aligned} \quad (20)$$

From (ii) of Property 2, $C_j(S) - d_j^B = t + p_i^A(1 + \sum_{l=1}^{r-1} p_{[l]}^X)^\alpha + p_j^B(1+\sum_{l=1}^{r-1} p_{[l]}^X + p_i^A)^\alpha - d_j^B \le 0$. Therefore, S dominates S'. □

Property 3. For schedule S with two jobs J_j^B and J_i^A scheduled in the rth and $(r+1)$th positions, S dominates S' if

(i) $p_i^B > p_j^A$,

(ii) $t + p_i^B(1 + \sum_{l=1}^{r-1} p_{[l]}^X)^\alpha + p_j^A(1 + \sum_{l=1}^{r-1} p_{[l]}^X + p_i^A)^\alpha < d_j^A$,

(iii) $t + p_i^B(1 + \sum_{l=1}^{r-1} p_{[l]}^X)^\alpha < d_i^B$.

Proof. From (i) of Property 3, $C_j(S) < C_i(S')$. Furthermore,

$$\begin{aligned} w_j^A T_j^A(S') &- w_j^A T_j^A(S) \\ &= w_j^A \max\left[t + p_j^A\left(1+\sum_{l=1}^{r-1} p_{[l]}^X\right)^\alpha - d_j^A, 0\right] \\ &\quad - w_j^A \max\left[t + p_i^B\left(1+\sum_{l=1}^{r-1} p_{[l]}^X\right)^\alpha \right. \\ &\qquad \left. + p_j^A\left(1+\sum_{l=1}^{r-1} p_{[l]}^X + p_i^B\right)^\alpha - d_j^A, 0\right] = 0, \end{aligned} \quad (21)$$

which is a consequence of (ii) of Property 3. In addition, $C_i(S) - d_i^B = t + p_i^B(1 + \sum_{l=1}^{r-1} p_{[l]}^X)^\alpha - d_i^B \le 0$ since $t + p_i^B(1 + \sum_{l=1}^{r-1} p_{[l]}^X)^\alpha < d_i^B$. Therefore, S dominates S'. □

Property 4. For schedule S with two jobs J_i^B and J_j^B scheduled in the rth and $(r+1)$th positions, S dominates S' if

(i) $p_i^B > p_j^B$,

(ii) $t + p_i^B(1 + \sum_{l=1}^{r-1} p_{[l]}^X)^\alpha < d_i^B$,

(iii) $t + p_i^B(1 + \sum_{l=1}^{r-1} p_{[l]}^X)^\alpha + p_j^B(1 + \sum_{l=1}^{r-1} p_{[l]}^X + p_i^B)^\alpha < d_j^B$.

Proof. From (i) of Property 4, $C_j(S) < C_i(S')$. Moreover, $C_i(S) - d_i^B = t + p_i^B(1 + \sum_{l=1}^{r-1} p_{[l]}^X)^\alpha - d_i^B \le 0$ since $t + p_i^B(1+\sum_{l=1}^{r-1} p_{[l]}^X)^\alpha < d_i^B$. Furthermore, from (iii) of Property 4, $C_j(S) - d_j^B = t + p_i^B(1+\sum_{l=1}^{r-1} p_{[l]}^X)^\alpha + p_j^B(1+\sum_{l=1}^{r-1} p_{[l]}^X + p_i^B)^\alpha - d_j^B \le 0$. Note that (17) is not considered because both J_i^B and J_j^B are jobs of agent B. As a result, S dominates S'. □

Property 5. If there is any unscheduled job J_i^B with $d_i^B < t$, the schedule is infeasible.

Proof. Since $d_i^B < t$, job J_i^B becomes tardy even though it is processed immediately. Therefore, the schedule is not feasible. □

Property 6. If there are two jobs J_i^X and J_j^B scheduled in the rth and $(r+1)$th positions, respectively, and job J_j^B satisfies $t + p_i^X(1 + \sum_{l=1}^{r-1} p_{[l]}^X)^\alpha > d_j^B$, the schedule is infeasible.

Proof. Since $t + p_i^X(1+\sum_{l=1}^{r-1} p_{[l]}^X)^\alpha > d_j^B$, job J_j^B becomes tardy. Therefore, the schedule is not feasible. □

A lower bound (LB) is also important to support an efficient search process for the B&B algorithm. The LB of a node is the expected minimum value of the objective function, which can be obtained by exploring the node. When the incumbent solution of the total weighted tardiness is z_c, an objective value better than z_c cannot be obtained by branching any node whose lower bound is larger than z_c (a criteria for fathoming). To calculate the LB of a certain node for the scheduling problem under consideration, we suggest the following procedure. For any node with partially scheduled and unscheduled jobs,

Step 1. Compute the total weighted tardiness of jobs already scheduled for agent A and denote this quantity by LB_PS.

Step 2. Let the largest due date of the unscheduled jobs for agent A be d.

Step 3. Consider a subproblem consisting of all unscheduled jobs for agent A with the objective of minimizing the total weighted tardiness, where the due dates are fixed to the common value d. If we do not consider aging effects, this subproblem can be optimized by scheduling jobs using the weighted shortest processing time (WSPT) rule [36]. Let LB_US denote the total weighted tardiness for the optimal solution. If there is no job for agent A, let LB_US = 0.

Step 4. For the node under consideration, let LB = LB_PS + LB_US.

3. Genetic Algorithm

Even though the efficiency of the B&B algorithm can be improved using the dominance properties and LB, its computational efficiency is relatively low because it is based on enumeration search. Thus, we propose an efficient genetic algorithm (GA) that is a metaheuristic for finding a near optimal solution. GAs are generally used to find solutions by mimicking the evolution process of an organism; they artificially contain the selection, crossover, and mutation procedures that occur in the course of evolution. The details of the suggested GA are discussed in the following subsections.

3.1. Representation Scheme of a Chromosome.

The first step of solving an optimization problem using a GA is to represent the genes on a chromosome. The most common binary encoding scheme uses 0 and 1, which is not suitable for our scheduling problem because a scheduling problem solution is a sequence of jobs. Therefore, we adopt permutation encoding designed for combinatorial optimization problems such as the travelling salesman problem [42]. Permutation encoding employs a structure of chromosomes as depicted in Figure 3, where one job corresponds to one gene. The positions of genes denote the orders of jobs in a sequence represented by the corresponding chromosome. To make the sequence feasible, we only consider schedules that include no tardy jobs for agent B.

3.2. Initial Population.

Metaheuristic solution algorithms, including GAs, have global search strategies that are significantly affected by the initial population. By using an appropriate initial population, a good solution can be obtained with a good fitness value or decreased computation time. We consider four different kinds of initial populations of size Q.

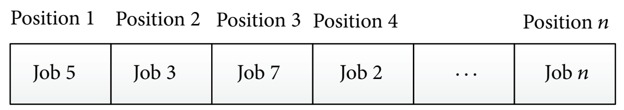

Figure 3: Representation of a chromosome using permutation encoding.

Initial Population 1. All jobs are scheduled randomly.

Initial Population 2. Jobs of agent B are arranged first using the earliest due date (EDD) rule, and continually jobs of agent A are arranged randomly.

Initial Population 3. Jobs of both agents A and B are arranged using the EDD rule, while considering the jobs of agent B first.

Initial Population 4. Jobs of agent B are arranged first using the EDD rule, and continually jobs of agent A are arranged using the weighted EDD (wEDD) rule.

3.3. Fitness Function. Our goal is to pass down a solution with promising features to the next generation to achieve continuous improvements; resultantly, repeated iterations from generation to generation are advantageous. A fitness function is used to evaluate GA solutions and measure how they satisfy the objective of the problem. A superior solution should have a larger fitness function value than all other solutions. Therefore, we define the fitness function inversely proportional to the total weighted tardiness of agent A as

$$f\_\text{value} = \frac{K}{\text{Total Weighted Tardiness}}, \tag{22}$$

where K is a constant.

3.4. Crossover Operation. A crossover operation is a genetic-type process used to produce new offspring where the features of two parent chromosomes are combined. The various crossover operators can be classified according to where the crossover occurs and whose traits are passed down. For example, a crossover operator, using one or two points, exchanges genes by choosing the corresponding points of parent chromosomes; in contrast, a uniform crossover operator makes offspring inherit features uniformly from parents. However, a general crossover operator is not suitable for the encoding scheme employed in this paper because duplication of jobs can occur in offspring. Thus, we apply the edge recombination operator (ERO), which can effectively generate a feasible solution to the TSP and other scheduling problems. The ERO generates offspring based on an edge map consisting of information for the edges between two adjacent jobs. The procedure for generating a new child is as follows [37]. Note that two offspring are generated from two parents.

Step 1. The edge map is constructed by analyzing two parent chromosomes and storing the edge information from two adjacent jobs. A schedule of offspring is generated based on the edge map. For example, consider two chromosomes P1: 2-3-1-4-5-6 and P2: 1-5-2-4-3-6. The edge map is derived as follows:

Job 1: 3-4-5
Job 2: 3-5-4
Job 3: 2-1-4-6
Job 4: 1-5-2-3
Job 5: 4-6-1-2
Job 6: 5-3

Step 2. The first job in the schedule is chosen randomly and removed from the edge map. If Job 1 is chosen as the first job, it is eliminated from the edge map.

Step 3. The next job is selected based on the connection information of the currently considered job in the edge map. If Job 1 was selected, Job 3, 4, or 5 can be selected as the next job. If there is no job that can be selected on the edge map, an arbitrary node can be accepted. However, before accepting one job, two mutation operators are applied sequentially, which are explained below.

Step 4. Steps 2 and 3 are repeated until there is no unscheduled job.

3.5. Mutation. Sometimes in nature an offspring chromosome is affected by mutations in the genetic process. A mutation is an unintended change of a genetic feature that results in the occurrence of new offspring with different features than the parental generation. This alteration can be a source of maintaining diversity in a species or as the foundation for evolution. A mutation operator, such as changing the generated sequence or causing a bit inversion, artificially imitates the real genetic operation occurring in nature. Using a simulated mutation operator can prevent a GA from converging to a local optimum. We develop the following two mutation operators occurring with a probability of p_m ($0 < p_m < 1$) that take the chromosomal structure and crossover operator into account, that is, mutation occurs when a randomly generated number is less than or equal to p_m. Mutation 1 is to consider a job not in the edge map and mutation 2 is to choose another job by rejecting the selected job.

Mutation 1. An edge not in the edge map may be selected even though the edge map of the current job is not empty.

Mutation 2. The selected job is rejected and one of remaining jobs is selected randomly.

3.6. Population Renewal and Termination Criteria. After a new generation with size Q is produced using the crossover and mutation operations, the fitness of each chromosome is evaluated. As a result, a new population of size Q is created by selecting chromosomes with high fitness values from the current population and newly generated offspring. The crossover and mutation operations are applied to the initial and new population by choosing parent chromosomes, where

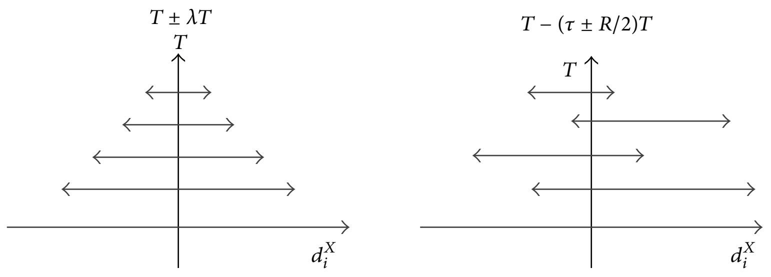

Figure 4: Ranges of due dates generated by two and three parameters.

a roulette wheel selection is performed by assigning selection probability values to chromosomes according to their fitness values. As a result, the solution space can be efficiently explored, and species diversity can be preserved.

Meanwhile, if an iteratively obtained solution meets predefined standards for stopping, the GA terminates since additional executions are insignificant. Examples of stopping conditions include reaching a fixed number of iterations or depleting the allocated computing capacity. We propose terminating the iterative process when there are no enhancements for N consecutive generations. This proposed termination condition uses the available computing power more flexibly than other criteria [43].

4. A Numerical Experiment

4.1. Experimental Design. We designed a numerical experiment to evaluate the performance of the proposed B&B algorithm and GA. Parameters for the scheduling problem under consideration were generated as follows. We set the value of the aging index α to 0.05 and considered three cases for the number of jobs; that is, $n = 8, 10, 12$. The normal processing time p_i^X and weight w_i^X for job J_i^X were generated using random integer values between 1 and 20, and both agents A and B processed the same number of jobs. An integer-valued due date d_i^X for job J_i^X was selected from a uniform distribution in the range

$$T - \left(\tau + \frac{R}{2}\right) T \le d_i^X \le T - \left(\tau - \frac{R}{2}\right) T, \qquad (23)$$

where T is the sum of normal processing times of all jobs, τ is the tardiness factor, and R is the due date range factor. Both τ and R are real numbers randomly generated between 0 and 1. From (23), we can generate a scheduling problem with jobs whose due dates are not too small or large. Moreover, due dates generated using three parameters (T, τ, and R) permit more various time windows including asymmetric regions that can be explored; this is not the case for time windows generated using only two parameters (λ and T) as shown in Figure 4. As a result, we are able to consider dynamic due date situations.

We planned a total of $3 \times 2 \times 4 = 24$ experiments; in particular, we considered three cases of n ($n = 8, 10, 12$), two tardiness factors ($\tau = 0.2, 0.4$), and four due date range factors ($R = 0.2, 0.4, 0.6, 0.8$). The reason for using only two levels of τ and four levels of R is that due dates generated using large values of τ are so tight that a job of agent B inevitably becomes tardy, which violates the feasibility condition. We randomly generated 30 problem instances for each configuration (n, τ, R). Additional parameters of the GA were determined experimentally; that is, $Q = 30$, $p_m = 0.01$, $K = 100$, and $N = 5$.

Performance of the B&B algorithm was analyzed by recording the average and standard deviation of the number of explored nodes and the elapsed execution time. However, the computational efficiency of the B&B algorithm was very low despite its ability to find an optimal solution. Thus, we assessed how effectively the proposed dominance properties and LB support the search process of the B&B algorithm. Despite its ability to execute in a short period of time, the GA cannot assure a global optimal solution. Since the quality of the obtained solution and computing efficiency are important, we evaluated the solution obtained using the proposed GA based on its proximity to the optimal solution; specifically, we used the mean and standard deviation of the error rate defined by

$$\text{Error rate}\,(\%) = \frac{V - V_O}{V_O} \times 100\,(\%), \qquad (24)$$

where V is the total weighted tardiness of agent A obtained by the GA and V_O is the optimal total weighted tardiness of agent A obtained by the B&B algorithm. The computing efficiency of the GA was also recorded.

4.2. Analysis of Experimental Results. We implemented the B&B algorithm and GA using MATLAB R2012 and performed the experiments on a PC platform with 16 GB RAM and an Intel(R) Core(TM) i5-3570 CPU 3.40 GHz. Table 1 summarizes the results of the experiment. Thirty problem instances were applied to each (n, τ, R) configuration, and we computed the mean and standard deviation of the execution time, the number of generated nodes for the B&B algorithm, and the error rate of the GA. We considered four different methods to generate an initial population GA$_i$ $(i = 1, 2, 3, 4)$, corresponding to the initial population i $(i = 1, 2, 3, 4)$ in Section 3.2.

The B&B algorithm suggested in this paper found optimal solutions in a reasonable amount of time. For example, for a scheduling problem with eight jobs, there exist 8! = 40,320 possible cases; the proposed B&B algorithm was able to find an optimal solution by exploring approximately 2,500–6,000 nodes, which is only 6–16% of all possible cases. This

Table 1: Experimental results of the B&B algorithm and GA.

n	τ	R	B&B				Error rate of the GA								Avg. computation time of GA			
			Number of nodes		Computation time (s)		GA_1 (R : R)		GA_2 (EDD : R)		GA_3 (EDD : EDD)		GA_4 (EDD : wEDD)		GA_1	GA_2	GA_3	GA_4
			Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.				
8	0.2	0.2	6,355.2	938.42	1.68	0.24	0.013	0.005	0.011	0.010	0.004	0.004	0.006	0.002	0.74	0.62	0.46	0.52
		0.4	6,048.0	1,341.33	1.62	0.33	0.019	0.007	0.009	0.009	0.009	0.010	0.011	0.007	0.73	0.66	0.45	0.47
		0.6	6,713.8	2,809.12	1.93	0.83	0.018	0.009	0.013	0.011	0.007	0.002	0.006	0.006	0.75	0.64	0.49	0.48
		0.8	7,507.8	4,212.69	2.07	1.38	0.021	0.008	0.018	0.012	0.008	0.005	0.012	0.004	0.77	0.67	0.52	0.54
	0.4	0.2	2,460.4	229.00	0.58	0.03	0.011	0.004	0.012	0.011	0.005	0.002	0.007	0.003	0.73	0.68	0.53	0.57
		0.4	2,578.8	859.68	0.61	0.55	0.009	0.003	0.019	0.008	0.008	0.005	0.006	0.002	0.72	0.65	0.57	0.60
		0.6	2,872.4	1,595.51	0.78	0.85	0.012	0.006	0.014	0.005	0.011	0.008	0.010	0.004	0.73	0.69	0.59	0.58
		0.8	2,987.8	2,135.12	0.82	1.16	0.011	0.008	0.009	0.007	0.013	0.010	0.012	0.001	0.79	0.70	0.54	0.55
10	0.2	0.2	228,568.0	43,947.85	66.91	13.23	0.008	0.004	0.007	0.005	0.004	0.001	0.005	0.002	1.11	0.88	0.78	0.77
		0.4	229,012.7	52,694.14	68.84	17.61	0.005	0.002	0.009	0.003	0.008	0.004	0.006	0.001	1.20	0.89	0.76	0.77
		0.6	254,233.1	65,395.87	79.99	20.80	0.009	0.005	0.008	0.008	0.007	0.005	0.010	0.003	1.18	0.88	0.73	0.75
		0.8	252,720.4	110,233.21	80.21	34.59	0.019	0.010	0.014	0.010	0.009	0.006	0.009	0.004	1.23	0.84	0.74	0.79
	0.4	0.2	24,394.4	7,282.12	6.53	1.77	0.008	0.002	0.004	0.002	0.005	0.003	0.005	0.002	1.12	0.93	0.76	0.78
		0.4	25,259.2	9,219.01	7.30	2.14	0.007	0.004	0.011	0.003	0.007	0.002	0.009	0.003	1.22	0.88	0.74	0.75
		0.6	25,560.3	14,496.15	7.80	4.75	0.014	0.007	0.012	0.004	0.008	0.002	0.004	0.004	1.19	0.86	0.71	0.76
		0.8	26,344.11	22,289.75	9.01	5.52	0.009	0.009	0.008	0.007	0.008	0.004	0.010	0.003	1.17	0.87	0.73	0.72
12	0.2	0.2	2,266,093.2	438,187.13	978.87	162.11	0.008	0.004	0.007	0.004	0.012	0.002	0.009	0.004	2.12	1.56	1.32	1.28
		0.4	2,680,916.3	1,235,682.2	1,056.78	553.86	0.007	0.005	0.010	0.005	0.008	0.003	0.007	0.005	2.08	1.54	1.27	1.29
		0.6	2,810,094	1,960,845.3	1,147.58	828.54	0.020	0.002	0.015	0.006	0.010	0.004	0.004	0.006	2.07	1.59	1.33	1.30
		0.8	2,746,277.3	2,151,950.5	1,220.55	1,020.48	0.014	0.008	0.012	0.004	0.009	0.005	0.008	0.007	2.06	1.61	1.35	1.34
	0.4	0.2	284,563.7	156,255.3	95.77	51.40	0.004	0.004	0.004	0.002	0.009	0.004	0.005	0.002	2.03	1.60	1.28	1.32
		0.4	295,702.9	210,762.7	97.67	60.25	0.005	0.001	0.005	0.001	0.007	0.004	0.002	0.002	2.04	1.58	1.29	1.33
		0.6	295,666.3	246,098.5	104.55	71.03	0.002	0.001	0.004	0.003	0.010	0.008	0.008	0.004	2.20	1.59	1.31	1.28
		0.8	296,856.3	291,585.5	108.94	79.36	0.011	0.005	0.008	0.002	0.009	0.008	0.004	0.003	2.18	1.57	1.34	1.27

ratio (i.e., the proportion of nodes explored to the total number of possible nodes), decreased as the number of jobs increased. Even for a scheduling problem with 12 jobs, which is the largest job size among the experimental configurations, the optimal solution could be found by visiting only 1% of all possible cases. These results imply that the suggested dominance properties and LB successfully support the search process of the B&B algorithm.

In terms of parameters, we observed that the number of nodes explored to derive an optimal solution decreased in configurations with a high value of τ, if the number of jobs remained constant. For $n = 10$ and $R = 0.2$, the scheduling problems with $\tau = 0.2$ required an average of 241,133 visited nodes to obtain an optimal solution, whereas scheduling problems with $\tau = 0.4$ required 25,389 visited nodes, that is, 10% of the former case. Increasing τ shortens the due date in (23). A tight due date reduces the number of nodes that must be visited since many nodes violate the dominance properties of feasibility, rendering them fathomed.

We also observed the influence of R on the outcome of configuration $(n, \tau) = (10, 0.2)$. As R increased from 0.2 to 0.8, the standard deviation of the number of nodes and execution times by the B&B increased by a factor of 2.5, that is, 43,947 to 110,233. This is because R and τ determine the due date, especially its deviation as in (23). Therefore, large values of R generate jobs with largely deviated due dates compared to jobs generated by low values of R. This deviation of due dates causes a significant difference in the standard deviation of the number of nodes even though the average number of nodes is similar. We observed the deviation was maximized in configuration $(n, \tau) = (8, 0.4)$; specifically, the largest standard deviation of the number of explored nodes was 10 times larger than the smallest one. This means that the number of nodes visited varies depending on the combination of parameters (τ, R) although the number of jobs n remains the same. As a result, there is a close relationship between the performance of the B&B algorithm and parameters (τ, R), which strongly influences the due dates.

We also determined that the GA was more efficient than the B&B algorithm because it took only less than 3 seconds, even in the worst case, and had a low error rate of approximately 1%. Furthermore, the GA was rarely affected by variations in parameters (τ, R); on the other hand, the mean and standard deviation of the B&B algorithm were drastically affected by the values of (τ, R). As the number of jobs n increased, the execution time of the GA increased robustly from 0.76 to 1.11 and 2.12 seconds, whereas those of the B&B algorithm increased from 1.68 to 66.91 and 979.81 seconds. The execution times of the GA were only affected by the number of jobs n, that is, changes in (τ, R) had no effect. The error rates of GA_i $(i = 1, 2, 3, 4)$ maintained a stable tendency around 1% and were independent of configuration changes.

We applied the paired t-test to statistically verify the performance differences of the four proposed GAs in terms of their error rates. We defined the variable d_i as the difference of the ith paired values of error rates of two corresponding GAs and computed the sample mean difference $\overline{d}$ and sample standard deviation s_d as follows:

$$\overline{d} = \frac{1}{n} \sum_{i=1}^{n} d_i,$$

Table 2: Results of paired t-test between two GAs.

Comparison target	t-statistic	Acceptance of H_0
GA_1 versus GA_2	1.2103	Accept ($\lvert t\rvert < 2.045$)
GA_1 versus GA_3	2.8579	Reject ($\lvert t\rvert > 2.045$)
GA_1 versus GA_4	3.8179	Reject ($\lvert t\rvert > 2.045$)
GA_2 versus GA_3	2.3929	Reject ($\lvert t\rvert > 2.045$)
GA_2 versus GA_4	3.5189	Reject ($\lvert t\rvert > 2.045$)
GA_3 versus GA_4	1.6627	Accept ($\lvert t\rvert < 2.045$)

$$s_d = \sqrt{\frac{\sum_{i=1}^{n} \left(d_i - \overline{d}\right)^2}{n-1}}, \tag{25}$$

where $n = 30$. We established the following hypothesis set

$$\begin{aligned} H_0 &: \mu_d = 0, \\ H_1 &: \mu_d \neq 0, \end{aligned} \tag{26}$$

where hypothesis H_1 means that the performance of two GAs under consideration is significantly different. Because $\overline{d}$ has a normal distribution with an unknown variance, using the Central Limit Theorem [44], the test t-statistic can be defined with 29 degrees of freedom as

$$t = \frac{\overline{d}}{s_d / \sqrt{n}}. \tag{27}$$

For a 95% confidence level, the rejection region of this test is $|t| > t_{0.025,29} = 2.045$; the t-statistics calculated for the error rates are given in Table 2. From the acceptance and rejection results in Table 2, we observed that there are two groups of GAs, $(\text{GA}_1, \text{GA}_2)$ and $(\text{GA}_3, \text{GA}_4)$, with similar error rates, whereas $(\text{GA}_1, \text{GA}_2)$ and $(\text{GA}_3, \text{GA}_4)$ have significant differences.

We also determined if the error rate for one group of GAs was larger than that of another group of GAs. We considered additional paired t-test and established a hypothesis set based on the experimental result such that $\overline{d} = \overline{X}_1 - \overline{X}_3 > 0$ as follows:

$$\begin{aligned} H_0 &: \mu_d^{13} \leq 0, \\ H_1 &: \mu_d^{13} > 0, \end{aligned} \tag{28}$$

where $H_1 : \mu_d^{13} > 0$ means that the error rate of GA_1 is larger than that of GA_3. We compared two GAs using a one-tailed t-test with the rejection region $t > t_{0.05,29} = 1.699$. The t-statistic calculated for the difference between GA_1 and GA_3 was $t = 2.8579$ as shown in Table 2, which is in the rejection region. Thus, we reject H_0 and conclude that $(\text{GA}_3, \text{GA}_4)$ is superior to $(\text{GA}_1, \text{GA}_2)$.

The experimental results imply that the accuracy and performance of the GA can be enhanced by using an initial population that is more suitable to the target problem. More specifically, a superior GA could be developed to find a suitable solution to the scheduling problem under consideration by assigning jobs of agent B first according to the EDD rule and continually the jobs of agent A according to either the EDD or wEDD rules. The EDD and wEDD rules are advantageous scheduling policies with respect to the due date and tardiness, which positively affects the objective function considered in this paper.

5. Conclusion

We considered a single-machine scheduling problem with two competing agents and a sum-of-processing-times-based aging effect. The objective function was to minimize the total weighted tardiness for the first agent under the restriction that no tardy jobs are permitted for the second agent. We suggested a B&B algorithm with efficient dominance properties and LB to find an optimal solution. Because this scheduling problem is NP-hard, we also developed an efficient GA by designing an appropriate representation structure of chromosomes, valuable initial populations, and crossover and mutation operators. To verify the performance of the suggested algorithms, we conducted a numerical experiment using randomly generated problem instances.

To date, very little research has been conducted on performance measures that consider tardiness for a two-agent, single-machine scheduling based on the sum-of-processing-times. This kind of scheduling problem is closely related to the due date and additional penalties accrued on top of existing costs. The due date is a crucial index for a firm to maintain a solid relationship with a client. Therefore, we anticipate that these results will be used to construct a new solution framework for scheduling problems with sum-of-processing-times-based aging effects in actual industrial fields such as rolling mill and metallurgical processes in the steel industry or remedying an epidemic process.

In the future, we will consider flow-shop, parallel machines, or a combination of other objective functions. Moreover, learning or aging effects with different functions will be considered as additional options. Furthermore, we will consider other metaheuristics such as simulated annealing or particle swarm optimization as potential solution approaches.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research was supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract UD140022PD, Korea, and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2014R1A1A2057194).

References

[1] M. J. Soomer and G. J. Franx, "Scheduling aircraft landings using airlines' preferences," *European Journal of Operational Research*, vol. 190, no. 1, pp. 277–291, 2008.

[2] M. A. Kubzin and V. A. Strusevich, "Planning machine maintenance in two-machine shop scheduling," *Operations Research*, vol. 54, no. 4, pp. 789–800, 2006.

[3] H. Cres and H. Moulin, "Scheduling with opting out: improving upon random priority," *Operations Research*, vol. 49, no. 4, pp. 565–577, 2001.

[4] K. R. Baker and J. C. Smith, "A multiple-criterion model for machine scheduling," *Journal of Scheduling*, vol. 6, no. 1, pp. 7–16, 2003.

[5] A. Agnetis, P. B. Mirchandani, D. Pacciarelli, and A. Pacifici, "Scheduling problems with two competing agents," *Operations Research*, vol. 52, no. 2, pp. 229–242, 2004.

[6] A. Agnetis, D. Pacciarelli, and A. Pacifici, "Multi-agent single machine scheduling," *Annals of Operations Research*, vol. 150, no. 1, pp. 3–15, 2007.

[7] T. C. Cheng, C. T. Ng, and J. J. Yuan, "Multi-agent scheduling on a single machine with max-form criteria," *European Journal of Operational Research*, vol. 188, no. 2, pp. 603–609, 2008.

[8] J. Y. T. Leung, M. Pinedo, and G. Wan, "Competitive two-agent scheduling and its applications," *Operations Research*, vol. 58, no. 2, pp. 458–469, 2010.

[9] C. T. Ng, T. C. Cheng, and J. J. Yuan, "A note on the complexity of the problem of two-agent scheduling on a single machine," *Journal of Combinatorial Optimization*, vol. 12, no. 4, pp. 387–394, 2006.

[10] H. Balasubramanian, J. Fowler, A. Keha, and M. Pfund, "Scheduling interfering job sets on parallel machines," *European Journal of Operational Research*, vol. 199, no. 1, pp. 55–67, 2009.

[11] N. H. Tuong and A. Soukhal, "Interfering job set scheduling on two-operation three-machine flowshop," in *Proceedings of the International Conference on Computing and Communication Technologies (RIVF '09)*, pp. 1–5, IEEE, July 2009.

[12] W. C. Lee, S. K. Chen, and C. C. Wu, "Branch-and-bound and simulated annealing algorithms for a two-agent scheduling problem," *Expert Systems with Applications*, vol. 37, no. 9, pp. 6594–6601, 2010.

[13] Y. Yin, T. C. E. Cheng, X. Yang, and C. C. Wu, "Two-agent single-machine scheduling with unrestricted due date assignment," *Computers & Industrial Engineering*, vol. 79, pp. 148–155, 2015.

[14] T. C. E. Cheng and G. Wang, "Single machine scheduling with learning effect considerations," *Annals of Operations Research*, vol. 98, no. 1–4, pp. 273–290, 2000.

[15] J. N. D. Gupta and S. K. Gupta, "Single facility scheduling with nonlinear processing times," *Computers and Industrial Engineering*, vol. 14, no. 4, pp. 387–393, 1988.

[16] S. Browne and U. Yechiali, "Scheduling deteriorating jobs on a single processor," *Operations Research*, vol. 38, no. 3, pp. 495–498, 1990.

[17] A. Bachman and A. Janiak, "Scheduling jobs with position-dependent processing times," *Journal of the Operational Research Society*, vol. 55, no. 3, pp. 257–264, 2004.

[18] J.-B. Wang and Z.-Q. Xia, "Flow-shop scheduling with a learning effect," *Journal of the Operational Research Society*, vol. 56, no. 11, pp. 1325–1330, 2005.

[19] G. Mosheiov, “Scheduling jobs under simple linear deterioration,” *Computers and Operations Research*, vol. 21, no. 6, pp. 653–659, 1994.

[20] P. S. Sundararaghavan and A. S. Kunnathur, “Single machine scheduling with start time dependent processing times: some solvable cases,” *European Journal of Operational Research*, vol. 78, no. 3, pp. 394–403, 1994.

[21] A. Bachman, T. C. E. Cheng, A. Janiak, and C. T. Ng, “Scheduling start time dependent jobs to minimize the total weighted completion time,” *Journal of the Operational Research Society*, vol. 53, no. 6, pp. 688–693, 2002.

[22] B. Alidaee and N. K. Womer, “Scheduling with time dependent processing times: review and extensions,” *Journal of the Operational Research Society*, vol. 50, no. 7, pp. 711–720, 1999.

[23] Y. Yin, T. C. E. Cheng, W. Long, C. C. Wu, and L. Jun, “Two-agent single-machine scheduling with deteriorating jobs,” *Computers & Industrial Engineering*, vol. 81, pp. 177–185, 2015.

[24] D. Biskup, “Single-machine scheduling with learning considerations,” *European Journal of Operational Research*, vol. 115, no. 1, pp. 173–178, 1999.

[25] G. Mosheiov, “Parallel machine scheduling with a learning effect,” *Journal of the Operational Research Society*, vol. 52, no. 10, pp. 1165–1169, 2001.

[26] G. Mosheiov, “Scheduling problems with a learning effect,” *European Journal of Operational Research*, vol. 132, no. 3, pp. 687–693, 2001.

[27] G. Mosheiov and J. B. Sidney, “Note on scheduling with general learning curves to minimize the number of tardy jobs,” *Journal of the Operational Research Society*, vol. 56, no. 1, pp. 110–112, 2005.

[28] W.-C. Lee and C.-C. Wu, “Minimizing total completion time in a two-machine flowshop with a learning effect,” *International Journal of Production Economics*, vol. 88, no. 1, pp. 85–93, 2004.

[29] T. E. Cheng, Q. Ding, and B. M. Lin, “A concise survey of scheduling with time-dependent processing times,” *European Journal of Operational Research*, vol. 152, no. 1, pp. 1–13, 2004.

[30] W.-H. Kuo and D.-L. Yang, “Minimizing the total completion time in a single-machine scheduling problem with a time-dependent learning effect,” *European Journal of Operational Research*, vol. 174, no. 2, pp. 1184–1190, 2006.

[31] Y. Yin, D. Xu, K. Sun, and H. Li, “Some scheduling problems with general position-dependent and time-dependent learning effects,” *Information Sciences*, vol. 179, no. 14, pp. 2416–2425, 2009.

[32] Y. Yin, M. Liu, J. Hao, and M. Zhou, “Single-machine scheduling with job-position-dependent learning and time-dependent deterioration,” *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems*, vol. 42, no. 1, pp. 192–200, 2012.

[33] P. Liu, N. Yi, X. Zhou, and H. Gong, “Scheduling two agents with sum-of-processing-times-based deterioration on a single machine,” *Applied Mathematics and Computation*, vol. 219, no. 17, pp. 8848–8855, 2013.

[34] W.-H. Wu, S.-R. Cheng, C.-C. Wu, and Y. Yin, “Ant colony algorithms for a two-agent scheduling with sum-of processing times-based learning and deteriorating considerations,” *Journal of Intelligent Manufacturing*, vol. 23, no. 5, pp. 1985–1993, 2012.

[35] W.-H. Wu, “A two-agent single-machine scheduling problem with learning and deteriorating considerations,” *Mathematical Problems in Engineering*, vol. 2013, Article ID 648082, 18 pages, 2013.

[36] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, Springer, Berlin, Germany, 2012.

[37] L. D. Whitley, T. Starkweather, and D. A. Fuquay, “Scheduling problems and traveling salesmen: the genetic edge recombination operator,” in *Proceedings of the 3rd International Conference on Genetic Algorithms (ICGA '89)*, pp. 133–140, 1989.

[38] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Kan, “Optimization and approximation in deterministic sequencing and scheduling: a survey,” *Annals of Discrete Mathematics*, vol. 5, pp. 287–326, 1979.

[39] K. R. Baker and D. Trietsch, *Principles of Sequencing and Scheduling*, John Wiley & Sons, New York, NY, USA, 2009.

[40] A. Jouglet and J. Carlier, “Dominance rules in combinatorial optimization problems,” *European Journal of Operational Research*, vol. 212, no. 3, pp. 433–444, 2011.

[41] R. Rudek, “Some single-machine scheduling problems with the extended sum-of-processing-time-based aging effect,” *The International Journal of Advanced Manufacturing Technology*, vol. 59, no. 1–4, pp. 299–309, 2012.

[42] A. Kumar, “Encoding schemes in genetic algorithm,” *International Journal of Advanced Research in IT and Engineering*, vol. 2, no. 3, pp. 1–7, 2013.

[43] S. N. Sivanandam and S. N. Deepa, *Introduction to Genetic Algorithms*, Springer, Berlin, Germany, 2008.

[44] A. J. Hayter, *Probability and Statistics*, International Thomson PUB, 1996.