IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received November 5, 2019, accepted November 21, 2019, date of publication November 28, 2019, date of current version December 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2956551

Algorithms for Minimizing Resource Consumption Over Multiple Machines With a Common Due Window

JEN-YA WANG
Department of Computer Science and Information Management, Hungkuang University, Taichung 43302, Taiwan
e-mail: jywang@sunrise.hk.edu.tw

This work was partially supported in part by the Ministry of Science and Technology of Taiwan, R.O.C., under Project MOST-108-2410-H-241-006.

ABSTRACT Scheduling on multiple machines has attracted considerable attention recently. However, most of traditional studies focused only on commercial cost and customer satisfaction. In fact, we are able to alleviate environmental damages via proper scheduling. This study explores a multi-machine scheduling problem with a due window. The objective is to minimize the total cost (including environmental cost). We develop a branch-and-bound algorithm (B&B) and two complementary lower bounds for optimally solving the problem when $n < 16$. Moreover, we propose an imperialist competitive algorithm (ICA) to obtain near-optimal schedules for large problem instances. Experimental results are provided to show the performance of the proposed algorithms.

INDEX TERMS Chemical industry, multi-machine scheduling, resource consumption, due window, imperialist competitive algorithm, branch-and-bound algorithm.

I. INTRODUCTION

Job scheduling has been studied in both industry and academia for long. A good scheduling algorithm meets its corresponding objectives, e.g., minimum tardiness or maximum profit. Due to the advances in technology, traditional scheduling algorithms may not fit today's environments. For example, parallel machines are commonly seen in many factories. All the jobs allocated to multiple machines should be considered as a whole. Scheduling each single machine by some traditional algorithms may not lead to the global maximum profit. Consequently, new issues need to be reinvestigated and new scheduling algorithms are called for.

In general, multi-machine scheduling is more difficult than single-machine scheduling. Not only the position order of jobs but also the capabilities of machines need to be taken into account. There are some multi-machine scheduling examples with different objectives. Lee and Wu [1] studied a multi-machine scheduling problem whose objective is to minimize the total completion time. In this problem, regular maintenance for each machine is inevitable. A metaheuristic was proposed to solve this NP-hard problem. Balin [2] solved another multi-machine scheduling problem. In this problem, machines are of different capabilities. Again, this problem is not easy to be solved optimally. Another metaheuristic was proposed to provide near-optimal schedules. Kayvanfar *et al.* [3] aimed to minimize makespan, earliness, tardiness in a parallel-machine environment. They also proposed a metaheuristic for this problem. In [4], CPUs could vary their speeds for facing different workloads in a clustered environment. Since low speed means less power consumption, these CPUs can be scheduled to avoid tardiness and save power. Thiruvady *et al.* [5] considered a scheduling problem in the mining industry. Multiple parallel machines were employed to minimize the total tardiness. In sum, the above examples show that multi-machine scheduling is a trend in today's industries. In [6], a multi-machine scheduling problem was considered to minimize the total tardiness under some restrictions on PM 2.5 emission. A branch-and-bound algorithm was developed to solve this this eco-aware scheduling problem.

Due to the rise of environmental awareness, traditional scheduling for pursuing maximum efficiency or minimum cost does not meet today's requirements. In a multi-machine environment, machines are of different power consumption rates, carbon emissions, and capacities. Fang and Lin [4] considered a multi-server computing environment. Not all CPUs need to be active at off-peak nighttime. Low power

The associate editor coordinating the review of this manuscript and approving it for publication was Jihwan P. Choi.

consumption is still able to meet the requirement of tardiness. Ji *et al.* [7] allocated a batch of jobs to machines with different capabilities. Their objective was to minimize the environmental cost with a restriction on a specified makespan. Given a specified carbon emission, Yeh *et al.* [8] aimed to minimize makespan. The above examples show that there exist some contradictions when we schedule jobs for deriving customer satisfaction as well as prevent our environments from being damaged. In the multi-machine environment described in [9], resources were limited. Consequently, some duplicate jobs were partially omitted to reduce processing time. Due to this idea, the resulting schedules could be more environmental-friendly than before.

Both tardiness and earliness are important research topics in job scheduling, since they directly involve customer satisfaction. A real-world mining example was shown in [5]. In this example, tardiness means missing shipping date, causing customer complaint, and increasing cost. Anzanello *et al.* [10] discussed how to improve tardiness and decrease cost by learning. With the consideration of learning effect, job scheduling becomes more different than before. On the other hand, some job scheduling problems must avoid or alleviate earliness due to the properties of their products, e.g., perishable food. There are very few studies whose research interests focus only on earliness. It is interesting that such scheduling is only found in chemical industry; e.g., [11]–[14]. This is because these chemicals need batch processing, stirring, or coagulating. In the field, the damage caused by earliness is much more severe than that caused by tardiness.

Due window is also a rising research topic. If this topic is ignored, earliness brings more inventory cost and tardiness causes worse customer satisfaction. However, jobs scheduled within a due window require neither earliness nor tardiness as possible as we can. Given a fixed due window and a maximum tardiness, Su and Tien [15] aimed to minimize the standard deviation of completion times. In [16]–[18], researchers adjusted the size of due window to reduce earliness or tardiness. In light of these studies, we learn that due window is not always fixed. It may be unknown and needs to be determined.

The applications of multi-machine scheduling are growing everywhere such as textile, electronics, construction, information industries. When scheduling multiple machines, we may find that some objectives are conflicted, e.g., tardiness and energy saving. To achieve no tardiness, we may choose some fast but energy-consuming machines. That is, customer satisfaction is at the cost of environmental damages. In light of this observation, how to balance multiple objectives over multiple machines draws increased attention.

When scheduling on multiple machines, outsourcing is also a choice for reducing tardiness or completion time. In [19], this is a multi-machine environment. Outsourcing is allowed if needed. Since the cost of outsourcing is much higher than the cost paid by producers themselves, outsourcing is the last resort to avoid tardiness. However, scheduling with outsourcing makes the producers have more flexibility to retain customer satisfaction.

TABLE 1. Three kinds of milling machines.

Asphalt milling machine	Processing speed	Environmental cost
Wirtgen W 35 DC	57 HP	43 kW
Wirtgen W 50 DC	127 HP	95 kW
Wirtgen WR 2000 XL	422 HP	315 kW

TABLE 2. An industrial problem instance.

Region	Processing time	Weight	Due window
1	40 hours	1.2	
2	20 hours	1.0	
3	40 hours	1.1	
4	20 hours	1.0	
5	20 hours	1.3	2018/11/17
6	80 hours	1.0	
7	20 hours	1.0	
8	40 hours	1.0	
9	40 hours	1.0	
10	20 hours	1.1	~
11	20 hours	1.2	
12	40 hours	1.0	
13	40 hours	1.0	
14	20 hours	1.0	
15	80 hours	1.0	
16	60 hours	1.0	
17	20 hours	1.0	2018/11/25
18	60 hours	1.0	
19	40 hours	1.1	
20	40 hours	1.0	
21	60 hours	1.0	

This study is motivated by the following real-world example. The example of multi-machine scheduling with due window is shown in Tables 1 and 2. There are three asphalt milling machines whose processing speeds and environmental costs are all different. Twenty one neighboring regions need to be paved with stone and asphalt within 17th Nov. and 25th Nov. The processing cost of each region can be roughly estimated by its area. However, the density of manholes or handholes slightly affects the complexity of trimming and the labor of traffic control. Earliness causes extra cost, since traffic signal installation or planting engineering had better be done first. Tardiness also leads to contractual fine. Consequently, all the jobs had better be finished within the due window. However, the internal resource (e.g., machines) is limited, so jobs may not be completed on schedule and outsourcing may be a choice. Clearly, such scheduling is meaningful and worthwhile to explore in more detail.

II. RELATED WORK

This study aims to balance scheduling performance, customer satisfaction, and environmental protection. Some related issues include multi-machine scheduling, energy saving, and due window. They are discussed as follows.

A. MULTI-MACHINE SCHEDULING

Multi-machine scheduling means high performance but also brings considerable complexity. Though most of such problems are NP-hard, e.g., [4], [7], [8], multi-machine scheduling is still needed in today's industries. As a company grows, it naturally purchases many facilities at different stages. Moreover, the specifications of machines may be different. Some machines waste power and emit excess carbon dioxide. However, during some time, the production capacity is greater than the requested amount. Consequently, some machines causing high pollution and high power consumption can to be shut down for a while.

In general, to minimize makespan or tardiness, we use all available machines and resources as possible as we can. From the past studies [20]–[22], we learn that no machine could be absent when minimizing completion time or tardiness. Some examples are given below.

In [3], jobs were non-preemptive and machines are non-identical. The objective was to minimize weighted makespan, total earliness, and total tardiness at the same time. It is noted that makespan directly affects customer satisfaction and grows in $O(n)$. Earliness and tardiness are also indicators of customer satisfaction and grow in $O(n^2)$. To achieve the minimum earliness, they may obtain a large makespan. Moreover, not all jobs were processed at time 0, i.e., some machines were idle at some time. In [21], machines were identical and jobs were preemptive. The objective was to minimize the total tardiness. Earliness was not considered in this study. Therefore, each optimal schedule started at time 0 and no machine was idle. Even so, this problem is still NP-hard. In a multi-machine environment [23], jobs had a common due date and learning effect and deterioration effect were considered. The objective was to minimize both tardiness and earliness. Some similar jobs could be accelerated because of learning effect and some jobs needed more processing time due to bad weather or darkness. Again, since earliness was considered, jobs did not always start at time 0. Although machines were identical, the problem is NP-hard. Moreover, tardiness or earliness was improved by multi-machine scheduling. However, the complexity and difficulty increased, since more resources implied more decision making.

In sum, compared with single-machine scheduling, multi-machine scheduling become an increasingly important trend. All the above studies aimed to increase productivity and shorten completion time without consideration of environmental issues. However, there might be some schedules which achieve similar results but are more eco-friendly. So this issue is worth considering.

B. ENVIRONMENTAL ISSUES

There are various environmental considerations when we perform multi-machine scheduling. For example, different performance indicators were considered in a data center that uses heterogeneous servers to meet users' various requests [24]. These indicators included energy cost, carbon emission, workload balance, and processing efficiency. The processing efficiency varied with carbon emission. It was debatable that all processing units are always keeping in high-performance status. On the other hand, Galizia and Quarati [25] consumed less energy to accomplish the same objective by adjusting the workloads of processing units in a grid environment.

In a multi-machine environment, an eco-aware schedule is not necessarily cost-efficient. As long as machines are different, power consumption, carbon emission, productivity are different. As shown in [7], to minimize makespan, a greedy approach would choose the fastest machine in a heterogeneous environment. However, the greedy approach would select the most environment-friendly machine if low carbon emission is the top priority. In [26], there were different sources of energy, e.g., immediately available steam or external gas. Their environmental costs were different. In the case of abundant resources, energy resulting in minor environmental damage is preferred. That is, different kinds of energy can be managed and scheduled.

Environmental protection and cost efficiency have always been a dilemma and we need to weigh their severities. In [27], there were several pumps of different capacities and a reservoir storing groundwater from different sources. Because the nighttime electricity was charged at a low price and frequent pump switching will caused wear and tear, we had better continue to pump water for a long time at night. However, some places were not abundant in water, so groundwater should be pumped intermittently in order not to cause land subsidence. For protecting environment, we were forced to pump groundwater during some daytime. Therefore, multi-machine scheduling could help us to meet the conflicted objectives. Another example was a multi-CPU scheduling problem [4]. Traditional multi-CPU scheduling algorithms focused on partitioning and sequencing, since their major concern was load balance. Note that the processing speed is adjustable in today's computing environments. High-speed brings us high performance as well as high power consumption. Consequently, load balance is no longer the only concern. A compromise between completion time and power consumption should be found. Multi-machine scheduling is an answer to such problems.

In short, job scheduling becomes more complex if environmental issues are taken into account. The final decision may be different from the optimal schedule suggested by traditional algorithms. Environmental protection might be at the cost of makespan or tardiness. Because traditional scheduling cannot solve today's problems, new eco-aware algorithms are called for.

C. DUE WINDOW

Due-window scheduling is commonly seen in many fields and it manages to compress all the completion times within a period of time. In [28], many kinds of due window were discussed. Due window originated from JIT (Just-In-Time). With due window, we guaranteed customers an expected delivery time. If all completion times could be controlled within a time period, a producer could reduce inventory, overproduction and avoid needless waiting time, extra delivery time, excess processing time, defective goods. The practical applications of due window can be found in semiconductor, project management (e.g., PERT/CPM), network industries, etc. Taking VoD (Video on Demand) as an example, the arrival times of multimedia network packets need to be controlled within a period of time, since early packets will clog memory buffers and tardy packets will cause jerky or choppy videos. The packets must arrive at an expected time interval.

Janiak *et al.* [28] divided due-window scheduling problems into four types. First, a common due window was specified for all jobs in advance, e.g., [29]–[32]. Their objectives were minimum earliness and tardiness. Such problems were almost NP-hard. Second, each job was assigned to a due window in advance, e.g., [33], [34]. These problems were also NP-hard and aimed to reduce earliness and tardiness. Third, a window size for all jobs was given in advance and we need to determine the beginning time of the common due window, e.g., [35], [36]. The problems looked simple, but they were actually not the case. Even for a single schedule, there are many possible starting points for the due window. So these problems are still NP-hard. Fourth, the starting point and the size of a due window were all unknowns, e.g., [37]–[39]. Since there were multiple unknowns, it meant we have more freedom to schedule jobs. Such problems were polynomially solvable and could be solved optimally in $O(n^3)$.

Some past research aimed to minimize earliness and tardiness simultaneously [23], [40], [41]. Min and Cheng [40] pointed that on-time delivery is important in textile, industrial, electronics industries, etc. Therefore, scheduling with due window is meaningful. If machines are heterogeneous, scheduling becomes more complicated. Kayvanfar *et al.* [41] simultaneously minimized the tardiness and earliness on a single machine. Although no due window was explicitly specified in this study, we could suppose that there exists a due window of size 0 for minimizing tardiness and earliness. The applications of due window were common in a data center with multiple servers [24], [42], [43]. Accelerating disks or CPUs help us meet the requirements during rush hour and decelerating them can save energy at nighttime. How to achieve a balance on both is important. Interestingly, industrial scheduling and computer science pursue the same purpose. Again, Toksari and Guner [23] considered both earliness and tardiness. All jobs were assigned a common delivery date, i.e., a due window of size 0. In this problem, earliness and tardiness would cause different penalties. Such a problem was also NP-hard.

Four points are drawn from the above observations. First, multi-machine scheduling has become an important research issue. Related applications are seen in many fields, e.g., manufacturing plants, data centers, pumping stations. Second, the related eco-aware issues are discussed less than 10% in multi-machine scheduling. This is because traditional scheduling emphasized customer satisfaction and tardiness was strictly controlled or even not allowed. Consequently, environment protection was sacrificed or ignored. Third, due window and environment protection are seldom considered simultaneously. This is because due window is also a customer-oriented concept and usually conflicts with environment protection. Fourth, since most of these problems are NP-hard, some metaheuristic algorithms are needed for generating near-optimal schedules for practical use. Table 3 summarizes past research regarding the three topics of multi-machine scheduling, environmental protection, and due window. It is clear that the three issues are seldom studied simultaneously, especially for heterogeneous machines. Therefore, the three issues are worth detailed exploration.

TABLE 3. Related research regarding the three topics.

Issue / Objective	Multi-machine scheduling	Environmental protection	Due window
Earliness	[3, 44, 45]		[45-47]
Number of Early Jobs	[32, 35, 48]		[32]
Tardiness	[3, 20, 49-54]	[6, 9, 55]	[45, 47, 56, 57]
Number of Tardy Jobs	[32, 35, 58]		[32, 59]
Makespan	[7, 22, 52, 60-65]	[7, 66, 67]	
Completion Time	[68-70]		[15]
Environmental Cost	[4, 7, 24, 27, 55, 66, 71]	[4, 7, 24, 27, 72]	
Window Assignment	[30, 34, 45, 73-76]		[30, 34, 73, 74, 77-79]

III. PROBLEM FORMULATION

Consider the real-world application shown in Tables 1 and 2 and formulate the job scheduling problem as follows. There are n jobs to be scheduled on m machines with a common due window $[e, d]$, where $e \le d$. All the jobs are available at time 0. Each machine can process at most one job at a time and each job can be allocated to any machine. Job j has a processing time p_j and a processing cost coefficient w_j. Machine i has a speed v_i and a unit production cost c_i(resource consumption). For example, the resource consumption of job j on machine i is $p_j w_j c_i / v_i$. Without loss of generality, we assume that $c_1/v_1 \le c_2/v_2 \le \ldots \le c_m/v_m$. Intuitively, v_i and c_i can be regarded as the second and third columns shown in Table 1. It is clear that machine 1 is the most economic. For a schedule π, the completion time of job j is denoted by $C_j(\pi)$ and the earliness and tardiness of job j are determined as $E_j(\pi) = \max\{0, e - C_j(\pi)\}$ and $T_j(\pi) = \max\{0, C_j(\pi) - d\}$, respectively. The objective is to minimize the total cost, i.e., $\sum_{i=1}^{m} \sum_{j@M_i}^{(p_j w_j c_i / v_i)}$, subject to $\sum_j (E_j(\pi) + T_j(\pi)) \le A$, where $j@M_i$ means job j is allocated to machine i and A is a specified limit. To meet this requirement, a decision maker can outsource some jobs at a unit production cost of c_{m+1}, where $c_{m+1} \ge c_m/v_m$.

A problem instance is shown in Fig. 1. Let $A = 18$, $[e, d] = [10, 12]$, $m = 2$, $n = 8$, $v_i = 2, 1$, $c_i = 1, 1$, $p_j = 2, 2, 4, 4, 8, 8, 12, 16$, $w_j = 2, 2, 2, 2, 1, 1, 1, 1$ for $i = 1, 2$ and $j = 1, 2, \ldots, 8$. For a schedule $\pi = (8, 4, 3, 5, 7, 0, 6, 2, 1, 0)$, two zeros separate the 8 jobs into

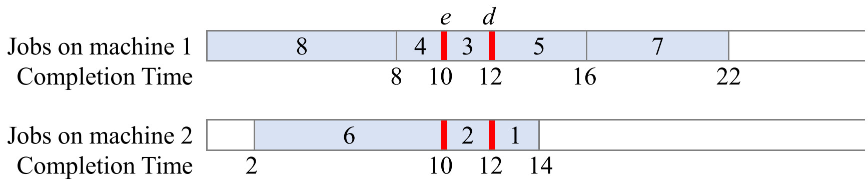

FIGURE 1. A problem instance.

Function *LB1*(α, β, m_α, c_α, t_α, A_α)

INPUT

α : determined job sequence

β : remaining jobs

m_α : current machine

c_α : current objective cost

t_α : current completion time on machine m_α, i.e., $\max_{j\in\alpha}\{C_j(\pi)\}$

A_α : current accumulated earliness and tardiness

OUTPUT

$c_\alpha + c_\beta$: lower bound

Step 1. Sort jobs in β in descending order of w_j / p_j.

Step 2. Set $p_{\max} = \max\{p_j \mid j \in \beta\}$.

Step 3. Construct buffer $B_{m_\alpha} = [t_\alpha, d + A - A_\alpha]$ for processing jobs on machine m_α and buffer $B_k = [e - p_{\max} / v_k, e] \cup [e, d] \cup [d, d + A - A_\alpha]$ for $k = m_\alpha + 1, m_\alpha + 2, ..., m$.

Step 4. Set $c_\beta = 0$.

Step 5. For $k = m_\alpha$ to $m+1$ do Step 6.

Step 6. Repeatedly allocate jobs one by one to buffer B_k in a preemptive way and accumulate the cost c_β until B_k is full.

Step 7. Output $c_\alpha + c_\beta$.

FIGURE 2. The proposed lower bound 1.

three parts. Since the third part is empty, no outsourcing is needed. The completion times of all the jobs are shown in Fig. 1. Therefore, the total tardiness and earliness is 18, i.e., $T_j = 2, 0, 0, 0, 4, 0, 10, 0$, $E_j = 0, 0, 0, 0, 0, 0, 0, 2$, for $j = 1, 2, \ldots, 8$. Note that small jobs with large weights are centralized within the due window $[e, d]$ and the starting time on machine 2 is 2. Moreover, the cost for machine 1 is 26 and the cost for machine 2 is 16. Therefore, the minimum resource consumption is 42.

The following theorem shows that the problem is NP-hard by reducing the 0-1 knapsack problem [80]. The 0-1 knapsack problem is defined as follows. There are n indivisible items and a knapsack. These items are of different values and weighs. We want to take as valuable a load as possible, but the load cannot exceed the maximum capacity of the knapsack.

Theorem 1: The proposed problem is NP-hard.

Proof: At first an instance I of the 0-1 knapsack problem is given. There are n positive values $\gamma_1, \gamma_2, \ldots, \gamma_n \in \mathbb{Z}$, n positive weights $\omega_1, \omega_2, \ldots, \omega_n \in \mathbb{Z}$, and a positive capacity $W \in \mathbb{Z}$. Which items should be taken such that we have maximum $\Sigma_{j \text{ is chosen}} \gamma_j$ with $\Sigma_{j \text{ is chosen}} \omega_j \leq W$?

For any given instance I of the 0-1 knapsack problem, we construct a corresponding instance I' of the corresponding decision version of our scheduling problem as follows:

-Number of machines: $m = 1$
-Speeds of machines: $v_i = 1$ for $i = 1$
-Costs of machines: $c_i = 1$ for $i = 1$
-Limit of total earliness and tardiness: $A = 0$
-Number of jobs: n
-Processing times: $p_j = \omega_j$ for $j = 1, \ldots, n$
-Job weights: $w_j = \gamma_j$ for $j = 1, \ldots, n$
-Due window: $[e, d] = [0, W]$

It is easy to verify that there exists an optimal schedule with the total earliness and tardiness less than or equal to 0 if and only if there exists an optimal solution for the 0-1 knapsack problem with the maximum load less than or equal to W. The proof is complete. ■

Compared with the 0-1 knapsack problem, the proposed problem is more difficult. The 0-1 knapsack problem needs only to determine an optimal combination (or partition) of valuable items, whereas the proposed problem should determine an optimal permutation of jobs. That is, the solution space of the proposed problem is much larger than that of the 0-1 knapsack problem. Moreover, even for a determined schedule, the starting time on each machine is also needed to be determined. Consequently, it is not suitable for solving this problem by only using some commercial programs. Some dedicated properties need to be

Function *LB2*(α , β , m_α , c_α , t_0 , t_α , A_α)
INPUT
α : determined job sequence
β : remaining jobs
m_α : current machine
c_α : current objective value, i.e., earliness and tardiness
t_0 : starting time on machine m_α
t_α : current completion time on machine m_α , i.e., $\max_{j\in\alpha}\{C_j(\pi)\}$
A_α : current accumulated earliness and tardiness
OUTPUT
$c_\alpha + c_\beta$: lower bound

Step 1. Sort jobs in β in descending order of w_j / p_j .
Step 2. Set $v_{\max} = \max\{v_k \mid k = m_\alpha, m_\alpha + 1, ..., m\}$.
Step 3. Set $p_{\max} = \max\{p_j \mid j \in \beta\}$.
Step 4. Construct buffer $B_{m_\alpha} = [t_0 - p_{\max} / v_{m_\alpha}, t_0] \cup [t_\alpha, d + (A - A_\alpha)v_{\max} / v_{m_\alpha}]$ for processing jobs on machine m_α and buffer $B_k = [e - p_{\max} / v_k, e] \cup [e, d]$ for $k = m_\alpha + 1, m_\alpha + 2, ..., m$.
Step 5 Set $c_\beta = 0$.
Step 6. For $k = m_\alpha$ to $m+1$ do Step 7.
Step 7. Repeatedly allocate jobs one by one to buffer B_k in a preemptive way and accumulate the cost c_β until B_k is full.
Step 8. Output $c_\alpha + c_\beta$.

FIGURE 3. **The proposed lower bound 2.**

Algorithm *B&B*(p_j , w_j , v_i , c_i , e , d , A , $\pi_{\min}$)
INPUT
p_j : processing time of job j
w_j : weight of job j
v_i : speed of machine i
c_i : cost of machine i
e : threshold for earliness
d : threshold for tardiness
A : limit of total earliness and tardiness
$\pi_{\min}$: initial solution
OUTPUT
$\pi_{\min}$: optimal schedule

Step 1. (Initialization) Set the objective cost $f(\pi_{\min})$ of the initial schedule. //Note that $f()$ means the objective function.
Step 2. (Branching) Apply Rules 1–9 and Lemmas 1–2 to delete the dominated branches.
Step 3. (Bounding) Compute the two lower bounds, i.e., *LB*1 and *LB*2, for each branch; Delete the branch if $\max\{LB1, LB2\} \geq f(\pi_{\min})$.
Step 4. If a lower objective cost $f(\pi)$ occurs at a leaf node, replace $f(\pi_{\min})$ with the lower cost $f(\pi)$ and set $\pi_{\min} = \pi$.
Step 5. Repeat Steps 2–5 until no branches are available.
Step 6. Output $f(\pi_{\min})$ and $\pi_{\min}$.

FIGURE 4. **The proposed B&B algorithm.**

developed to shrink the searching area in the solution space.

IV. BRANCH-AND-BOUND ALGORITHM

To optimally solve the problem, we propose a branch-and-bound algorithm for small problem instances. First, several dominance rules are proposed for pruning unnecessary branches in a search tree. Second, two complementary lower bounds are developed for trimming some schedules whose partial costs are higher than the lower bounds

A. DOMINANCE RULES

Some symbols used to develop the dominance rules are introduced as follows. Suppose that $\pi = (\alpha, i, j, \beta)$ and $\pi' = (\alpha, j, i, \beta)$ are two schedules of jobs, where α is a determined partial schedule and β is an undetermined

Algorithm $GA(r_c, r_m, r_{ls}, N)$
INPUT
r_c : crossover rate
r_m : mutation rate
r_{ls} : local search rate
N : population size
OUTPUT
$\pi_{\min}$: final schedule
Step 1. (Initialization) N schedules are randomly generated.
Step 2. (Evaluation) For each schedule π_i, its fitness is $\sqrt{f(\pi_i)} / \sum_{k=1}^{N} \sqrt{f(\pi_k)}$. For each violation of π_i, a penalty of 1,000 is accumulated in the objective cost of π_i. If a π_i achieves lower cost, set $\pi_{\min} = \pi_i$. //Note that $f()$ means the objective function.
Step 3. (Selection) In the original population, schedules will be chosen by a standard roulette wheel selection to form a new generation.
Step 4. (Crossover) To form the new population, $r_c N$ selected schedules are obtained by a shift-insert crossover operation and $(1-r_c)N$ selected schedules directly survive into the new generation.
Step 5. (Mutation) For each schedule π_i in the new generation, there is a mutation probability of r_m.
Step 6. (Evaluation) For each schedule π_i in the new generation, its fitness is computed again.
Step 7. (Local Search) In the new population, $r_{ls}N$ schedules are randomly selected and improved by a local search of *SA*.
Step 8. Repeat Steps 2–8 until some stopping criterion is met.
Step 9. Output the final schedule $\pi_{\min}$.

FIGURE 5. The proposed genetic algorithm.

Function $SA(\lambda, T_{ini}, T_0, K, \pi)$
INPUT
λ : cooling coefficient
T_{ini} : initial temperature
T_0 : freezing temperature
K : iteration number
π : initial schedule
OUTPUT
$\pi_{\min}$: final schedule
BEGIN
Step 1. Set the initial temperature, i.e., $T = T_{ini}$.
Step 2. While ($T > T_0$) do Steps 3–8.
Step 3. For $k = 1$ to K do Steps 4–7.
Step 4. Generate a neighbor schedule π' by taking a random walk (i.e., interchange of pairwise jobs) starting from π.
Step 5. Generate a random real number r with $0 \le r \le 1$.
Step 6. Set $\varepsilon = f(\pi) - f(\pi')$. //Note that $f()$ means the objective function.
Step 7. If ($\varepsilon > 0$) or ($r < e^{-\varepsilon/T}$) then $\pi_{\min} = \pi'$.
Step 8. Set $T = \lambda T$.
Step 9. Output the final schedule $\pi_{\min}$.

FIGURE 6. The proposed local search.

partial sequence. The only difference between π and π' is a pairwise interchange of two adjacent jobs i and j on some machine k. For simplicity, let $\max_{j\in\alpha}\{C_j(\pi)\} = t_0$, $C_i(\pi) = t_0 + p_i/v_k = t_1$, $C_j(\pi) = t_1 + p_j/v_k = t_2$, $C_j(\pi') = t_0 + p_j/v_k = t_3$, and $C_i(\pi') = t_3 + p_i/v_k = t_4 = t_2$.

Rules 1 and 2 show that a larger job should be scheduled first if both schedules cause earliness. Since the proofs are similar, only the proof of the first dominance rule is given below.

Rule 1: If $t_1 < t_3 < e \le t_4$, then π' dominates π.

Proof: We observe the difference of earliness for the two schedules. Since $E_j(\pi) = \max\{0, e - t_2\} = \max\{0, e - t_4\} = E_i(\pi')$ and $E_j(\pi') = \max\{0, e - C_j(\pi')\} = e - t_3 < e - t_1 = \max\{0, e - C_i(\pi)\} = E_i(\pi)$. So π' dominates π. The proof is complete. ■

Rule 2: If $t_1 < t_3 < t_4 \le e$, then π' dominates π.

Rules 3–6 deal with the situation that both earliness and tardiness occur on a single machine. If a larger job j is processed first, the total earliness or tardiness can be improved.

Algorithm $ICA(r_c, r_m, r_{ls}, M, N)$

INPUT

r_c : assimilation rate (the crossover rate of GA)

r_m : revolution rate (the mutation rate of GA)

r_{ls} : local search rate

M : number of empires

N : number of citizens (the population size of GA)

OUTPUT

$\pi_{\min}$: best king (final schedule)

Step 1. (Initialization) Randomly generate N citizens (like the chromosomes of GA) and randomly assign them to these empires. Evaluate each citizen and determine a king for each empire.

Step 2. (Assimilation) Select $r_c N$ citizens by the standard roulette wheel selection and force each of them to move towards its corresponding king (like the crossover operation of GA).

Step 3. (Rebellion) Randomly select $r_m N$ citizens and let each of them defect to another empire (like the mutation operation of GA).

Step 4. (Local Search) Randomly select $r_{ls} N$ citizens and apply the local search (i.e., *SA*) to each selected citizen.

Step 5. (Evaluation) Evaluate each citizen and determine a new king for each empire.

Step 6. (Competition) Randomly choose a citizen from the weakest empire (i.e., its king has the largest objective cost) and randomly assign it to another empire.

Step 7. Repeat Steps 2–6 until some stopping criteria are met.

Step 8. Output the best king $\pi_{\min}$.

FIGURE 7. The proposed ICA algorithm.

TABLE 4. Parameter settings.

Parameter	Range	Meaning
n	5,10,15,25,50	number of jobs
m	1,2,3	number of machines
c_i	[1.0,5.0], 10.0	unit cost of machine i
v_i	[1.0,5.0]	processing speed of machine i
p_j	{1,2,…,100}	processing time of job j
w_j	[1.0,2.0]	processing cost coefficient of job j
e	$\{100,101,\ldots,Round(0.1\sum_{j=1}^{n} p_j)\}$	due window
d	$\{e,e+1,\ldots,e+500l\}$	due window
l	[0.25,0.75]	control parameter for due window
a	[0.0,1.0]	control parameter for limit A (=300a)
N	50n	population size of GA (ICA)
r_c	0.8	crossover rate of GA (assimilation rate of ICA)
r_m	0.05	mutation rate of GA (revolution rate of ICA)
r_{ls}	0.05	local search rate of GA and ICA
λ	0.95	cooling coefficient of SA
T_{ini}	100.0	initial temperature of SA
T_0	0.0	freezing temperature of SA
K	100	iteration number of SA
M	3	number of empires of ICA

Rule 3: If $t_1 < e < d \leq t_3$ and $e - t_1 > t_3 - d$, then π' dominates π.

Rule 4: If $t_1 < e \leq t_3 \leq d$, then π' dominates π.

Rule 5: If $t_3 \leq e < d \leq t_1$ and $t_1 - d > e - t_3$, then π' dominates π.

Rule 6: If $t_1 < t_3 \leq d$ and $t_1 < e$, then π' dominates π.

Rules 7–9 show that a smaller job j should be scheduled first if both schedules cause tardiness.

Rule 7: If $e \leq t_3 < t_1$ and $d < t_1$, then π' dominates π.

Rule 8: If $d \leq t_0 < t_3 < t_1$, then π' dominates π.

Rule 9: If $e \leq t_3 \leq d < t_1$, then π' dominates π.

Let $h_k(x) = p_j$ be a function for some allocated jobs on machine k, where job j is scheduled at the xth position. Lemma 1 shows that there exists an optimal schedule such that $h_k(x)$ is always concave upward, where $h_k(x)$ means the process time of the job scheduled at the xth position on machine k. By this lemma, we can prune many unnecessary branches whose processing times are not concave upward on a machine k.

Lemma 1: There exists an optimal sequence for each machine k such that $h_k(x)$ is concave upward.

Proof: Consider there are three situations for two adjacent jobs i and j on machine k. First, they cause earliness. Then by Rules 1 and 2, a larger job should be processed first. Second, no earliness and tardiness occurs. Let the two jobs scheduled with $h_k(x)$ concaves upward. That makes no harm to the objective cost. Third, they cause tardiness. Then by Rules 7–9, a smaller job should be processed first. Therefore, there exists an optimal schedule with $h_k(x)$ concaves upward. The proof is complete. ■

TABLE 5. The performance of B&B and ICA for $n = 5$.

			B&B				GA				ICA			
			Nodes		Run Time		Run Time		REP		Run Time		REP	
m	l	A	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
1	0.25	0	13.90	66	0.00	0.00	0.11	0.13	0.00	0.00	0.13	0.22	0.00	0.00
		100	6.34	66	0.00	0.00	0.11	0.13	0.00	0.00	0.13	0.14	0.00	0.00
		200	0.00	0	0.00	0.00	0.11	0.13	0.00	0.00	0.13	0.16	0.00	0.00
		300	1.74	87	0.00	0.00	0.11	0.13	0.00	0.00	0.13	0.14	0.00	0.00
	0.50	0	10.00	66	0.00	0.00	0.11	0.13	0.00	0.00	0.13	0.14	0.00	0.00
		100	1.32	66	0.00	0.00	0.11	0.13	0.00	0.00	0.13	0.14	0.00	0.00
		200	0.00	0	0.00	0.00	0.11	0.13	0.00	0.00	0.13	0.14	0.00	0.00
		300	2.98	87	0.00	0.00	0.11	0.13	0.00	0.00	0.13	0.14	0.00	0.00
	0.75	0	6.22	66	0.00	0.00	0.11	0.13	0.00	0.00	0.13	0.14	0.00	0.00
		100	1.46	41	0.00	0.00	0.11	0.13	0.00	0.00	0.13	0.14	0.00	0.00
		200	1.52	76	0.00	0.00	0.11	0.13	0.00	0.00	0.13	0.17	0.00	0.00
		300	0.00	0	0.00	0.00	0.11	0.13	0.00	0.00	0.13	0.14	0.00	0.00
2	0.25	0	17.04	66	0.00	0.02	0.12	0.14	0.00	0.00	0.14	0.16	0.00	0.00
		100	0.00	0	0.00	0.00	0.12	0.13	0.00	0.00	0.14	0.17	0.00	0.00
		200	0.00	0	0.00	0.00	0.12	0.14	0.00	0.00	0.14	0.16	0.00	0.00
		300	2.12	106	0.00	0.00	0.12	0.14	0.00	0.00	0.15	0.16	0.00	0.00
	0.50	0	8.78	98	0.00	0.00	0.12	0.14	0.00	0.00	0.14	0.17	0.00	0.00
		100	0.00	0	0.00	0.00	0.12	0.14	0.00	0.00	0.14	0.16	0.00	0.00
		200	0.00	0	0.00	0.00	0.12	0.14	0.00	0.00	0.14	0.16	0.00	0.00
		300	0.00	0	0.00	0.00	0.13	0.14	0.00	0.00	0.14	0.16	0.00	0.00
	0.75	0	5.84	66	0.00	0.00	0.12	0.13	0.00	0.00	0.14	0.16	0.00	0.00
		100	1.04	52	0.00	0.00	0.12	0.13	0.00	0.00	0.14	0.16	0.00	0.00
		200	0.00	0	0.00	0.00	0.12	0.13	0.00	0.00	0.14	0.16	0.00	0.00
		300	0.00	0	0.00	0.00	0.12	0.13	0.00	0.00	0.14	0.16	0.00	0.00
3	0.25	0	12.24	66	0.00	0.02	0.13	0.14	0.00	0.00	0.16	0.17	0.00	0.00
		100	0.72	36	0.00	0.00	0.14	0.16	0.00	0.00	0.16	0.17	0.00	0.00
		200	0.00	0	0.00	0.00	0.14	0.14	0.00	0.00	0.16	0.16	0.00	0.00
		300	0.00	0	0.00	0.00	0.14	0.16	0.00	0.00	0.16	0.17	0.00	0.00
	0.50	0	6.22	60	0.00	0.02	0.13	0.16	0.00	0.00	0.16	0.17	0.00	0.00
		100	0.00	0	0.00	0.00	0.13	0.14	0.00	0.00	0.16	0.16	0.00	0.00
		200	0.00	0	0.00	0.00	0.14	0.16	0.00	0.00	0.16	0.17	0.00	0.00
		300	0.00	0	0.00	0.00	0.14	0.16	0.00	0.00	0.16	0.16	0.00	0.00
	0.75	0	6.26	60	0.00	0.00	0.13	0.14	0.00	0.00	0.16	0.17	0.00	0.00
		100	1.14	57	0.00	0.00	0.14	0.14	0.00	0.00	0.16	0.16	0.00	0.00
		200	0.00	0	0.00	0.00	0.14	0.16	0.00	0.00	0.16	0.17	0.00	0.00
		300	0.00	0	0.00	0.00	0.14	0.16	0.00	0.00	0.16	0.17	0.00	0.00

The concept of outsiders is described as follows. Let n_L be the number of left-hand jobs whose completion times less than e and n_R be the number of right-hand jobs whose completion times greater than d. Then Lemma 2 shows that there exists an optimal schedule which has the same number of left-hand outsiders and right-hand outsiders, i.e., $n_L = n_R$, for each machine k. By Lemma 2, we can determine the starting time of jobs allocated to a machine.

Lemma 2: Given jobs allocated to a machine (machine k), for each optimal sequence on the machine, $n_L = n_R$.

Proof: We prove it by contradiction. Assume there are no such optimal sequence with $n_L = n_R$ for machine k. Without loss of generality, suppose α is an optimal sequence with $n_L < n_R$. Let δ be the minimum nonzero earliness E_j and tardiness T_j for all jobs allocated to machine k. Then we force all jobs to be processed 0.5δ later than their original starting times. Namely, all their original completion times C_j becomes $C_j + 0.5\delta$. Then the objective gain becomes a negative value, i.e., $0.5(n_L\delta - n_R\delta)$. This contradicts that α is an optimal sequence on the machine with $n_L < n_R$. The proof is complete. ■

B. LOWER BOUND

To accelerate the B&B algorithm, we propose two complementary lower bounds. Again, let $\pi = (\alpha, \beta)$ be an incomplete schedule, where α is a determined partial schedule and β is an undetermined partial sequence. When computing the lower bounds, we assume that jobs in β are preemptive.

Fig. 2 shows the details of the first lower bound. Assume that c_α is the current cost, m_α is the current machine, t_α is the current makespan on machine m_α, and A_α is the current accumulated earliness and tardiness. We sort jobs in β in descending order of w_j/p_j and determine the largest job in β (Steps 1 and 2). Since there is a quota of $(A - A_\alpha)$ for earliness and tardiness, we allow for overtime work on machine m_k for $k = m_\alpha + 1, m_\alpha + 2, \ldots, m$ (Step 3). Moreover, we take advantage of the period of processing time $[e - p_{\max}/v_k, e]$ before the due window. We assume that jobs processed during the period cause no earliness and tardiness. Then, we can allocate valuable jobs (i.e., larger w_j/p_j) one by one to economic machines (i.e., smaller c_k/v_k) in a preemptive way (Steps 4–6). Finally, the lower bound is returned in Step 7.

Fig. 3 shows the details of the second lower bound. We assume that $\max_{j \in \alpha}\{C_j(\pi)\} = t_0$. Note that the most

TABLE 6. The performance of B&B and ICA for $n = 10$.

			B&B				GA				ICA			
			Nodes		Run Time		Run Time		REP		Run Time		REP	
m	l	A	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
1	0.25	0	27462.98	334274	0.29	3.20	0.49	0.61	0.00	0.00	0.64	0.77	0.00	0.00
		100	50919.48	362838	0.60	3.81	0.51	0.70	0.21	4.25	0.70	0.84	0.00	0.00
		200	22590.28	210535	0.27	2.31	0.53	0.67	0.00	0.00	0.71	0.84	0.00	0.00
		300	10024.52	111666	0.12	1.31	0.56	0.91	0.09	3.06	0.73	0.80	0.06	3.06
	0.50	0	69949.14	749204	0.68	6.82	0.51	0.61	0.00	0.00	0.67	0.75	0.00	0.00
		100	37647.28	401229	0.42	4.10	0.54	0.73	0.00	0.00	0.71	0.78	0.00	0.00
		200	47415.28	512614	0.53	5.43	0.55	0.62	0.00	0.00	0.74	0.92	0.00	0.00
		300	9446.94	150078	0.12	1.90	0.57	0.84	0.01	0.63	0.76	0.81	0.00	0.00
	0.75	0	53923.90	749204	0.53	6.99	0.53	0.61	0.00	0.00	0.70	1.05	0.00	0.00
		100	41732.78	675044	0.44	6.60	0.54	0.59	0.04	2.01	0.73	1.23	0.00	0.00
		200	10758.52	210405	0.13	2.45	0.55	0.59	0.00	0.00	0.73	0.80	0.00	0.00
		300	7639.36	163921	0.09	1.92	0.56	0.61	0.10	4.96	0.73	0.76	0.10	4.96
2	0.25	0	35670.54	749204	0.37	6.93	0.50	0.53	0.10	3.80	0.68	1.05	0.19	9.07
		100	19903.50	396502	0.24	4.21	0.53	0.84	0.07	1.18	0.74	0.94	0.02	1.18
		200	19234.64	258781	0.22	2.76	0.53	0.56	0.14	5.99	0.75	0.84	0.00	0.00
		300	8246.08	126560	0.10	1.42	0.53	0.58	0.00	0.00	0.75	0.80	0.00	0.00
	0.50	0	42056.70	708884	0.43	6.74	0.51	0.62	0.01	0.36	0.71	1.09	0.00	0.00
		100	16394.24	172604	0.20	2.11	0.53	0.58	0.04	0.91	0.75	1.01	0.00	0.00
		200	599.62	29981	0.01	0.37	0.54	0.58	0.04	1.83	0.75	1.05	0.00	0.00
		300	8651.54	214487	0.10	2.37	0.53	0.58	0.02	0.72	0.74	0.83	0.01	0.36
	0.75	0	79905.72	749204	0.78	7.02	0.53	0.86	0.00	0.00	0.71	0.78	0.00	0.00
		100	62027.34	563766	0.68	6.55	0.55	1.11	0.91	45.08	0.73	0.81	0.00	0.00
		200	13480.64	229296	0.16	2.59	0.54	0.58	0.02	0.63	0.74	0.89	0.01	0.28
		300	922.90	46145	0.01	0.56	0.55	0.59	0.01	0.35	0.74	0.80	0.01	0.35
3	0.25	0	75503.20	749204	0.76	6.82	0.53	0.89	0.14	3.98	0.70	0.86	0.00	0.00
		100	22794.04	295176	0.27	3.34	0.54	1.00	0.12	2.77	0.75	1.05	0.03	0.85
		200	11262.78	176609	0.13	1.93	0.54	0.58	0.04	0.87	0.77	1.26	0.00	0.00
		300	12490.10	222107	0.14	2.42	0.54	0.58	0.03	0.89	0.76	0.80	0.01	0.25
	0.50	0	27381.74	554684	0.29	5.43	0.53	0.56	0.00	0.11	0.71	0.78	0.00	0.00
		100	34534.60	544635	0.39	5.43	0.54	0.56	0.08	1.55	0.74	0.80	0.03	0.88
		200	4462.28	199040	0.05	2.25	0.54	0.84	0.07	3.32	0.77	1.23	0.00	0.00
		300	0.00	0	0.00	0.00	0.55	0.58	0.01	0.46	0.77	1.20	0.00	0.00
	0.75	0	12810.10	331176	0.14	3.25	0.53	0.58	0.00	0.00	0.74	1.31	0.00	0.00
		100	22170.98	306731	0.26	3.39	0.55	0.61	0.01	0.59	0.75	0.84	0.01	0.45
		200	7266.62	153499	0.09	1.90	0.56	0.59	0.05	2.37	0.78	1.11	0.00	0.00
		300	3992.46	105952	0.05	1.36	0.57	0.61	0.01	0.51	0.78	0.87	0.00	0.00

economic machine is not necessarily the fast machine. Consequently, we increase the processing speed of machine m_α and assume that all the remaining quota, i.e., $(A - A_\alpha)$ is consumed only by this machine. Therefore, no earliness and tardiness occur on the other machines and the time buffers of other machines are narrower than those of lower bound 1.C. Branch-and-bound algorithm To optimally solve the problem, we employ a depth-first search to develop a B&B algorithm which includes the following basic six steps (Fig. 4). First, we borrow a near-optimal solution from a metaheuristic algorithm, e.g., GA or ICA, (see next subsection) and build a search tree. In Steps 2 and 3, dominance rules and lower bounds are employed to eliminate some branches that the optimal solutions do not reside in. Step 3 updates the current global minimum if a new local minimum occurs. Note that $f()$ means the objective function. Repeat the depth-first search and obtain the global minimum.

V. METAHEURISTIC ALGORITHMS

When the problem size is large, we propose an imperialist competitive algorithm (ICA) to provide near-optimal solutions. Moreover, we also employ a genetic algorithm (GA) as a benchmark for evaluating the performance of ICA.

A. GENETIC ALGORITHM (GA)

A simple genetic algorithm (GA) is used as a benchmark and the details are shown in Fig. 5. In Step 1, we randomly generate N schedules, where N is the population size. For example, we have five jobs ($n = 5$) and two machines ($m = 2$). A schedule (1,5,3,0,2,0,4) means that jobs 1, 5, 3 are allocated to machine 1, job 2 is allocated to machine 2, and job 4 is outsourced, where the zeros mean separators. Note that some schedules may be invalid. In Step 2, we compute the fitness for each schedule for later selection and crossover. In Steps 3 and 4, a roulette wheel selection is employed for selection. We select $r_c N$ solutions to form the next generation by crossover and $(1 - r_c)N$ solutions to directly survive into the next generation. For each pair of selected solutions, a shift-and-insert crossover is conducted. Consider a solution as a circular queue. We randomly choose two integers a and b from $\{0,1,\ldots, n\text{-}1\}$. If $a < b$, we first take the job at position a away,

TABLE 7. The performance of B&B and ICA for $n = 15$.

			B&B					GA					ICA				
			Nodes		Run Time			Run Time		REP			Run Time		REP		
m	l	A	Mean	Max	Mean	Max	NA	Mean	Max	Mean	Max	NA	Mean	Max	Mean	Max	NA
1	0.25	0	8687466.87	79520474	230.53	2056.81	12	1.41	2.34	1.22	21.61	12	2.12	4.18	0.24	5.32	12
		100	10443159.55	62697916	261.76	1533.04	12	1.45	2.45	1.17	10.10	12	2.26	4.35	0.30	3.49	12
		200	7149356.89	93297077	173.05	2217.57	13	1.61	4.67	2.07	21.47	13	2.41	3.99	0.56	7.88	13
		300	5842394.07	56441024	152.00	1470.28	7	1.50	2.15	0.35	3.82	7	2.07	2.93	0.23	3.82	7
	0.50	0	3721244.00	94118209	94.69	2396.85	19	1.58	2.54	0.27	3.49	19	2.34	4.23	0.12	3.02	19
		100	7134050.67	67611043	168.25	1591.37	11	1.52	2.37	0.66	13.65	11	2.30	3.78	0.20	4.00	11
		200	4967775.60	81131885	125.64	2029.71	8	1.50	2.03	0.15	2.41	8	2.18	3.10	0.12	4.09	8
		300	2424016.42	61632046	58.90	1468.48	5	1.55	1.93	0.17	2.36	5	2.16	2.65	0.06	1.40	5
	0.75	0	1770360.78	63203313	39.90	1423.92	14	1.59	2.39	0.13	3.57	14	2.26	2.87	0.00	0.00	14
		100	13243166.85	95078198	331.92	2350.31	16	1.55	4.87	1.55	11.85	16	2.24	3.71	0.65	5.48	16
		200	4621671.82	52232796	109.85	1335.06	23	1.47	2.22	1.05	7.07	23	2.36	3.84	0.23	2.23	23
		300	8437599.57	82097615	193.87	1857.91	22	1.53	2.70	1.55	23.22	22	2.47	5.26	0.12	1.95	22
2	0.25	0	5445114.21	97772554	136.30	2366.86	11	1.45	2.53	0.65	5.71	11	2.25	3.54	0.25	4.68	11
		100	226131.00	4628874	5.79	118.20	14	1.48	2.09	0.20	5.54	14	2.30	3.85	0.00	0.15	14
		200	4359229.90	84590522	100.95	1852.26	9	1.49	2.48	0.27	5.66	9	2.43	4.45	0.06	1.15	9
		300	4105277.75	63204366	107.32	1712.16	14	1.55	4.82	0.46	9.30	14	2.33	3.81	0.08	2.01	14
	0.50	0	1445959.95	21816858	35.66	495.76	8	1.58	4.76	0.98	29.58	8	2.38	3.98	0.03	1.09	8
		100	418585.38	18836342	11.24	505.79	5	1.52	2.50	0.07	3.00	5	2.37	4.32	0.01	0.48	5
		200	15658706.71	84025270	388.00	2032.27	16	1.59	4.01	2.85	59.99	16	2.48	3.78	0.23	5.03	16
		300	11828472.41	99206146	283.13	2361.65	21	1.65	3.34	0.79	6.13	21	2.43	4.31	0.07	0.82	21
	0.75	0	4250486.49	52921698	97.04	1164.35	11	1.55	2.45	0.68	6.53	11	2.52	4.34	0.03	0.54	11
		100	11939213.51	82956616	300.02	2036.69	13	1.54	3.00	2.60	36.70	13	2.33	3.45	0.13	2.22	13
		200	5374951.50	85961952	130.63	1970.43	12	1.52	2.87	0.49	9.58	12	2.35	3.78	0.01	0.32	12
		300	1751259.66	27535073	42.37	661.21	3	1.47	2.12	0.11	2.65	3	2.33	4.32	0.00	0.11	3
3	0.25	0	2609580.00	55968313	64.57	1364.39	9	1.50	2.64	0.51	10.17	9	2.31	3.26	0.17	3.69	9
		100	4231731.59	70788897	102.96	1747.48	9	1.51	2.68	0.12	2.56	9	2.44	5.57	0.00	0.00	9
		200	417201.95	17939684	10.01	430.39	7	1.48	2.04	0.09	2.37	7	2.28	3.85	0.00	0.00	7
		300	8687466.87	79520474	230.53	2056.81	12	1.41	2.34	1.22	21.61	12	2.12	4.18	0.24	5.32	12
	0.50	0	10443159.55	62697916	261.76	1533.04	12	1.45	2.45	1.17	10.10	12	2.26	4.35	0.30	3.49	12
		100	7149356.89	93297077	173.05	2217.57	13	1.61	4.67	2.07	21.47	13	2.41	3.99	0.56	7.88	13
		200	5842394.07	56441024	152.00	1470.28	7	1.50	2.15	0.35	3.82	7	2.07	2.93	0.23	3.82	7
		300	3721244.00	94118209	94.69	2396.85	19	1.58	2.54	0.27	3.49	19	2.34	4.23	0.12	3.02	19
	0.75	0	7134050.67	67611043	168.25	1591.37	11	1.52	2.37	0.66	13.65	11	2.30	3.78	0.20	4.00	11
		100	4967775.60	81131885	125.64	2029.71	8	1.50	2.03	0.15	2.41	8	2.18	3.10	0.12	4.09	8
		200	2424016.42	61632046	58.90	1468.48	5	1.55	1.93	0.17	2.36	5	2.16	2.65	0.06	1.40	5
		300	1770360.78	63203313	39.90	1423.92	14	1.59	2.39	0.13	3.57	14	2.26	2.87	0.00	0.00	14

then shift jobs one by one from positions a+1, a+2,..., b to positions a, a+1,..., b-1, and finally insert the previously taken job into position b. Similarly, $a > b$, we first take the job at position a away, then shift jobs one by one from positions a+1, a+2,..., n-1,0,1,..., b to positions a, a+1,..., n-1,0,1,..., b-1, and finally insert the previously taken job into position b. In Step 5, there is a mutation probability of r_m that a solution mutates in the new generation. Moreover, we perform a local search to enhance the solution quality of GA by a simple algorithm of simulated annealing (SA) in Step 7. In the last step, the algorithm terminates if the run time reaches $2n$ seconds or no improvements are made during the recent $20n$ generations.

A simulated annealing (SA) algorithm is used as a local search (Fig. 6). SA is motivated by annealing in metals. The phenomenon is a gradually stabilized process which occurs when heated or melted metal cools down slowly. High temperature means a random state and it is relatively easy for molecules to reach a new thermal equilibrium. Therefore, in Step 6, SA jumps to a new position mainly due to discovery of a local minimum and slightly due to forced change (like mutation of GA). Moreover, the completion time of cooling process is controlled by the cooling coefficient λ in Step 8. After searching the local area of π, a new schedule π with lower objective cost is returned.

B. IMPERIALIST COMPETITIVE ALGORITHM (ICA)

The imperialist competitive algorithm (ICA) was first introduced by [81], [82]. ICA is based on the competitions of empires. There are M empires and all of them have N citizens. Each citizen can be viewed as a chromosome in GA. The king of an empire is defined as a citizen who achieves the minimum cost in the corresponding empire. That is, a king is a local minimum in the solution space. In general, we can regard each empire in ICA as a population in GA. The main difference is that ICA has a rebellion operation and achieves more diversification force [83].

The details of ICA are shown in Fig. 7. Like GA, each citizen is encoded as a random permutation of 1,2,..., n, and (m-1) zeros, where a zero means a separator (Step 1). For each violation in the encoding of a citizen, a penalty of 1,000 is accumulated in its objective cost. In Step 2, each citizen is forced to move towards its king. The assimilation operation is done by a shift-and-insert crossover like GA. The main idea

TABLE 8. The comparison between the two lower bounds for $n = 10$.

			B&B				$\text{B\&B}_{\text{LB1}}$				$\text{B\&B}_{\text{LB2}}$			
			Nodes		Run Time		Nodes		Run Time		Nodes		Run Time	
m	l	A	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
1	0.25	0	27462.98	334274	0.31	3.51	57204.54	614132	0.61	5.85	27462.98	334274	0.30	3.37
		100	50919.48	362838	0.65	4.12	74494.74	465868	0.87	4.85	50919.48	362838	0.62	3.95
		200	22590.28	210535	0.29	2.56	23760.88	212818	0.29	2.51	22590.28	210535	0.28	2.48
		300	10023.62	111666	0.13	1.42	10114.18	111834	0.13	1.37	10023.62	111666	0.13	1.37
	0.50	0	69949.14	749204	0.73	7.30	107092.66	749204	1.09	7.05	69949.14	749204	0.71	7.05
		100	37646.86	401229	0.45	4.43	47418.90	484994	0.53	5.06	37646.86	401229	0.43	4.28
		200	47415.28	512614	0.54	5.49	48605.58	513214	0.53	5.32	47415.28	512614	0.52	5.32
		300	9447.62	150078	0.12	1.81	9514.76	151502	0.12	1.76	9447.62	150078	0.12	1.76
	0.75	0	53923.90	749204	0.55	7.29	76152.92	749204	0.78	7.07	53923.90	749204	0.54	7.05
		100	41732.78	675044	0.46	6.90	49485.96	690044	0.53	6.79	41732.78	675044	0.45	6.66
		200	10758.02	210405	0.14	2.59	11273.32	211311	0.14	2.53	10758.02	210405	0.13	2.50
		300	7638.86	163921	0.10	2.03	7680.38	164011	0.10	1.97	7638.86	163921	0.10	1.98
2	0.25	0	35681.20	749204	0.40	7.49	55075.72	749204	0.62	7.32	35681.20	749204	0.39	7.30
		100	19896.26	396502	0.26	4.59	29077.68	533160	0.35	5.60	19908.08	396502	0.25	4.43
		200	19234.64	258781	0.24	3.00	20041.18	272127	0.24	3.06	19771.52	258781	0.24	2.93
		300	8246.08	126560	0.11	1.54	8415.28	127848	0.11	1.50	8254.24	126560	0.10	1.48
	0.50	0	42056.70	708884	0.45	6.99	70464.90	708884	0.76	6.74	42056.70	708884	0.44	6.77
		100	16403.98	172604	0.21	2.17	22708.78	250599	0.27	2.87	16445.74	172604	0.20	2.09
		200	599.62	29981	0.01	0.44	599.98	29999	0.01	0.41	724.98	36249	0.01	0.48
		300	8651.28	214487	0.11	2.50	8736.86	215692	0.11	2.43	8665.56	214487	0.11	2.43
	0.75	0	79905.72	749204	0.83	7.33	110566.96	749204	1.10	7.08	79905.72	749204	0.80	7.10
		100	62027.34	563766	0.73	7.21	72875.90	689849	0.81	8.08	62922.04	563766	0.72	7.04
		200	13479.72	229296	0.17	2.78	14032.28	232074	0.17	2.73	13667.38	238085	0.17	2.78
		300	921.36	46068	0.01	0.61	954.76	47738	0.01	0.62	921.36	46068	0.01	0.59
3	0.25	0	75508.10	749204	0.81	7.29	124367.28	749204	1.28	7.10	75508.10	749204	0.78	7.05
		100	22845.56	295176	0.29	3.45	29574.88	353349	0.35	3.85	23867.30	298038	0.29	3.35
		200	11263.44	176609	0.14	2.08	11689.20	180537	0.14	2.07	11690.76	180604	0.14	2.08
		300	12480.28	222107	0.15	2.56	12535.92	222286	0.15	2.48	12810.72	225086	0.15	2.53
	0.50	0	27384.02	554684	0.31	5.65	71562.96	749204	0.75	7.08	27384.02	554684	0.30	5.46
		100	34474.50	544635	0.42	5.77	44161.86	566908	0.51	5.77	34732.00	544635	0.41	5.62
		200	4462.28	199040	0.06	2.42	4690.92	205755	0.06	2.51	4496.64	200758	0.06	2.39
		300	0.00	0	0.00	0.00	0.00	0	0.00	0.00	0.00	0	0.00	0.00
	0.75	0	12811.08	331176	0.15	3.54	35693.94	561284	0.39	5.57	12811.08	331176	0.14	3.42
		100	22179.38	306731	0.28	3.64	28503.94	348041	0.34	3.92	22711.84	306731	0.28	3.54
		200	7266.62	153499	0.10	1.98	7812.76	160001	0.10	1.98	7509.20	157538	0.10	1.97
		300	3992.46	105952	0.051	1.404	4020.88	105952	0.05	1.34	4219.00	117279	0.05	1.50

behind assimilation is similar to the crossover operation in GA or the movement in particle swarm optimization (PSO). In Steps 3 and 4, a citizen may rebel and defect to another empire. Like GA, we also employ a local search of SA to enhance the solution quality. All the N citizens are evaluated and the weakest empire is punished (Steps 5 and 6). Like GA, ICA terminates if the run time reaches $2n$ seconds or no improvements are made during the recent $20n$ generations.

VI. EXPERIMENTAL RESULTS

The experiments are divided into three parts. In the first part, to evaluate the performance of B&B, we solve the problem optimally when the problem size is small. The related statistics are recorded, e.g., average node and run time. In the second part, ICA is used to obtain near-optimal solutions when the problem size is large. To evaluate the performance of ICA, the performance of GA is viewed as a benchmark. In the third part, two sensitivity tests are conducted to observe the influence of control parameters on the objective cost.

Table 4 summarizes the parameters used in the experiments. Parameters n, m, c_i, v_i, w_j, p_j, e, d, and A have already been defined in Section 3. Parameters c_i and v_i are two real numbers randomly chosen from $[1.0, 5.0]$, w_j is a real number randomly chosen from $[1.0, 2.0]$. Assume that the unit cost of outsourcing, i.e., c_{m+1}, is 10.0. Processing time p_j follows a discrete uniform distribution over $\{1, 2, \ldots, 100\}$. Parameters l and a are used to control the width of due window and limit A respectively. For GA and ICA, both population sizes are N and their crossover rates, mutation rates, and local search rates are identical. Moreover, both of them employ the same local search SA with the same settings. All the algorithms are implemented in Pascal and executed on an Intel Core i7 @ 3.40GHz with 8 GB RAM in a Windows 7 SP1 environment. For each setting, 50 random trials are conducted and recorded. For the two metaheuristic algorithms, the stopping criteria are the run time is over $2n$ seconds or solution quality cannot be improved more during recent $20n$ generations.

In the first part, Table 5 shows the solution quality and run time of the proposed algorithms for $n = 5$. The relative error percentage (REP) is defined as $(f - f_{BB})/f_{BB} \times 100\%$, where f means the objective cost obtained by GA or ICA and f_{BB} is the optimal objective cost obtained by B&B. All the execution speeds of the proposed algorithms are measured in second.

TABLE 9. The performance of ICA for $n = 25$.

			GA				ICA			
			Run Time		RDP		Run Time		RDP	
m	l	A	Mean	Max	Mean	Max	Mean	Max	Mean	Max
1	0.25	100	4.37	6.57	0.62	4.46	6.77	20.08	0.04	1.11
		200	4.70	9.69	1.49	9.14	8.13	18.42	0.02	0.82
		300	5.20	15.97	2.57	16.01	9.06	21.09	0.02	0.69
	0.50	100	4.90	10.08	1.30	7.87	7.77	14.70	0.02	1.10
		200	5.27	10.17	1.55	6.54	8.85	16.82	0.04	1.11
		300	4.92	9.52	2.65	29.02	8.85	19.81	0.00	0.00
	0.75	100	4.93	10.61	1.41	9.20	8.32	14.26	0.00	0.04
		200	5.63	11.23	1.66	15.03	8.80	14.68	0.01	0.72
		300	5.42	8.42	2.72	16.45	9.78	21.11	0.00	0.00
2	0.25	100	5.86	14.74	4.32	15.09	9.68	20.14	0.03	1.02
		200	6.20	19.94	5.08	27.21	9.87	28.39	0.00	0.00
		300	6.64	21.30	5.46	37.53	9.54	20.02	0.01	0.53
	0.50	100	6.06	15.82	2.18	12.23	9.63	17.11	0.04	1.78
		200	6.19	18.92	2.24	22.80	9.96	22.54	0.00	0.06
		300	6.66	22.20	4.62	48.38	10.53	16.38	0.00	0.00
	0.75	100	5.87	16.77	1.84	10.40	9.60	17.16	0.03	0.95
		200	7.06	26.36	2.28	14.36	9.96	20.62	0.00	0.00
		300	5.80	25.96	2.21	20.64	10.19	18.81	0.00	0.00
3	0.25	100	6.30	16.38	4.18	28.95	9.24	17.13	0.06	2.73
		200	6.35	17.44	7.17	50.37	10.48	21.56	0.00	0.00
		300	6.48	19.72	7.47	65.08	10.53	21.20	0.00	0.00
	0.50	100	6.11	15.34	2.40	10.15	9.61	19.03	0.00	0.00
		200	6.89	20.11	4.10	37.85	9.49	20.36	0.00	0.00
		300	6.07	15.20	3.37	53.37	10.85	21.45	0.00	0.00
	0.75	100	6.02	14.98	2.09	18.49	10.15	22.18	0.00	0.00
		200	6.01	16.44	1.99	22.98	9.92	19.81	0.00	0.00
		300	5.97	15.93	2.06	24.40	10.69	21.50	0.00	0.00

It is seen that the average nodes of B&B decreases when the number of machines increases. Multi-machine scheduling seems easier to solve for B&B for some fixed n. B&B also takes more run time to solve an instance with an early due window (i.e., small l). This is because we have enough room for containing small jobs and need more trial and errors to obtain the optimal schedules. On the other hand, when the limit A is small, it also means we have less flexibility to adjusting these small jobs and therefore B&B will consume more execution time to find the optimal schedule. For both metaheuristic algorithms, their solution quality are the same for $n = 5$. However, ICA needs more run time to deal with the competition between the empires.

Table 6 shows the solution quality and run time of the proposed algorithms for $n = 10$. Again, the effects of m, l, A on B&B are similar. However, the difficulty increases when m increases. Multi-machine scheduling is still harder than single-machine scheduling. The most difficult setting is $m = 2$, $l = 0.75$, $A = 0$, since the average node is 79905. On the other hand, as n increases, ICA achieve lower REPs than GA in general though it takes a little more execution time. The design of multiple empires now are effective.

Table 7 shows the solution quality and run time of the proposed algorithms for $n = 15$. The symbol NA means the optimal solutions are not available. This is because we force B&B to quit if the nodes exceeds one hundred million. It implies that some metaheuristic algorithms are needed for providing near-optimal solutions for $n > 15$. On the other hand, the REPs of ICA are relatively low compared with GA. Therefore, ICA can be a good candidate for solving large problem instances. Note that B&B always generates the optimal solutions but consumes more run time. On the other hand, although ICA takes a little longer run time than GA, ICA always provides higher solution quality (i.e., lower REP). Namely, ICA's wider biodiversity is helpful to improve solution quality in this problem.

Table 8 shows the performance of the two proposed lower bounds for $n = 10$. In this table, B&B cooperates closely with two lower bounds, $B\&B_{LB1}$ means a branch-and-bound algorithm works with dominance rules and LB1 only, and $B\&B_{LB2}$ means a branch-and-bound algorithm is equipped with dominance rules and LB2 only. As shown in the mean node columns, LB2 is very powerful, especially for dealing with the situations of early due windows and small limits, e.g., $m = 1$, $L = 0.25$, $A = 0$. On the other hand, LB1 is good at pruning unnecessary nodes for more machines and larger limits, e.g., $m = 2$, $L = 0.5$, $A = 200$. However, without the help of LB2, $B\&B_{LB1}$ will take more runt time than B&B and $B\&B_{LB2}$. It is clear that both lower bounds complement each other.

As shown in the above experiments, B&B can optimally solve the problem within 5 minutes for small problem instances. On the other hand, ICA can provide high-quality solutions within $2n$ seconds for a large n. It is qualified for solving the real-world instance in Table 2.

In the second part, Tables 9 and 10 shows the performances of two metaheuristic algorithms for large problem instances. The relative deviation percentage (RDP) is defined as $(f - f_{\min})/f_{\min} \times 100\%$, where f means the objective cost obtained by GA or ICA and $f_{\min}$ is the minimum cost obtained

TABLE 10. The performance of ICA for $n = 50$.

			GA				ICA			
			Run Time		RDP		Run Time		RDP	
m	l	A	Mean	Max	Mean	Max	Mean	Max	Mean	Max
1	0.25	100	26.24	79.08	0.56	2.04	33.95	63.79	0.01	0.22
		200	29.06	59.20	0.75	1.78	39.13	87.17	0.01	0.31
		300	26.03	74.04	1.21	4.56	39.06	77.02	0.00	0.14
	0.50	100	24.24	54.41	0.99	5.72	34.80	64.82	0.01	0.44
		200	25.91	62.70	1.31	5.25	38.12	85.57	0.00	0.09
		300	29.15	67.91	1.83	5.98	45.33	89.65	0.01	0.51
	0.75	100	28.33	100.03	1.27	3.98	40.43	99.37	0.01	0.12
		200	27.00	57.85	2.28	8.19	49.80	100.04	0.00	0.04
		300	27.64	54.71	3.23	11.05	52.81	100.03	0.00	0.00
2	0.25	100	34.04	93.84	1.79	5.79	41.26	83.71	0.00	0.01
		200	34.26	89.37	2.58	10.69	43.55	100.01	0.00	0.09
		300	34.83	91.79	3.06	11.47	48.15	100.01	0.01	0.39
	0.50	100	38.17	100.03	2.29	10.73	45.68	100.06	0.00	0.22
		200	46.75	100.04	4.22	18.64	56.13	100.06	0.00	0.00
		300	47.88	100.03	5.14	31.75	52.10	100.06	0.01	0.68
	0.75	100	42.87	100.04	3.20	11.00	59.98	100.06	0.01	0.49
		200	51.91	100.04	5.06	17.53	62.04	100.08	0.00	0.00
		300	49.95	100.04	4.60	16.82	64.31	100.06	0.00	0.10
3	0.25	100	42.87	100.04	3.10	14.53	45.75	100.04	0.01	0.24
		200	45.07	100.04	4.38	15.47	44.28	88.39	0.00	0.00
		300	50.48	100.04	5.98	23.55	56.69	100.04	0.00	0.00
	0.50	100	49.19	100.04	4.45	17.22	58.31	100.06	0.00	0.00
		200	44.99	100.03	6.05	25.07	60.82	100.06	0.00	0.00
		300	53.40	100.04	7.27	34.33	62.44	100.08	0.00	0.00
	0.75	100	46.33	100.01	3.78	18.63	58.75	100.04	0.00	0.00
		200	49.58	100.04	4.53	21.92	57.03	100.07	0.00	0.00
		300	40.29	100.04	5.07	29.50	56.92	100.06	0.00	0.00

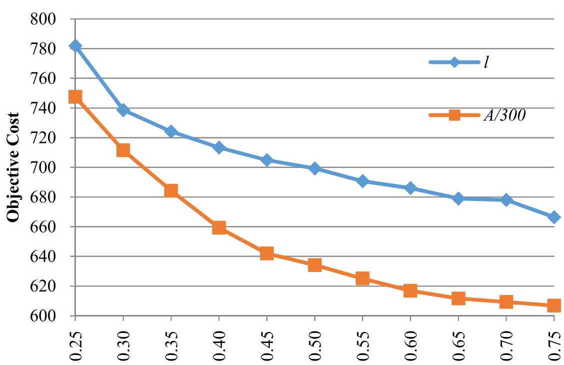

FIGURE 8. The effects of control parameters on the objective cost.

TABLE 11. The effects of input parameters on the objective cost.

	processing time (p_j)		production cost (c_i)		production speed (v_i)	
increase of a parameter (%)	objective cost	change (%)	objective cost	change (%)	objective cost	change (%)
-15	557.114	-20.33	605.651	-13.39	848.502	21.34
-10	597.286	-14.58	636.958	-8.91	788.858	12.81
-5	646.374	-7.56	668.203	-4.44	738.987	5.68
0	699.264	0.00	699.264	0.00	699.264	0.00
5	740.892	5.95	730.377	4.45	650.271	-7.01
10	788.671	12.79	761.518	8.90	607.529	-13.12
15	836.936	19.69	792.929	13.39	575.381	-17.72
correlation coefficient	0.999627		0.999999		-0.997053	

by GA and ICA. For most settings, ICA achieves lower costs and its RDPs are always less than 2.73%. On the other hand, compared with GA, ICA takes less run time when n is large. This is because biodiversity is easy to be achieved by ICA. Compared with GA, ICA has multiple dependent populations and GA just has one. Some potential solutions of ICA are therefore not weeded out.

In the third part, a sensitivity test is provided in Fig. 8 to show the influences of parameters l and a (i.e., $A/300$). The default values of other parameters are $n = 10$ and $m = 2$. It is clear that the objective cost will decrease intensively if we have large due windows. That is, we have more flexibility to schedule jobs. Moreover, loose limits of earliness and tardiness also leads to lower objective costs. The decreasing speed of a is faster than that of l. It implies that we had better negotiate a large limit instead of requesting a wider due window.

Table 11 provides another sensitivity test for three input parameters (i.e., p_j, c_i, v_i). An increase of processing time (15%) will cause an increase of objective cost (19.69%). The more processing time a job has, the more objective cost we must pay. An increase of production cost of each machine (15%) will lead to an increase of objective cost (13.39%). The objective cost is directly affected by the input parameters. On the other hand, an increase of production speed of each machine (15%) will result in a decrease of objective cost (17.72%). It means that we can earn profit by improving the factory facilities, although investing them require some cost.

VII. CONCLUSION

This study considers a multi-machine scheduling problem with a due window. Given a limit of total earliness and tardiness, we aim to minimize the total cost including environmental damage. The difficulty of this problem resides in the variety of machines and distinction between jobs. A branch-and-bound algorithm (B&B) is proposed. Using the dominance rules and two complementary lower bounds, B&B deals successfully with the instances of $n \leq 15$ and $m \leq 3$. Moreover, an imperialist competitive algorithm (ICA) is developed to provide near-optimal solutions. The experimental results show that ICA performs well for $n \leq 50$. In the future, we may consider due windows with different penalties for earliness and tardiness and develop some metaheuristic for obtaining near-optimal schedules for a large n, e.g., 100.

REFERENCES

[1] W.-C. Lee and C.-C. Wu, "Multi-machine scheduling with deteriorating jobs and scheduled maintenance," *Appl. Math. Model.*, vol. 32, no. 3, pp. 362–373, 2008.
[2] S. Balin, "Non-identical parallel machine scheduling using genetic algorithm," *Expert Syst. Appl.*, vol. 38, no. 6, pp. 6814–6821, 2011.
[3] V. Kayvanfar, G. M. Komaki, A. Aalaei, and M. Zandieh, "Minimizing total tardiness and earliness on unrelated parallel machines with controllable processing times," *Comput. Oper. Res.*, vol. 41, pp. 31–43, Jan. 2014.
[4] K.-T. Fang and B. M. T. Lin, "Parallel-machine scheduling to minimize tardiness penalty and power cost," *Comput. Ind. Eng.*, vol. 64, no. 1, pp. 224–234, 2013.
[5] D. Thiruvady, G. Singh, A. T. Ernst, and B. Meyer, "Constraint-based ACO for a shared resource constrained scheduling problem," *Int. J. Prod. Econ.*, vol. 141, no. 1, pp. 230–242, 2013.
[6] J.-Y. Wang, "A branch-and-bound algorithm for minimizing the total tardiness of a three-agent scheduling problem considering the overlap effect and environment protection," *IEEE Access*, vol. 7, pp. 5106–5123, 2019.
[7] M. Ji, J.-Y. Wang, and W.-C. Lee, "Minimizing resource consumption on uniform parallel machines with a bound on makespan," *Comput. Oper. Res.*, vol. 40, no. 12, pp. 2970–2974, 2013.
[8] W. C. Yeh, M.-C. Chuang, and W.-C. Lee, "Uniform parallel machine scheduling with resource consumption constraint," *Appl. Math. Model.*, vol. 39, no. 8, pp. 2131–2138, 2015.
[9] J.-Y. Wang, "Minimizing the total weighted tardiness of overlapping jobs on parallel machines with a learning effect," *J. Oper. Res. Soc.*, to be published.
[10] M. J. Anzanello, F. S. Fogliatto, and L. Santos, "Learning dependent job scheduling in mass customized scenarios considering ergonomic factors," *Int. J. Prod. Econ.*, vol. 154, pp. 136–145, Aug. 2014.
[11] L. Gimeno, M. T. M. Rodrigues, L. C. A. Rodrigues, and W. Alvarenga, "Mixed integer linear programming and constrained based search approaches in a multipurpose batch plant short term scheduling problem," *Comput. Aided Chem. Eng.*, vol. 8, pp. 1039–1044, Jan. 2000.
[12] P. A. Castro and I. E. Grossmann, "An efficient MILP model for the short-term scheduling of single stage batch plants," *Comput. Chem. Eng.*, vol. 30, nos. 6–7, pp. 1003–1018, 2006.
[13] P. Castro and I. Grossmann, "Multiple time grid continuous-time formulation for the short term scheduling of multiproduct batch plants," *Comput. Aided Chem. Eng.*, vol. 21, pp. 2093–2098, Jan. 2006.
[14] J. Lamothe, F. Marmier, M. Dupuy, P. Gaborit, and L. Dupont, "Scheduling rules to minimize total tardiness in a parallel machine problem with setup and calendar constraints," *Comput. Oper. Res.*, vol. 39, no. 6, pp. 1236–1244, 2012.
[15] L.-H. Su and Y.-Y. Tien, "Minimizing mean absolute deviation of completion time about a common due window subject to maximum tardiness for a single machine," *Int. J. Prod. Econ.*, vol. 134, no. 1, pp. 196–203, 2011.
[16] E. Gerstl and G. Mosheiov, "Due-window assignment problems with unit-time jobs," *Appl. Math. Comput.*, vol. 220, pp. 487–495, Sep. 2013.
[17] N. Shirvani, R. Ruiz, and S. Shadrokh, "Cyclic scheduling of perishable products in parallel machine with release dates, due dates and deadlines," *Int. J. Prod. Econ.*, vol. 156, pp. 1–12, Oct. 2014.
[18] Z. Xingong and W. Yong, "Single-machine scheduling CON/SLK due window assignment problems with sum-of-processed times based learning effect," *Appl. Math. Comput.*, vol. 250, pp. 628–635, Jan. 2015.
[19] A. J. Ruiz-Torres, J. C. Ho, and F. J. Lopez, "Generating Pareto schedules with outsource and internal parallel resources," *Int. J. Prod. Econ.*, vol. 103, no. 2, pp. 810–825, 2006.
[20] M. M. Mazdeh, F. Zaerpour, A. Zareei, and A. Hajinezhad, "Parallel machines scheduling to minimize job tardiness and machine deteriorating cost with deteriorating jobs," *Appl. Math. Model.*, vol. 34, no. 6, pp. 1498–1510, 2010.
[21] I. Sariçiçek and C. Çelik, "Two meta-heuristics for parallel machine scheduling with job splitting to minimize total tardiness," *Appl. Math. Model.*, vol. 35, no. 8, pp. 4117–4126, 2011.
[22] W. von Hoyningen-Huene and G. P. Kiesmuller, "Evaluation of the expected makespan of a set of non-resumable jobs on parallel machines with stochastic failures," *Eur. J. Oper. Res.*, vol. 240, no. 2, pp. 439–446, 2015.
[23] M. D. Toksari and E. Güner, "Parallel machine earliness/tardiness scheduling problem under the effects of position based learning and linear/nonlinear deterioration," *Comput. Oper. Res.*, vol. 36, no. 8, pp. 2394–2417, 2009.
[24] S. K. Garg, C. S. Yeo, A. Anandasivam, and R. Buyya, "Environment-conscious scheduling of hpc applications on distributed cloud-oriented data centers," *J. Parallel Distrib. Comput.*, vol. 71, no. 6, pp. 732–749, 2011.
[25] A. Galizia and A. Quarati, "Job allocation strategies for energy-aware and efficient grid infrastructures," *J. Syst. Softw.*, vol. 85, no. 7, pp. 1588–1606, 2012.
[26] M. Rager, C. Gahm, and F. Denz, "Energy-oriented scheduling based on evolutionary algorithms," *Comput. Oper. Res.*, vol. 54, pp. 218–231, Feb. 2015.
[27] J. Y. Wang, F. G. Chen, and J. S. Chen, "A green pump scheduling algorithm for minimizing power consumption and land depletion," *Concurrent Eng.-Res. Appl.*, vol. 21, no. 2, pp. 121–128, 2013.
[28] A. Janiak, W. A. Janiak, T. Krysiak, and T. Kwiatkowski, "A survey on scheduling problems with due windows," *Eur. J. Oper. Res.*, vol. 242, no. 2, pp. 347–357, Apr. 2015.
[29] Z.-L. Chen and C.-Y. Lee, "Parallel machine scheduling with a common due window," *Eur. J. Oper. Res.*, vol. 136, no. 3, pp. 512–527, 2002.
[30] E. Gerstl and G. Mosheiov, "Due-window assignment with identical jobs on parallel uniform machines," *Eur. J. Oper. Res.*, vol. 229, no. 1, pp. 41–47, 2013.
[31] R. H. Huang, C. L. Yang, and H. T. Huang, "Parallel machine scheduling with common due windows," *J. Oper. Res. Soc.*, vol. 61, no. 4, pp. 640–646, 2010.
[32] W.-K. Yeung, C. Oğuz, and T.-C. E. Cheng, "Two-machine flow shop scheduling with common due window to minimize weighted number of early and tardy jobs," *Naval Res. Logistics*, vol. 56, no. 7, pp. 593–599, 2009.
[33] A. Janiak, W. A. Janiak, and R. Januszkiewicz, "Algorithms for parallel processor scheduling with distinct due windows and unit-time jobs," *Bull. Polish Acad. Sci.-Tech. Sci.*, vol. 57, no. 3, pp. 209–215, 2009.
[34] J. Behnamian, M. Zandieh, and S. M. T. F. Ghomi, "Due window scheduling with sequence-dependent setup on parallel machines using three hybrid metaheuristic algorithms," *Int. J. Adv. Manuf. Technol.*, vol. 44, nos. 7–8, pp. 795–808, 2009.
[35] W.-K. Yeung, C. Oguz, and T. C. E. Cheng, "Minimizing weighted number of early and tardy jobs with a common due window involving location penalty," *Ann. Oper. Res.*, vol. 108, nos. 1–4, pp. 33–54, 2001.
[36] W. K. Yeung, C. Oğuz, and T. C. E. Cheng, "Single-machine scheduling with a common due window," *Comput. Oper. Res.*, vol. 28, no. 2, pp. 157–175, Feb. 2001.
[37] J.-B. Wang and C. Wang, "Single-machine due-window assignment problem with learning effect and deteriorating jobs," *Appl. Math. Model.*, vol. 35, no. 8, pp. 4017–4022, 2011.
[38] T. C. E. Cheng, S.-J. Yang, and D.-L. Yang, "Common due-window assignment and scheduling of linear time-dependent deteriorating jobs and a deteriorating maintenance activity," *Int. J. Prod. Econ.*, vol. 135, no. 1, pp. 154–161, 2012.
[39] G. Mosheiov and A. Sarig, "Scheduling with a common due-window: Polynomially solvable cases," *Inf. Sci.*, vol. 180, no. 8, pp. 1492–1505, 2010.
[40] L. Min and W. Cheng, "Genetic algorithms for the optimal common due date assignment and the optimal scheduling policy in parallel machine earliness/tardiness scheduling problems," *Robot. Comput.-Integr. Manuf.*, vol. 22, no. 4, pp. 279–287, 2006.
[41] V. Kayvanfar, I. Mahdavi, and G. M. Komaki, "Single machine scheduling with controllable processing times to minimize total tardiness and earliness," *Comput. Ind. Eng.*, vol. 65, no. 1, pp. 166–175, 2013.

[42] H. Yuan, I. Ahmad, and C.-C. J. Kuo, "Performance-constrained energy reduction in data centers for video-sharing services," *J. Parallel Distrib. Comput.*, vol. 75, pp. 29–39, Jan. 2015.
[43] Í. Goiri, J. L. Berral, J. O. Fitó, F. Julià, R. Nou, J. Guitart, R. Gavaldà, and J. Torres, "Energy-efficient and multifaceted resource management for profit-driven virtualized data centers," *Future Gener. Comput. Syst.*, vol. 28, no. 5, pp. 718–731, 2012.
[44] C.-Y. Cheng and L.-W. Huang, "Minimizing total earliness and tardiness through unrelated parallel machine scheduling using distributed release time control," *J. Manuf. Syst.*, vol. 42, pp. 1–10, Jan. 2017.
[45] A. Janiak, W. Janiak, M. Y. Kovalyov, E. Kozan, and E. Pesch, "Parallel machine scheduling and common due window assignment with job independent earliness and tardiness costs," *Inf. Sci.*, vol. 224, pp. 109–117, Mar. 2013.
[46] F. Ahmadizar and S. Farhadi, "Single-machine batch delivery scheduling with job release dates, due windows and earliness, tardiness, holding and delivery costs," *Comput. Oper. Res.*, vol. 53, pp. 194–205, Jan. 2015.
[47] M. Ji, W. Y. Zhang, L. J. Liao, T. C. E. Cheng, and Y. Y. Tan, "Multitasking parallel-machine scheduling with machine-dependent slack due-window assignment," *Int. J. Prod. Res.*, vol. 57, no. 6, pp. 1667–1684, 2019.
[48] W. Li, Z. Zhang, H. Liu, and J. Yuan, "Online scheduling of equal-length jobs with incompatible families on multiple batch machines to maximize the weighted number of early jobs," *Inf. Process. Lett.*, vol. 112, no. 12, pp. 503–508, 2012.
[49] B. Alidaee and D. Rosa, "Scheduling parallel machines to minimize total weighted and unweighted tardiness," *Comput. Oper. Res.*, vol. 24, no. 8, pp. 775–788, 1997.
[50] I. A. Chaudhry and P. R. Drake, "Minimizing total tardiness for the machine scheduling and worker assignment problems in identical parallel machines using genetic algorithms," *Adv. Manuf. Technol.*, vol. 42, nos. 5–6, pp. 581–594, 2009.
[51] S. A. Kravchenko and F. Werner, "Minimizing total tardiness on parallel machines with preemptions," *J. Scheduling*, vol. 15, no. 2, pp. 193–200, 2012.
[52] F. Yu, P. Wen, and S. Yi, "A multi-agent scheduling problem for two identical parallel machines to minimize total tardiness time and makespan," *Adv. Mech. Eng.*, vol. 10, no. 2, 2018, Art. no. 1687814018756103.
[53] C.-H. Lee, "A dispatching rule and a random iterated greedy metaheuristic for identical parallel machine scheduling to minimize total tardiness," *Int. J. Prod. Res.*, vol. 56, no. 6, pp. 2292–2308, 2018.
[54] I. A. Chaudhry and I. A. Q. Elbadawi, "Minimisation of total tardiness for identical parallel machine scheduling using genetic algorithm," *Sādhanā*, vol. 42, no. 1, pp. 11–21, 2017.
[55] G. Gong, Q. Deng, X. Gong, W. Liu, and Q. Ren, "A new double flexible job-shop scheduling problem integrating processing time, green production, and human factor indicators," *J. Cleaner Prod.*, vol. 174, pp. 560–576, Feb. 2018.
[56] G. H. Wan and B. P.-C. Yen, "Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties," *Eur. J. Oper. Res.*, vol. 142, no. 2, pp. 271–281, 2002.
[57] Y. Wu and D. W. Wang, "Optimal single-machine scheduling about a common due window with earliness/tardiness and additional penalties," *Int. J. Syst. Sci.*, vol. 30, no. 12, pp. 1279–1284, 1999.
[58] H. G. Kahlbacher and T. C. E. Cheng, "Parallel machine scheduling to minimize costs for earliness and number of tardy jobs," *Discrete Appl. Math.*, vol. 47, no. 2, pp. 139–164, 1993.
[59] L. Zhang, Y. Zhang, and Q. Bai, "Two-stage medical supply chain scheduling with an assignable common due window and shelf life," *J. Combinat. Optim.*, vol. 37, no. 1, pp. 319–329, 2019.
[60] C.-L. Chen and C.-L. Chen, "A bottleneck-based heuristic for minimizing makespan in a flexible flow line with unrelated parallel machines," *Comput. Oper. Res.*, vol. 36, no. 11, pp. 3073–3081, 2009.
[61] K. Zhao, X. Lu, and M. Gu, "A new approximation algorithm for multi-agent scheduling to minimize makespan on two machines," *J. Scheduling*, vol. 19, no. 1, pp. 21–31, 2016.
[62] M. Gu, J. Gu, and X. Lu, "An algorithm for multi-agent scheduling to minimize the makespan on *m* parallel machines," *J. Scheduling*, vol. 21, no. 5, pp. 483–492, 2018.
[63] S. Wang, X. Wang, J. Yu, S. Ma, and M. Liu, "Bi-objective identical parallel machine scheduling to minimize total energy consumption and makespan," *J. Clean. Prod.*, vol. 193, pp. 424–440, Aug. 2018.
[64] S.-S. Li, R.-X. Chen, Q. Feng, and C.-W. Jiao, "Parallel-machine scheduling with job-dependent cumulative deterioration effect and rejection," *J. Combinat. Optim.*, vol. 38, no. 3, pp. 957–971, 2019.
[65] I. Kucukkoc, "MILP models to minimise makespan in additive manufacturing machine scheduling problems," *Comput. Oper. Res.*, vol. 105, pp. 58–67, May 2019.
[66] S. A. Mansouri, E. Aktas, and U. Besikci, "Green scheduling of a two-machine flowshop: Trade-off between makespan and energy consumption," *Eur. J. Oper. Res.*, vol. 248, no. 3, pp. 772–788, 2016.
[67] S.-W. Lin and K.-C. Ying, "Uniform parallel-machine scheduling for minimizing total resource consumption with a bounded makespan," *IEEE Access*, vol. 5, pp. 15791–15799, 2017.
[68] K. Li and S.-L. Yang, "Non-identical parallel-machine scheduling research with minimizing total weighted completion times: Models, relaxations and algorithms," *Appl. Math. Model.*, vol. 33, no. 4, pp. 2145–2158, Apr. 2009.
[69] R. Mellouli, C. Sadfi, C. Chu, and I. Kacem, "Identical parallel-machine scheduling under availability constraints to minimize the sum of completion times," *Eur. J. Oper. Res.*, vol. 197, no. 3, pp. 1150–1165, 2009.
[70] S. Rubaiee and M. B. Yildirim, "An energy-aware multiobjective ant colony algorithm to minimize total completion time and energy cost on a single-machine preemptive scheduling," *Comput. Ind. Eng.*, vol. 127, pp. 240–252, Jan. 2019.
[71] K. Li, X. Zhang, J. Y.-T. Leung, and S. L. Yang, "Parallel machine scheduling problems in green manufacturing industry," *J. Manuf. Syst.*, vol. 38, pp. 98–106, Jan. 2016.
[72] J.-Y. Wang and J.-S. Chen, "A data partition method for MEMS-based storage devices in a distributed computing environment," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 23, no. 1, pp. 101–115, 2013.
[73] H. Zhao, H. Ma, G. Han, and L. Zhao, "A ptas for common due window scheduling with window penalty on identical machines," in *Proc. Int. Conf. Comput. Appl. Syst. Modeling*, Shanxi, China, Oct. 2010, pp. 648–652.
[74] A. Janiak, W. Janiak, M. Y. Kovalyov, and F. Werner, "Soft due window assignment and scheduling of unit-time jobs on parallel machines," *4OR-A Quart. J. Oper. Res.*, vol. 10, no. 4, pp. 347–360, 2012.
[75] A. Janiak, W. Janiak, and M. Y. Kovalyov, "Due window assignment and scheduling on parallel machines: A FPTAS for a bottleneck criterion," *Bull. Polish Acad. Sci.-Tech. Sci.*, vol. 62, no. 4, pp. 805–808, 2014.
[76] H.-B. Shi and J.-B. Wang, "Research on common due window assignment flowshop scheduling with learning effect and resource allocation," *Eng. Optim.*, to be published.
[77] J. Li, D. Xu, and H. Li, "Single machine due window assignment scheduling problem with precedence constraints and fuzzy processing times," *J. Intell. Fuzzy Syst.*, vol. 34, no. 6, pp. 4301–4314, 2018.
[78] B. Mor, "Minmax common due-window assignment and scheduling on a single machine with two competing agents," *J. Oper. Res. Soc.*, vol. 69, no. 4, pp. 589–602, 2018.
[79] L. Liu, J.-J. Wang, F. Liu, and M. Liu, "Single machine due window assignment and resource allocation scheduling problems with learning and general positional effects," *J. Manuf. Syst.*, vol. 43, no. 1, pp. 1–14, Apr. 2017.
[80] R. C. T. Lee, S. S. Tseng, R. C. Chang, and Y. T. Tsai, *Introduction to the Design & Analysis of Algorithms*. New York, NY, USA: McGraw-Hill, 2005.
[81] M. Yousefi, M. Yousefi, and A. N. Darus, "A modified imperialist competitive algorithm for constrained optimization of plate-fin heat exchangers," *J. Power Energy*, vol. 226, no. 8, pp. 1050–1059, 2012.
[82] S. Talatahari, B. F. Azar, R. Sheikholeslami, and A. H. Gandomi, "Imperialist competitive algorithm combined with chaos for global optimization," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 17, no. 3, pp. 1312–1319, 2012.
[83] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. Surv.*, vol. 35, no. 3, pp. 268–308, Sep. 2003.

JEN-YA WANG received the Ph.D. degree in computer science and engineering from National Chung Hsing University, Taiwan, in 2009. He is currently a Professor with the Department of Computer Science and Information Management, Hungkuang University, Taiwan. His research interests include optimization algorithm, database systems, patent search, medical image, and artificial intelligence.