International Journal of Innovative Computing, Information and Control
Volume **8**, Number **5(A)**, May **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 3279–3297

UNRELATED PARALLEL MACHINE SCHEDULING WITH SEQUENCE- AND MACHINE-DEPENDENT SETUP TIMES AND DUE DATE CONSTRAINTS

Kuo-Ching Ying[1] and Shih-Wei Lin[2,*]

[1]Department of Industrial Engineering and Management
National Taipei University of Technology
No. 1, Sec. 3, Chung-hsiao E. Rd., Taipei 10608, Taiwan

[2]Department of Information Management
Chang Gung University
No. 259, Wen-Hwa 1st Rd., Kwei-Shan, Tao-Yuan 333, Taiwan
*Corresponding author: swlin@mail.cgu.edu.tw

Received January 2011; revised May 2011

Abstract. *This study deals with the unrelated parallel machine scheduling problem with sequence- and machine-dependent setup times under due date constraints, a core topic for numerous industrial applications. In view of the computational complexity, an artificial bee colony (ABC) algorithm is presented to minimize the total tardiness. The performance of the proposed ABC algorithm is evaluated by comparing its solutions with those of state-of-the-art algorithms on the same benchmark problem set. Computational results show that the proposed ABC algorithm significantly outperforms existing algorithms for most problem combinations. This study offers a useful contribution to the growing body of both theoretical and practical ABC algorithms useful in scheduling problems.*
Keywords: Unrelated parallel machines, Total tardiness, Sequence-dependent setup times, Machine-dependent setup times

1. Introduction.

The problem addressed in this study is the scheduling of N available jobs on M unrelated parallel machines with sequence- and machine-dependent setup times to minimize the total tardiness under due date constraints. Placing unrelated machines in parallel to enhance routing flexibility is common in many production systems in the textile, chemical, electronics manufacturing, plastics forming and service industries [1]. In an unrelated parallel machines environment, the machines are non-identical to one another and cannot be fully correlated by simple rate adjustments. Since the speeds of different machines do not have constant relationships, the processing time of jobs depends not only upon the job but also the characteristics of machine to which it is assigned. In a sense, each unrelated machine has its own matrix of sequence-dependent setup times.

A sequence- and machine-dependent setup time is usually incurred when switching between different types of jobs [2,3]. This type of setup can be found in many manufacturing processes (e.g., drilling operations for printed circuit board fabrication and dicing operations for semiconductor wafer manufacturing) [4]. With increasing on-time delivery requirements from customers, tardiness-related measures have become one of the most active research criteria for different scheduling problems over the past two decades [5]. Given the importance of this issue for industry, total tardiness is adopted as performance criterion in this study. In addition, we consider the constraint that certain jobs have deadlines in order to meet the real-world requirement that the orders of primary customers must not be delayed. Using the regular three-field scheduling notation [6], the problem addressed in this study can be signified by $Rm\left|s_{mjk}, d_j\right|\sum T_j$ which is employed throughout this paper.

Efficient scheduling is crucial to improving the performance of manufacturing systems for survival in today's intensely competitive business environment [7,8]. Since the initial study of McNaughton [9], a growing body of research has been directed at the parallel machine scheduling problem (PMSP). In the literature, the possible machine environments of PMSPs can be generally classified into identical, uniform and unrelated machines in parallel [10], in which the unrelated parallel machine environments can be characterized as machines that perform the same function but have different capabilities or capacities, and represent a generalized environment of the previous two categories. While extensive work has been carried out on different PMSPs, the unrelated PMSP has been far less studied than the other two environments [11].

Exact algorithms for unrelated PMSPs have been presented in Lancia [12], Liaw et al. [13], and more recently in Shim and Kim [14] and Rocha et al. [15]. As the majority of unrelated PMSPs are known to be NP-hard [16], developing efficient exact algorithms for practical sized problems is a challenge for researchers and practitioners. The high computational burden and time constraints motivate researchers to seek efficient approximation algorithms that generate (near-) optimal solutions with relatively lower computational costs [17-19].

Currently, available approximation algorithms for solving unrelated PMSPs can be generally classified into two categories: constructive heuristics (CHs) and improvement heuristics (IHs). For CHs such as those proposed and evaluated by Suresh and Chaudhuri [20], Adamopoulos and Pappis [21], Bank and Werner [22], Logendran and Subur [23], Logendrana et al. [24] and Zhang et al. [25], once a job sequence is determined, it is fixed and cannot be reversed. IHs start with an initial solution and then provide a scheme for iteratively obtaining an improved solution. Well-known research on this approach includes Bruno et al. [26], Horn [27], Hariri and Potts [28], Azizoglu and Kirca [29], Bank and Werner [22], Weng et al. [30], Al-Salem [31] and Ghirardi and Potts [32]. Recent interest in developing efficient IHs has focused on metaheuristic-based algorithms which provide a rather general algorithmic framework that can be applied to different optimization problems with minor modification [33]. Metaheuristic-based IHs for unrelated PMSP include: genetic algorithm (GA)-based algorithm [34], simulates annealing (SA)-based algorithm [34-37], Tabu search (TS)-based algorithm [23,34,38,39] and iterated greedy (IG)-based algorithm [40]. For more on the theoretical and practical advances of related algorithms, we refer the reader to the survey work of Mokotoff [41] and Pfund et al. [42].

The above studies have contributed significantly to the search for (near-) optimal solutions for different extremely challenging unrelated PMSPs. However, based on the literature review, the $Rm\left|s_{mjk}, d_j\right|\sum T_j$ problem remains underrepresented in the research literature. To the best of our knowledge, except for Chen [37] and Lin et al. [40] who addressed the $Rm\left|s_{mjk}, d_j\right|\sum T_j$ problem, studies either did not consider sequence- and machine-dependent setup times and/or dealt with other performance criteria without due date constraints.

In view of both the theoretical challenges and the broad industrial implications, in this study, an artificial bee colony (ABC)-based algorithm is presented to address the $Rm\left|s_{mjk}, d_j\right|\sum T_j$ problem. The new proposed ABC-based algorithm has four unique features: (1) a new solution representation that is suitable for solving the $Rm\left|s_{mjk}, d_j\right|\sum T_j$ problem and can facilitate the decoding of a schedule; (2) a new neighborhood solution generation approach that takes advantage of the destruction and construction phases of the iterated greedy (IG)-based algorithm [40] is incorporated into the procedures of the proposed ABC-based algorithm; (3) a crossover operation from GA-based algorithm is applied to explore local variants of the current solutions; and (4) a new local search approach is presented to efficiently improve the incumbent solution. The remainder of this article is structured as follows: following the definition of the $Rm\,|s_{mjk}, d_j\,|\sum T_j$ problem in Section 2, the proposed ABC-based algorithm is elaborated in Section 3; through computational experiments on the same benchmark problem set in Section 4, the effectiveness and efficiency of the proposed ABC-based algorithm is empirically evaluated by comparing its performance against state-of-the-art metaheuristic-based algorithms; finally, concluding remarks and recommendations for future research are presented in Section 5.

2. Problem Definition.

The $Rm\,|s_{mjk}, d_j\,|\sum T_j$ problem considered in this paper can be formulated as a more complex variation of the unrelated PMSP, which involves sequence- and machine-dependent setup times, and deadline constraints. Since the classic PMSP with sequence-dependent setup times is strongly NP-hard [43], the $Rm\,|s_{mjk}, d_j\,|\sum T_j$ problem is clearly strongly NP-hard. To formulate this intractable problem, consider a set of N given jobs, each requiring a single operation on exactly one of the M unrelated parallel machines. Each job can be classified into one of L mutually and collectively exhaustive product types. Let p_{mj} $(m = 1, 2, \dots, M;\ j = 1, 2, \dots, N)$ be the processing time of job j, which depends on its assigned machine m. If two consecutive jobs have the same product type, the setup time will equal zero. Associated with each job j is an assigned due date d_j $(j = 1, 2, \dots, N)$ in which certain jobs of primary customers have deadline constraints. Prior to processing each job k, if there is a switch between different types of jobs, a sequence- and machine-dependent setup time s_{mjk} $(m = 1, 2, \dots, M;\ j, k = 1, 2, \dots, N$ and $j \neq k)$ is incurred when job k is scheduled immediately following job j on the same machine m.

Moreover, the $Rm\,|s_{mjk}, d_j\,|\sum T_j$ problem considered in this study satisfies the following major assumptions:

- All jobs are available for processing at the beginning (i.e., time zero) of the planning horizon, and have no precedence constraints among them.
- Each machine can handle no more than one job at a time and is persistently available to process all scheduled jobs when required.
- Each job can be processed by exactly one of the free machines at any given time without preemption; that is the process cannot be interrupted before completion once a job starts to undergo processing.
- The number of jobs is larger than the number of machines; i.e., $N > M$ in order to avoid trivial cases.
- The number of jobs, their processing times, their due dates and their setup times are non-negative integers that have been identified beforehand.

With these definitions, the objective is to find a feasible schedule $\pi = \{\pi_1, \pi_2, \dots, \pi_M\}$ for all of the jobs that minimizes the total tardiness $\sum T_j$, where π_m $(m = 1, 2, \dots, M)$ represents the sequence of jobs on machine m; $T_j = \max\{C_j - d_j, 0\}$ in which C_j denotes the completion time of job j.

3. The Proposed ABC-based Algorithm.

The ABC algorithm is a novel swarm based metaheuristic algorithm that was introduced by Karaboga [44] for optimizing numerical problems. Satisfactory ABC-based algorithms' results have been reported for application to the numerical test functions optimization problem [45], structural inverse analysis [46], the leaf-constrained minimum spanning tree problem [47], the assignment problem [48] multi-objective design optimization of composites [49], the visual target recognition problem [50] and clustering analysis [51].

ABC algorithm is inspired by the intelligent foraging behavior of a honeybee swarm. The algorithm posits foraging artificial bees that are grouped into three groups: employed bees, onlookers and scouts. A bee that is currently exploiting a food source is called an employed bee. A bee that is waiting in the hive to make a decision regarding a food source is called an onlooker. A bee that is performing a random search for a new food source is called a scout. An employed bee that abandons a food source becomes a scout. When employed bees and onlookers perform the exploitation process in the search space, the scouts control the exploration process. In the original ABC algorithm, each solution to the problem of interest is treated as a food source and represented by an n-dimensional real-valued vector, and the fitness of the solution is represented by the amount of nectar in the associated food resource. The number of employed bees or onlookers is set equal to the number of food sources in the population. In other words, for each food source, only one bee is employed.

Like other swarm intelligence-based approaches, the ABC algorithm is iterative. It begins with a population of randomly generated food sources (solutions). A typical iteration of ABC proceeds as follows: the employed bees are placed on their food sources, and the onlookers are also placed on food sources in a manner that depends on their nectar amounts; scouts are sent to the search area to discover new food sources. Finally, the best food source yet found is recorded to memory. This process is repeated until the termination condition is satisfied.

The following subsection gives a detailed discussion of the proposed ABC-based algorithm for solving the $Rm\,|s_{mjk}, d_j\,|\sum T_j$ problem. First, the solution representation (encoding scheme) that makes a solution recognizable to the proposed algorithm and the initial solution generated approach is presented. Then, the fitness value, the neighborhood solution, the local search (LS) approach, the employed bee phase, the onlooker phase, the scout phase and the parameters are explained. Finally, the procedures of the proposed ABC-based algorithm are described and an example is given.

3.1. **Representation of solution and initial solutions.** The solution representation in the original ABC algorithm is an n-dimensional real-valued vector which may be not suitable for solving the $Rm\,|s_{mjk}, d_j\,|\sum T_j$ problem. Accordingly, permutation-based representation can facilitate the decoding of a schedule. Therefore, in this study, it is represented by a string of numbers that consist of a permutation of N jobs and is given by the set $\{1, 2, \ldots, N\}$ and $M - 1$ zeros, where M is the number of machines. For example, a solution representation [15, 6, 5, 3, 8, 17, 0, 1, 18, 9, 7, 11, 2, 19, 0, 10, 13, 14, 4, 16, 12, 20] can be decoded as follows. There are 20 jobs to be processed by three unrelated machines. The operating sequence of the jobs on machines 1, 2 and 3 are 15-6-5-3-8-17, 1-18-9-7-11-2-19 and 10-13-14-4-16-12-20, respectively.

To generate the initial solution, the primary customers and the shortest completion time first (PCSCTF) rule, proposed by Lin et al. [40], is used for the $Rm\,|s_{mjk}, d_j\,|\sum T_j$ problem. The PCSCTF rule has two steps:

***Step* 1.** Divide the jobs into two classes: primary customers and non-primary customers. Rank the jobs within each class in ascending order of their due dates.

***Step* 2.** Sequentially, dispatch each of the jobs within the class of primary customers to the machine which has the shortest completion time for the current assigned job. Subsequently, use the same method to arrange each of the jobs within the class of non-primary customers until all jobs are assigned.

Since the jobs of primary customers cannot be delayed, the PCSCTF rule that dispatches them first is sensible. However, it is still possible to obtain an infeasible solution that violates the due date constraints. In view of the fact that sometimes accepting an infeasible solution may help the algorithm to jump out of a local optimal, in this study, an infeasible solution can be temporarily accepted with a penalty cost (P_{cost}) per unit time of tardiness for each delayed job with respect to the primary customers. Finally, the initial solution is improved by applying the LS approach (see Section 3.4) to the solution obtained by the PCSCTF rule.

3.2. **Fitness value.** The quality of a solution is determined by the value of its fitness function. Due to the due date constraints of primary customers, a penalty cost is added to the fitness function when an infeasible solution is obtained. In this study, the penalty cost is calculated as the unit penalty times the total tardiness of the delayed jobs of primary customers. Meanwhile, the fitness function of a solution is defined as $fit(\pi) = 1/[1 + Penalty(\pi) + TT(\pi)]$, where $Penalty(\pi)$ denotes the penalty cost of a (infeasible) schedule π and $TT(\pi)$ represents the total tardiness cost of a schedule π. This equation reveals that a lower total tardiness cost and penalty cost corresponds to a higher fitness value.

3.3. **Neighborhood solutions.** In this study, the employed bees generate food sources (solutions) in the neighborhood close to their current positions. This process is structured by a permutation-based representation. Based on the permutation-based neighborhood structure, the destruction and construction phases of the IG algorithm [52] are applied to generate neighborhood solutions of the current solution. The procedure for generating a neighborhood solution is denoted as $IG(\pi, \alpha)$ herein, where π is the current solution and α is a parameter. Briefly, the two steps of $IG(\pi, \alpha)$, i.e., destruction and construction phases, are described as follows:

***Step* 1.** (*Destruction phase*): Randomly select α unrepeatable jobs from π. Delete them from π and add them to π_{R} in the order in which they were chosen, where π_{R} is the sequence of the α removed jobs.

***Step* 2.** (*Construction phase*): Sequentially, reinsert the jobs of π_{R} into π_{D} until a complete solution is obtained, where π_{D} is the partial sequence of π that is left after α jobs have been removed. During the construction procedure, the best solution (π_{new}) that is obtained by inserting the jobs of π_{R} into all possible positions of the current subsequence is recorded.

3.4. **LS approach.** In this study, the LS approach outlined in Figure 1 was applied to improve π_{new}. As shown in Figure 1, the LS approach improves π_{new} by iteratively exchanging jobs in the solution with jobs positioned after them. Notably, when a better solution (π_{temp}) is obtained, the LS procedure is applied again by setting the index values to the initial value.

3.5. **Employed bee phase.** If the solution (π^i) corresponding to an employed bee x_i ($\forall i$) equals π_{best}, a new solution π^i_{new} for this employed bee is generated from the neighborhood solutions described in Section 3.3. Otherwise, a new solution π^i_{new} for this employed bee is generated by the following steps of order crossover [53,54]:

***Step* 1.** Select a substring from π_{best} at random.

***Step* 2.** Produce a proto-child by copying the substring into its corresponding positions.

***Step* 3.** Delete the jobs/machines that are already in the substring from the second parent. The resulting sequence of jobs/machines contains the jobs that the proto-child needs.

***Step* 4.** Place the jobs into the unfixed positions of the proto-child from left to right according to the order of the sequence to produce an offspring.

```
Procedure Local_Search ( $\pi_{new}$ )
  Begin
    $L = N + M - 1$ (the number of element in the solution list);
    DoExhange:
      for ( $i = 1; i \le L - 1; i{+}{+}$ )
      {
        for ( $j = i + 1; j \le L; j{+}{+}$ )
        {
          $\pi_{temp}$ = solution obtained by exchange the $i^{th}$ and $j^{th}$ elements of $\pi_{new}$;
          If ( $obj(\pi_{temp})$ is better than $obj(\pi_{new})$ )
          {
            $\pi_{new} = \pi_{temp}$;
            Go to DoExhange;
          }
        }
      }
      return $\pi_{new}$;
  End
```

FIGURE 1. Local search procedure

If π^i_{new} is better than the best solution (π^*) found so far by the proposed ABC-based algorithm, or the relative difference of fitness function values between π^i_{new} and π^*, that is $[fit(\pi^*) - fit(\pi^i_{new})]/fit(\pi^*)$, is smaller than a predetermined value (*threshold*), the LS approach is applied to improve π^i_{new}.

A new neighborhood solution π^i_{new} is always accepted if it is better than or equal to π^i. Therefore, once π^i_{new} is obtained, it is compared to π^i. If $fit(\pi^i_{new}) \ge fit(\pi^i)$, then π^i is replaced by π^i_{new} and π^i_{new} becomes a new member of the population; otherwise, π^i is retained.

3.6. **Onlooker phase.** An onlooker evaluates the nectar information taken from all the employed bees and selects a solution (food source) π^i with a probability p_i, given by $p_i = f_i / \sum_{i=1}^{SN} f_i$, where f_i is the amount of nectar (fitness value) of π^i and *SN* is the number of food sources. Clearly, a higher f_i corresponds to a higher probability that π^i is selected. Once, a solution π^i is selected by an onlooker, a new solution (π^i_{new}) is generated using the neighboring procedure described in Section 3.3. Then, the LS approach as applied in the employed bee phase can be implemented on π^i_{new}. If the π^i_{new} obtained from the LS procedure has at least as much nectar as π^i, then π^i_{new} will replace π^i and become a new member of the population.

3.7. **Scout phase.** If π^i cannot be further improved within a predetermined number of trials (i.e., *limit*), then it is assumed to be abandoned, and the corresponding employed bee becomes a scout. The scout generates a new food source randomly.

3.8. **Parameters.** The proposed ABC-based algorithm starts with seven parameters: *SN*, *threshold*, *limit*, $\alpha_{\max}$, $G_{\max}$, $G_{\text{non-improving}}$ and P_{cost}. *SN* represents the number of food sources, which equals the number of the employed bees or onlookers. That is, the colony size is $2 \times SN$. *limit* is the number of trials after which a food source is assumed to be abandoned, and *threshold* is used to determine whether the LS approach is applied after the neighborhood solution has been generated. $\alpha_{\max}$ is the maximal value for α which is used for generating neighborhood solutions with respect to $IG(\pi, \alpha)$. Before $IG(\pi, \alpha)$

```
ABC ($SN$, $threshold$, $limit$, $\alpha$, $G_{\max}$, and $G_{\text{non-improving}}$)
{
    Generate initial population $Pop = (\pi^1, \pi^2, \ldots, \pi^{SN})$ by the PCSFCF rule and randomly for $\pi^1$ and
        $(\pi^2, \ldots, \pi^{SN})$, respectively, and improve them by the local search approach;
    $G$=0; $G_{non}$=0;
    Calculate the fitness of each solution $fit(\pi^i)$, $i \in Pop$;
    Do ($G \le G_{\max}$ and $G_{non} \le G_{non-improving}$)
    {
        $G$=$G$+1; $G_{non}$= $G_{non}$ +1;
        //Place the employed bees on their food sources.
        For $i = 1, 2, \ldots, SN$
        {
            $\alpha$ is determined by randomly choosing an integer value between 1 and $\alpha_{\max}$;
            Obtain new solution $\pi^i_{new}$ by IG($\pi^i$, $\alpha$);
            If $[fit(\pi^i_{new}) - fit(\pi^*)]/fit(\pi^*) < threshold$ or $fit(\pi^i_{new}) \ge fit(\pi^*)$ Perform LS($\pi^i_{new}$);
            If $fit(\pi^i_{new}) \ge fit(\pi^i)$ replace $\pi^i$ by $\pi^i_{new}$;
        }
        //Place the onlooker bees on the food sources depending on their nectar amounts.
        For $i = 1, 2, \ldots, SN$
        {
            Selects a food source $\pi^i$ depending on its probability value $p_i$ calculated by $p_i = f_i / \sum_{i=1}^{SN} f_i$;
            $\alpha$ is determined by randomly choosing an integer value between 1 and $\alpha_{\max}$;
            Obtain new solution $\pi^i_{new}$ by IG($\pi^i$, $\alpha$);
            If $[fit(\pi^i_{new}) - fit(\pi^*)]/fit(\pi^*) < threshold$ or $fit(\pi^i_{new}) \ge fit(\pi^*)$ Perform LS($\pi^i_{new}$);
            If $fit(\pi^i_{new}) \ge fit(\pi^i)$ replace $\pi^i$ by $\pi^i_{new}$;
        }
        // Send the scouts to the search area for discovering new food sources.
        For $i = 1, 2, \ldots, SN$
        {
            If (food source $\pi^i$ is not changed in successive $limit$ times) then regenerated $\pi^i$ randomly;
        }
        // Memorize the best food source found so far.
        For $i = 1, 2, \ldots, SN$
        {
            If $fit(\pi^i_{new}) \ge fit(\pi^*)$
            {
                $\pi^* = \pi^i$;
                $G_{non} = 0$;
            }
        }
    }
    Output the best food source found so far ($\pi^*$);
}
```

FIGURE 2. Pseudo-code of the proposed ABC approach

is used, the value of α is obtained by randomly generating an integer value between 1 and $\alpha_{\max}$. That is, the value of α may be changed for each bee in each iteration. $G_{\max}$ is the maximum number of iterations and the $G_{\text{non-improving}}$ is the allowable number of generations for which the best obtained objective function value is not improved by the proposed ABC-based algorithm. P_{cost} is the penalty cost per unit time of tardiness for each delayed job with respect to the primary customers.

3.9. **Procedures of the proposed ABC-based algorithm.** Figure 2 presents the pseudo-code of the proposed ABC-based algorithm. In the first step, the initial population $Pop = (\pi^1, \pi^2, \ldots, \pi^{SN})$ is generated by the PCSCTF rule (π^1) or by random permutation ($\pi^2, \ldots, \pi^{SN}$), and then the initial solutions within the initial population are improved by the local search approach; Subsequently, the fitness value is calculated for each solution in the initial population. In the following main loop, each iteration involves procedures in the following order. First, the employed bees are placed on their food sources. Second, the onlookers are placed on the food sources in a manner that depends on the amount of nectar in each source. Third, the scouts are sent to the search area to discover new food sources. Fourth, the best food source found so far is saved to memory (π^*). At the end of each iteration, the termination condition is checked. If the number of iterations G exceeds $G_{\max}$ or π^* is not improved in $G_{\text{non-improving}}$ successive generations, then the proposed ABC algorithm is terminated; otherwise, a new iteration will begin and the iteration number G is increased by one. Following the termination of the proposed ABC-based algorithm, the (near) global optimal solution can be derived from π^*.

3.10. **An example.** To demonstrate the main procedures of the proposed ABC-based algorithm, we offer the following example. Suppose there are 10 jobs to be scheduled on two unrelated parallel machines (detailed data for this problem is shown in Tables 1-3). The penalty cost (P_{cost}) per unit time of tardiness for each delayed job with respect to the primary customers is set to 500, and the *threshold* value for determining whether to perform the LS approach is set to 5.

3.10.1. *Initial solutions.* Suppose the initial solutions (food sources) are obtained and shown in Table 4. At the current state, the best solution found by the algorithm is solution #5.

Table 1. Due-date, primary customer label and processing time in machines 1 and 2

Job	Due Date	Primary Customer	Processing Time in Machine 1	Processing Time in Machine 2
1	321	Yes	96	83
2	173	No	63	75
3	493	No	67	77
4	124	Yes	43	58
5	404	No	87	72
6	197	No	45	60
7	203	Yes	28	29
8	348	No	72	65
9	209	No	73	65
10	82	No	31	34

TABLE 2. Setup time between jobs in machine 1

S_{ij}	1	2	3	4	5	6	7	8	9	10
1	0	51	87	37	87	87	17	17	51	87
2	29	0	40	72	40	40	25	25	0	40
3	13	70	0	9	0	0	72	72	70	0
4	10	24	65	0	65	65	55	55	24	65
5	13	70	0	9	0	0	72	72	70	0
6	13	70	0	9	0	0	72	72	70	0
7	74	48	29	57	29	29	0	0	48	29
8	74	48	29	57	29	29	0	0	48	29
9	29	0	40	72	40	40	25	25	0	40
10	13	70	0	9	0	0	72	72	70	0

TABLE 3. Setup time between jobs in machine 2

S_{ij}	1	2	3	4	5	6	7	8	9	10
1	0	31	57	25	57	57	28	28	31	57
2	67	0	36	84	36	36	72	72	0	36
3	4	58	0	48	0	0	43	43	58	0
4	64	66	49	0	49	49	50	50	66	49
5	4	58	0	48	0	0	43	43	58	0
6	4	58	0	48	0	0	43	43	58	0
7	83	61	70	15	70	70	0	0	61	70
8	83	61	70	15	70	70	0	0	61	70
9	67	0	36	84	36	36	72	72	0	36
10	4	58	0	48	0	0	43	43	58	0

TABLE 4. Initial solutions (food sources) and their total tardiness values

Sol.	Sequence	Total Tardiness	Sol.	Sequence	Total Tardiness
1	4-2-7-6-5-3-0-10-1-9-8	78	4	10-2-1-0-7-8-4-6-5-3-9	473
2	4-2-9-6-0-10-1-7-8-5-3	91	5	4-2-7-8-0-10-6-1-9-5-3	68
3	10-6-4-2-9-5-3-0-8-7-1	136	6	7-4-2-8-0-10-6-5-1-9-3	186

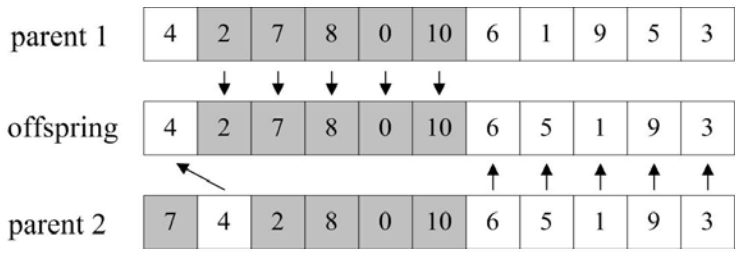

FIGURE 3. Illustration of the crossover operator in the proposed ABC approach

3.10.2. *Send employed bee.* In the employed bee phase, $IG(\pi, \alpha)$ is applied to the current best solution (π^*), i.e., solution #5. Other solutions will be used to generate neighborhood solutions by the crossover operation with solution #5. Figure 3 shows how the crossover operation is applied to solution #6. The solutions obtained by employed bees before local search are shown in Table 5. Because of the relative difference of fitness function values (i.e., $[fit(\pi^i_{new}) - fit(\pi^*)]/fit(\pi^*)$) of solutions #1, #2, #4 and #6 are less than 5, the

LS approach is applied to these solutions, respectively. New solutions obtained after local search are listed in Table 6, which shows that solutions #1, #2 and #4 are improved by the LS approach. Since a better solution obtained in this phase will substitute for its old food source, solutions (food sources) #4 and #6 are replaced. Table 7 lists all of the new solutions (food sources) obtained at the end of this phase.

TABLE 5. Solutions obtained in "send employee bee" phase (before local search)

Sol.	Sequence	Total Tardiness	Sol.	Sequence	Total Tardiness
1	4-2-8-7-6-5-3-0-10-1-9	152	4	2-7-8-4-0-10-6-1-9-5-3	232
2	4-2-7-9-6-0-10-1-8-5-3	287	5	4-2-8-10-1-5-0-3-7-9-6	38186
3	10-6-4-2-5-3-0-1-9-8-7	73192	6	4-2-7-8-0-10-6-5-1-9-3	140

TABLE 6. Solutions obtained in "send employee bee" phase (after local search)

Sol.	Sequence	Total Tardiness	Sol.	Sequence	Total Tardiness
1	4-2-7-8-6-5-3-0-10-9-1	144	4	7-4-2-8-0-10-6-1-9-5-3	114
2	4-2-7-6-8-0-10-1-9-5-3	121	5	4-2-8-10-1-5-0-3-7-9-6	38186
3	10-6-4-2-5-3-0-1-9-8-7	73192	6	4-2-7-8-0-10-6-5-1-9-3	140

TABLE 7. Updated solutions (food sources) and their total tardiness values

Sol.	Sequence	Total Tardiness	Sol.	Sequence	Total Tardiness
1	4-2-7-6-5-3-0-10-1-9-8	78	4	7-4-2-8-0-10-6-1-9-5-3	114
2	4-2-9-6-0-10-1-7-8-5-3	91	5	4-2-7-8-0-10-6-1-9-5-3	68
3	10-6-4-2-9-5-3-0-8-7-1	136	6	4-2-7-8-0-10-6-5-1-9-3	140

3.10.3. *Send onlooker bee.* An onlooker evaluates the nectar information taken from all of the employed bees and selects a new solution (food source) with a certain probability. Suppose the new solution #4 is evaluated and chosen by an onlooker, and then the neighborhood solution of the new solution #4 is generated by applying $IG(\pi, \alpha)$ as shown

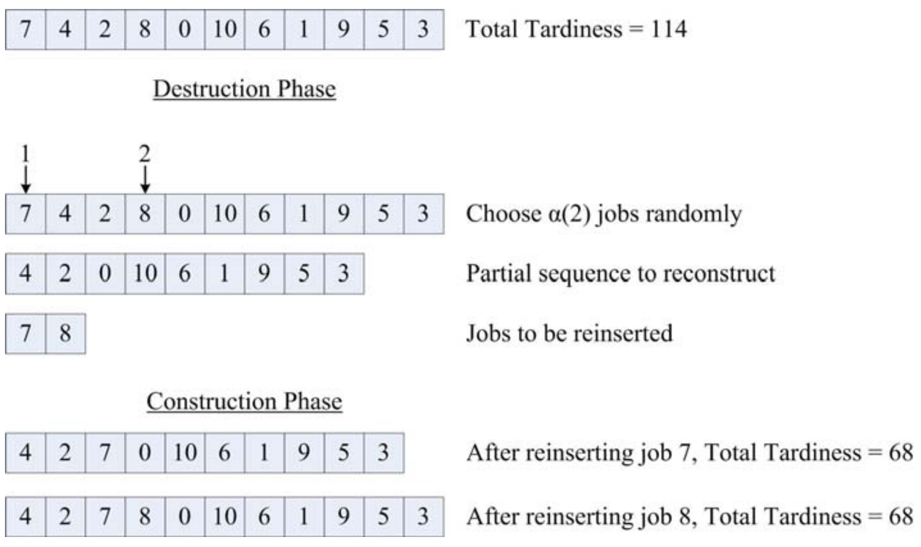

FIGURE 4. Illustration of the $IG(\pi, \alpha)$ procedures ($\alpha = 2$)

in Figure 4. Table 8 lists the neighborhood solutions obtained by applying $IG(\pi, \alpha)$ for all of the new solutions. After each onlooker has found a neighborhood solution, the LS approach may be applied to the individual neighborhood solutions. Again, the LS approach will be applied to these solutions (Table 9) and will update them once a better solution was found. As Table 10 shows, in this example, solutions #3 and #4 are replaced by better solutions.

TABLE 8. Solutions obtained in the "send outlook bee" phase (before local search)

Sol.	Sequence	Total Tardiness	Food Source	Sol.	Sequence	Total Tardiness	Food Source
1	4-2-7-6-5-3-0-10-1-9-8	78	1	4	6-4-2-9-5-3-0-8-7-10-1	179	3
2	6-4-9-10-0-1-7-8-5-3-2	571	2	5	4-2-9-6-0-10-1-7-8-5-3	91	2
3	4-2-7-8-0-6-10-1-9-5-3	80	5	6	4-2-7-8-0-10-6-1-9-5-3	68	4

TABLE 9. Solutions obtained in the "send outlook bee" phase (after local search)

Sol.	Sequence	Total Tardiness	Food Source	Sol.	Sequence	Total Tardiness	Food Source
1	4-2-7-6-5-3-0-10-1-9-8	78	1	4	10-4-2-9-5-3-0-8-7-6-1	65	3
2	6-4-9-10-0-1-7-8-5-3-2	571	2	5	4-2-9-6-0-10-1-7-8-5-3	91	2
3	4-2-7-8-0-10-6-1-9-5-3	68	5	6	4-2-7-8-0-10-6-1-9-5-3	68	4

TABLE 10. Updated solutions (food sources) and their total tardiness values

New Sol.	Sequence	Total Tardiness	New Sol.	Sequence	Total Tardiness
1	4-2-7-6-5-3-0-10-1-9-8	78	4	4-2-7-8-0-10-6-1-9-5-3	68
2	4-2-9-6-0-10-1-7-8-5-3	91	5	4-2-7-8-0-10-6-1-9-5-3	68
3	10-4-2-9-5-3-0-8-7-6-1	65	6	4-2-7-8-0-10-6-5-1-9-3	140

4. **Computational Results and Discussion.**

4.1. **Test problems.**

The performance of the proposed ABC-based algorithm was evaluated by a total of 960 test instances from a well-known benchmark problem set. The benchmark problem set was generated by Chen [37], who presented an effective heuristic based on a modified apparent-tardiness-cost-with-setup procedure, a random descent heuristic (RDH) and an SA-based heuristic (SA_HEU) with designed improvement procedures for solving the $Rm\,|s_{mjk}, d_j\,|\sum T_j$ problem. The analytical results demonstrated by Chen [37] showed that the SA_HEU significantly outperforms the MATCS procedure and the RDH.

As summarized in Table 11, the benchmark problem set involves the following six factors: number of jobs (N), number of machines (M), number of families (F), due date priority factor (τ), due date range factor (R) and proportion of jobs from primary customers (P). The number of jobs was set to 30, 50, 70 and 90; the number of machines to 4, 6 and 8; and each of the remaining four factors has two levels. For each combination of the six factors, five instances were randomly generated. Therefore, the test bed was comprised of $4 \times 3 \times 2^4 \times 5 = 960$ benchmark instances. The files of test instances and the solutions of the SA_HEU and the RDH are available via Chen's web site (http://140.134.72.86/data-set.htm).

Table 11. Summarized levels of the six factors for Chen's benchmark problem set

Factor	Level
Number of jobs (N)	30, 50, 70, 90
Number of machines (M)	4, 6 ,8
Number of job families (F)	$\lfloor N/7 \rfloor + 1$, $\lfloor N/8 \rfloor + 1$
Due date priority factor (τ)	0.4, 0.8
Due date range factor (R)	0.4, 1
Proportion of jobs from primary customers (P)	0.2, 0.3

4.2. **Results and discussion.** The effectiveness and efficiency of the proposed ABC-based algorithm is compared with that of existing algorithms found in the literature. In addition to the SA_HEU and the RDH [37], we also compared the computational results of the proposed ABC-based algorithm with the IG-based heuristic (IG_HEU) [40], which is a state-of-the-art metaheuristic-based algorithm performed on the same benchmark problem sets. To compare these algorithms on the same basis, the proposed ABC-based algorithm was coded in C computer language and each instance was replicated 10 times on a PC with an Intel Pentium IV (3.4 GHz) CPU. The RDH, the SA_HEU and the IG_HEU were similarly executed.

Parameter selection can influence the quality of results. In this study, the optimized parameter values were chosen based on both solution quality and computational efficiency. In the preliminary optimization stage, the following combinations of the parameter values were tested on 20 randomly selected instances from problem sets: $SN = 5, 6, 7, 8, 9$; *threshold* = 2, 3, 4, 5, 6, 7; *limit* = 100, 150, 200, 250; $\alpha_{\max} = 3, 4, 5, 6$; $G_{\max} = 600, 700, 800, 900, 1000$; $G_{\text{non-improving}} = 150$, 180, 210, 240, 270; $P_{\text{cost}} = 300$, 400, 500, 600, 700. Based on extensive preliminary tests, the following parameters were used in the experimental results reported in this paper: $SN = 6$, *threshold* = 5, *limit* = 150, $\alpha_{\max} = 5$, $G_{\max} = 900$, $G_{\text{non-improving}} = 210$ and $P_{\text{cost}} = 500$.

The computational results are summarized in Table 12. As depicted in Table 12, the effectiveness of these algorithms was compared by listing the mean value of the $\sum T_j$ of the best solutions among 10 trials for each instance of a specific sub-problem set. On the whole, the total average value of the $\sum T_j$ obtained by the proposed ABC-based algorithm is 5041.0. Compared with the corresponding values of 5483.6, 5165.4 and 5089.2 obtained by the RDH, the SA_HEU and the IG_HEU, respectively, the proposed ABC-based algorithm outperforms the RDH, the SA_HEU and the IG_HEU. At the same time, the mean values of the $\sum T_j$ were improved by the proposed ABC-based approach for most sub-problem sets, while total average improvement rates of 8.07%, 2.41% and 0.95% were reached for the RDH, the SA_HEU and the IG_HEU, respectively. In view of the fact that the computational time of the proposed ABC-based approach is nearly the same as the SA_HEU and the IG_HEU, the results clearly indicate the superiority of the proposed ABC-based algorithm.

To offer further insight, the effectiveness of the proposed ABC-based algorithm was compared with that of the SA_HEU and IG_HEU across six indexes: Min. Sol., Max. Sol., Ave. Sol., No. tie, No. better and No. worse, where Min. Sol., Max. Sol and Ave. Sol. represent the average minimal, maximal and mean $\sum T_j$ among 10 trials for each instance

TABLE 12. Computational results of RDH, SA_HEU, IG_HEU and the proposed ABC-based algorithm

		RDH		SA_HEU		IG_HEU		ABC		Solution of	Solution of	Solution of
		Solution	CPU Time	Solution	CPU time	Solution	CPU Time	Solution	CPU Time	RDH improved by ABC (%)	SA_HEU improved by ABC (%)	IG_HEU improved by ABC (%)
N	30	1,483.0	181.29	1,412.5	5.71	1,396.0	1.02	1,393.5	1.26	6.04	1.35	0.18
	50	3,177.0	157.54	2,986.7	12.26	2,955.8	6.21	2,939.7	6.03	7.47	1.57	0.54
	70	6,498.1	150.05	6,101.6	26.06	6,061.1	22.04	5,992.3	23.14	7.78	1.79	1.14
	90	10,776.4	151.28	10,160.8	47.36	9,943.8	64.55	9,838.3	63.06	8.71	3.17	1.06
M	4	8,335.8	156.61	7,941.5	35.63	7,767.0	23.91	7,693.7	25.01	7.70	3.12	0.94
	6	5,127.2	161.27	4795.6	19.54	4,760.4	23.96	4,715.6	24.93	8.03	1.67	0.94
	8	2,987.9	162.23	2,759.1	13.38	2,740.1	22.50	2,713.6	20.18	9.18	1.65	0.97
F	$\lfloor N/7 \rfloor + 1$	5,486.9	156.91	5166.5	22.64	5,092.1	23.40	5,042.1	23.58	8.11	2.41	0.98
	$\lfloor N/8 \rfloor + 1$	5,480.4	163.17	5164.3	23.06	5,086.2	23.50	5,039.8	23.17	8.04	2.41	0.91
τ	0.4	3.6	20.08	1.4	0.97	1.4	0.07	1.4	0.16	61.11	0.00	0.00
	0.8	10,963.6	300.00	10,329.4	44.73	10,177.0	46.84	10,080.5	46.59	8.05	2.41	0.95
R	0.4	9,440.0	157.55	9,047.7	27.07	8,924.2	20.43	8,828.3	28.32	6.48	2.42	1.07
	1.0	1,527.3	162.53	1,283.1	18.63	1,254.1	26.49	1,253.6	18.43	17.92	2.30	0.04
P	0.2	5,230.5	158.17	4,967.1	22.67	4,922.7	22.80	4,869.9	24.59	6.89	1.96	1.07
	0.3	5,736.7	161.90	5,363.7	23.03	5,255.7	24.10	5,212.0	22.16	9.15	2.83	0.83
Total Average		5,483.6	160.04	5,165.4	22.85	5,089.2	23.45	5,041.0	23.37	8.07	2.41	0.95

Table 13. Detailed comparisons of the proposed ABC-based algorithm with SA_HEU and IG_HEU

		SA_HEU			IG_HEU			ABC			Best solution		
		Min. Sol.	*Max. Sol.*	*Ave. Sol.*	*Min. Sol.*	*Max. Sol.*	*Ave. Sol.*	*Min. Sol.*	*Max. Sol.*	*Ave. Sol.*	*No. tie*	*No. better*	*No. worse*
N	30	1404.3	1419.2	1412.5	1389.7	1404.8	1396.0	1389.2	1402.8	1393.5	187	51	2
	50	2961.1	3004.9	2986.7	2912.3	3010.7	2955.8	2903.7	3001.0	2939.7	149	76	15
	70	6040.1	6140.0	6101.6	5942.3	6199.2	6061.1	5891.2	6129.2	5992.3	127	92	21
	90	10020.5	10254.3	10160.8	9731.5	10169.0	9943.8	9649.3	10054.1	9838.3	125	95	20
M	4	7861.5	7995.0	7941.5	7644.8	7894.9	7767.0	7590.2	7819.7	7693.7	185	129	6
	6	4737.6	4835.7	4795.6	4666.6	4871.2	4760.4	4634.8	4826.5	4715.6	177	131	12
	8	2720.4	2783.1	2759.1	2670.4	2821.6	2740.1	2650.1	2794.0	2713.6	226	54	40
F	$\lfloor N/7 \rfloor + 1$	5104.2	5206.7	5166.5	4994.2	5198.3	5092.1	4956.8	5149.3	5042.1	287	160	33
	$\lfloor N/8 \rfloor + 1$	5108.7	5202.5	5164.3	4993.7	5193.5	5086.2	4959.9	5144.2	5039.8	301	154	25
τ	0.4	1.3	1.5	1.4	1.3	1.5	1.4	1.3	1.6	1.4	480	0	0
	0.8	10211.7	10407.7	10329.4	9986.6	10390.4	10177.0	9915.4	10291.9	10080.5	108	314	58
R	0.4	8972.8	9100.7	9047.7	8800.9	9060.0	8924.2	8733.4	8948.9	8828.3	289	174	17
	1.0	1240.2	1308.5	1283.1	1187.0	1331.9	1254.1	1183.3	1344.6	1253.6	299	140	41
P	0.2	4920.4	4997.3	4967.1	4837.8	5016.2	4922.7	4799.2	4960.3	4869.9	299	154	27
	0.3	5292.5	5411.9	5363.7	5150.1	5375.6	5255.7	5117.5	5333.2	5212.0	289	160	31
Total Average		5106.5	5204.6	5165.4	4993.9	5195.9	5089.2	4958.4	5146.7	5041.0	588	314	58

TABLE 14. Paired t-tests on *Min. Sol.*, *Ave. Sol.* and *Max. Sol.*

ABC vs.		**SA_HEU**			**IG_HEU**		
		Min. Sol.	*Max. Sol.*	*Ave. Sol.*	*Min. Sol.*	*Max. Sol.*	*Ave. Sol.*
N	30	(4.823, 0.000)*	(4.089, 0.000)	(5.534, 0.000)	(0.750, 0.227)	(0.643, 0.260)	(1.564, 0.059)
	50	(7.562, 0.000)	(0.451, 0.326)	(6.043, 0.000)	(2.471, 0.007)	(1.414, 0.079)	(3.481, 0.000)
	70	(6.661, 0.000)	(0.458, 0.324)	(5.036, 0.000)	(4.567, 0.000)	(3.613, 0.000)	(4.964, 0.000)
	90	(7.156, 0.000)	(3.309, 0.001)	(5.700, 0.000)	(4.013, 0.000)	(3.971, 0.000)	(4.506, 0.000)
M	4	(7.134, 0.000)	(3.951, 0.000)	(5.910, 0.000)	(3.610, 0.000)	(3.543, 0.000)	(4.202, 0.000)
	6	(5.617, 0.000)	(0.492, 0.311)	(4.343, 0.000)	(4.186, 0.000)	(3.404, 0.000)	(4.918, 0.000)
	8	(7.538, 0.000)	(–1.040, 0.150)	(5.601, 0.000)	(3.455, 0.000)	(2.663, 0.004)	(4.048, 0.000)
F	$\lfloor N/7 \rfloor + 1$	(6.619, 0.000)	(2.093, 0.018)	(4.909, 0.000)	(3.712, 0.000)	(3.740, 0.000)	(4.554, 0.000)
	$\lfloor N/8 \rfloor + 1$	(7.784, 0.000)	(3.108, 0.004)	(6.645, 0.000)	(5.216, 0.000)	(3.974, 0.000)	(5.469, 0.000)
τ	0.4	(1.000, 0.159)	(–0.761, 0.224)	(1.487, 0.069)	(0.000, 0.500)	(–1.104, 0.155)	(–0.830, 0.204)
	0.8	(10.675, 0.000)	(3.505, 0.000)	(8.167, 0.000)	(6.052, 0.000)	(5.540, 0.000)	(7.131, 0.000)
R	0.4	(8.592, 0.000)	(4.794, 0.000)	(7.245, 0.000)	(5.948, 0.000)	(6.967, 0.000)	(7.362, 0.000)
	1.0	(7.991, 0.000)	(–4.448, 0.000)	(4.757, 0.000)	(1.131, 0.000)	(–1.694, 0.045)	(0.144, 0.443)
P	0.2	(8.719, 0.000)	(3.304, 0.001)	(7.688, 0.000)	(3.912, 0.000)	(4.496, 0.000)	(4.872, 0.000)
	0.3	(6.785, 0.000)	(2.549, 0.006)	(5.264, 0.000)	(4.793, 0.000)	(3.245, 0.001)	(5.032, 0.000)
Overall		(10.097, 0.000)	(3.483, 0.000)	(7.902, 0.000)	(5.943, 0.000)	(5.451, 0.000)	(6.951, 0.000)

*(t-value, p-value)

of a specific sub-problem set, while the No. tie, No. better and No. worse denote that the number of Min. Sol. obtained by the proposed ABC-based approach is equal to, better, or worse than that of the SA_HEU and IG_HEU, respectively. Table 13 shows that the total average of Min. Sol., Max. Sol., Ave. Sol. of the proposed ABC-based algorithm is 4958.4, 5146.7 and 5041.0, respectively. Compared with the corresponding values of 5106.5, 5204.6 and 5165.4 obtained by the SA_HEU, and 4993.9, 5195.9 and 5089.2 obtained by IG_HEU, the proposed ABC-based approach outperforms the SA_HEU and the IG_HEU on all of the performance indexes. No. tie, No. better and No. worse obtained by the proposed ABC-based algorithm were 588, 314 and 58, respectively, meaning that 32.71% of its solutions were better than those obtained by SA_HEU and the IG_HEU, while 61.25% solutions remained the same. From these computational results, it is clear that the main advantage of the proposed ABC-based algorithm over other state-of-the-art algorithms is its effectiveness in finding (near-) optimal solutions.

To further verify the effectiveness of the proposed ABC-based algorithm, paired t-tests were performed on Min. Sol., Max. Sol., Ave. Sol. to compare them against the SA_HEU and the IG_HEU. The analytical results are listed in Table 14. At the 5% confidence level, the results in Table 14 show that the proposed ABC-based approach significantly outperformed the SA_HEU and IG_HEU with respect to Min. Sol., Max. Sol. and Ave. Sol. These statistical results confirm the superiority of the proposed ABC-based approach. Judging from the above experimental results, it is clear that this study has made an important step towards reducing the gap between theoretical progress and industrial practice.

5. Concluding Remarks.

The $Rm\,|s_{mjk}, d_j|\sum T_j$ problem is important in practical terms, but is currently underrepresented in the research literature. In view of theoretical challenges and broad industrial implications, an ABC-based algorithm is proposed in this study to reduce the gap between theory and practice. To evaluate the applicability of the proposed ABC-based algorithm, its performance was compared with that of the best available algorithms on the same benchmark problem set. The analytical results clearly demonstrate that the proposed ABC-based algorithm is relatively more effective and efficient in minimizing total tardiness than the state-of-the-art metaheuristic-based algorithms. Judging from the experimental results, it is clear that this study represents a step towards bridging the gap between theoretical progress and industrial practice for the $Rm\,|s_{mjk}, d_j|\sum T_j$ problem. With few algorithms currently available as solutions for this NP-hard problem, we hope that practitioners will use the ABC-based algorithm presented in this paper to solve real-world $Rm\,|s_{mjk}, d_j|\sum T_j$ scheduling problems commonly found in the textile, chemical, electronic manufacturing and service industries.

This paper is offered as a contribution to the growing body of work on useful theoretical and practical optimization approaches to the $Rm\,|s_{mjk}, d_j|\sum T_j$ problem. Many interesting topics in this area deserve further attention. First, we believe that the ABC-based algorithm proposed herein will also be helpful in other scheduling problems. Second, it would be worthwhile to develop additional efficient and effective exact algorithms and meta-heuristics in this area. Third, the proposed ABC-based algorithm should be extended to solve similar unrelated PMSPs with sequence- and machine-dependent setup times under due date constraints in subsequent studies. Fourth, more papers that report and explain the real-world experiences of the $Rm\,|s_{mjk}, d_j|\sum T_j$ problem would be beneficial to researchers and practitioners in this area. Finally, to diminish the gap between theory and practice, solutions for multi-criteria unrelated PMSPs with different performance criteria and setup time configurations should be explored in future research.

Acknowledgment. This research was partially supported by the National Science Council of Taiwan and the Chang Gung University under contract numbers NSC 100-2221-E-027-040-MY2 and UARPD3A0031, respectively.

REFERENCES

[1] J.-F. Chen, Unrelated parallel machine scheduling with secondary resource constraints, *International Journal of Advanced Manufacturing Technology*, vol.26, no.3, pp.285-292, 2005.

[2] G. Rabadi, R. J. Moraga and A. Al-Salem, Heuristics for the unrelated parallel machine scheduling problem with setup times, *Journal of Intelligent Manufacturing*, vol.17, no.1, pp.85-97, 2006.

[3] K.-C. Ying, Z.-J. Lee and S.-W. Lin, Makespan minimization for scheduling unrelated parallel machines with setup times, *Journal of Intelligent Manufacturing*, 2010.

[4] C.-L. Chen and C.-L. Chen, Hybrid metaheuristics for unrelated parallel machine scheduling with sequence-dependent setup times, *International Journal of Advanced Manufacturing Technology*, vol.43, no.1-2, pp.161-169, 2009.

[5] S.-W. Lin, S.-Y. Chou and K.-C. Ying, A sequential exchange approach for minimizing earliness-tardiness penalties of single-machine scheduling with a common due date, *European Journal Operational Research*, vol.177, no.2, pp.1294-1301, 2007.

[6] R. Graham, E. Lawler, J. Lenstra and A. R. Kan, Optimization and approximation in deterministic sequencing and scheduling: A survey, *Annals of Discrete Mathematics*, vol.5, pp.287-326, 1979.

[7] K.-C. Ying and S.-W. Lin, Multi-heuristic desirability ant colony system heuristic for non-permutation flowshop scheduling problems, *International Journal of Advanced Manufacturing Technology*, vol.33, no.7-8, pp.793-802, 2007.

[8] Y. Li, Y. Yang, L. Zhou and R. Zhu, Observations on using problem-specific genetic algorithm for multiprocessor real-time task scheduling, *International Journal of Innovative Computing, Information and Control*, vol.5, no.9, pp.2531-2540, 2009.

[9] R. McNaughton, Scheduling with deadlines and loss functions, *Management Science*, vol.6, no.1, pp.1-12, 1959.

[10] T. C. E. Cheng and C. C. S. Sin, A state-of-the-art review of parallel-machine scheduling research, *European Journal Operational Research*, vol.47, no.3, pp.271-292, 1990.

[11] C. Dhaenens-Flipo, A bicriterion approach to deal with a constrained single-objective problem, *International Journal of Production Economics*, vol.74, no.1-3, pp.93-101, 2001.

[12] G. Lancia, Scheduling jobs with release dates and tails on two unrelated parallel machines to minimize the makespan, *European Journal Operational Research*, vol.120, no.2, pp.277-288, 2000.

[13] C.-F. Liaw, Y.-K. Lin, C.-Y. Chen and M. Chen, Scheduling unrelated parallel machines to minimize total weighted tardiness, *Computers & Operations Research*, vol.30, no.12, pp.1777-1789, 2003.

[14] S.-O. Shim and Y.-D. Kim, Minimizing total tardiness in an unrelated parallel-machine scheduling problem, *Journal of the Operational Research Society*, vol.58, no.3, pp.346-354, 2007.

[15] P. L. Rocha, M. G. Ravetti, G. R. Mateus and P. M. Pardalos, Exact algorithms for a scheduling problem with unrelated parallel machines and sequence and machine-dependent setup times, *Computers & Operations Research*, vol.35, no.4, pp.1250-1264, 2008.

[16] K.-C. Ying and H.-M. Cheng, Dynamic parallel machine scheduling with sequence-dependent setup times using an iterated greedy heuristic, *Expert Systems with Applications*, vol.37, no.4, pp.2848-2852, 2010.

[17] K.-C. Ying, S.-W. Lin and Z.-J. Lee, Hybrid-directional planning: Improving improvement heuristics for scheduling resource-constrained projects, *International Journal of Advanced Manufacturing Technology*, vol.41, no.3-4, pp.358-366, 2009.

[18] J.-F. Lin, Scheduling parallel tasks with intra-communication overhead in a grid computing environment, *International Journal of Innovative Computing, Information and Control*, vol.7, no.2, pp.881-896, 2011.

[19] G. Kim, S.-S. Kim, I.-H. Kim, D.-H. Kim, V. Mani and J.-K. Moon, An efficient simulated annealing with a valid solution mechanism for TDMA broadcast scheduling problem, *International Journal of Innovative Computing, Information and Control*, vol.7, no.3, pp.1181-1191, 2011.

[20] V. Suresh and D. Chaudhuri, Minimizing maximum tardiness for unrelated parallel machines, *International Journal of Production Economics*, vol.34, no.2, pp.223-229, 1994.

[21] G. I. Adamopoulos and C. P. Pappis, Scheduling under a common due-date on parallel unrelated machines, *European Journal Operational Research*, vol.105, no.3, pp.494-501, 1998.

[22] J. Bank and F. Werner, Heuristic algorithms for unrelated parallel machine scheduling with a common due date, release dates, and linear earliness and tardiness penalties, *Mathematical and Computer Modelling*, vol.33, no.4-5, pp.363-383, 2001.

[23] R. Logendran and F. Subur, Unrelated parallel machine scheduling with job splitting, *IIE Transactions*, vol.36, no.4, pp.359-371, 2004.

[24] R. Logendrana, B. McDonellb and B. Smucker, Scheduling unrelated parallel machines with sequence-dependent setups, *Computers & Operations Research*, vol.34, no.11, pp.3420-3438, 2007.

[25] Z. Zhang, L. Zheng and M. X. Weng, Dynamic parallel machine scheduling with mean weighted tardiness objective by Q-learning, *International Journal of Advanced Manufacturing Technology*, vol.34, no.9-10, pp.968-980, 2007.

[26] L. G. Bruno, E. G. Coffman and R. Sethi, Scheduling independent tasks to reduce mean finishing time, *Communications of the ACM*, vol.17, no.7, pp.382-387, 1974.

[27] W. A. Horn, Minimizing average flow time with parallel machines, *Operations Research*, vol.21, no.3, pp.846-847, 1973.

[28] A. M. A. Hariri and C. N. Potts, Heuristics for scheduling unrelated parallel machines, *Computers & Operations Research*, vol.18, no.3, pp.323-331, 1991.

[29] M. Azizoglu and O. Kirka, Scheduling jobs on unrelated parallel machines to minimize regular total cost functions, *IIE Transactions*, vol.31, no.2, pp.153-159, 1999.

[30] M. Weng, J. Lu and H. Ren, Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective, *International Journal of Production Economics*, vol.70, no.3, pp.215-226, 2001.

[31] A. Al-Salem, Scheduling to minimize makespan on unrelated parallel machines with sequence dependent setup times, *Engineering Journal of the University of Qatar*, vol.17, pp.177-187, 2004.

[32] M. Ghirardi and C. N. Potts, Makespan minimization for scheduling unrelated parallel machines: A recovering beam search approach, *European Journal Operational Research*, vol.165, no.2, pp.457-467, 2005.

[33] K.-C. Ying and C.-J. Liao, An ant colony system for permutation flow-shop sequencing, *Computers & Operations Research*, vol.31, no.5, pp.791-801, 2004.

[34] C. A. Glass, C. N. Potts and P. Shade, Unrelated parallel machine scheduling using local search, *Mathematical and Computer Modelling*, vol.20, no.2, pp.41-52, 1994.

[35] D.-W. Kim, K.-H. Kim, W. Jang and F.-F. Chen, Unrelated parallel machine scheduling with setup times using simulated annealing, *Robotics and Computer-Integrated Manufacturing*, vol.18, no.3-4, pp.223-231, 2002.

[36] D.-W. Kim, D.-G. Na and F.-F. Chen, Unrelated parallel machine scheduling with setup times and total weighted tardiness objective, *Robotics and Computer-Integrated Manufacturing*, vol.19, no.1-2, pp.173-181, 2003.

[37] J.-F. Chen, Scheduling on unrelated parallel machines with sequence- and machine-dependent setup times and due-date constraints, *International Journal of Advanced Manufacturing Technology*, vol.44, no.11-12, pp.204-1212, 2009.

[38] B. Srivastava, An effective heuristic for minimizing makespan on unrelated parallel machines, *Journal of the Operational Research Society*, vol.49, no.8, pp.886-894, 1997.

[39] J.-F. Chen, Minimization of maximum tardiness on unrelated parallel machines with process restrictions and setups, *International Journal of Advanced Manufacturing Technology*, vol.29, no.5-6, pp.557-563, 2006.

[40] S.-W. Lin, C.-C. Lu and K.-C. Ying, Minimization of total tardiness on unrelated parallel machines with sequence- and machine-dependent setup times under due date constraints, *International Journal of Advanced Manufacturing Technology*, vol.43, no.1-4, pp.353-361, 2011.

[41] E. Mokotoff, Parallel machine scheduling problems: A survey, *Asia-Pacific Journal Operational Research*, vol.18, pp.193-242, 2001.

[42] M. Pfund, J. W. Fowler and J. N. D. Gupta, A survey of algorithms for single and multi-objective unrelated parallel-machine deterministic scheduling problems, *Journal of the Chinese Institute of Industrial Engineers*, vol.21, no.3, pp.230-241, 2004.

[43] J. D. Ullman, NP-complete scheduling problem, *Journal of Computers and System Sciences*, vol.10, no.3, pp.384-93, 1975.

[44] D. Karaboga and B. Akay, A comparative study of artificial bee colony algorithm, *Applied Mathematics and Computation*, vol.214, no.1, pp.108-132, 2009.

[45] P.-W. Tsai, J.-S. Pan, B.-Y. Liao and S.-C. Chu, Enhanced artificial bee colony optimization, *International Journal of Innovative Computing, Information and Control*, vol.5, no.12(B), pp.5081-5092, 2009.
[46] F. Kang, J. Li and Q. Xu, Structural inverse analysis by hybrid simplex artificial bee colony algorithms, *Computers & Structures*, vol.87, no.13-14, pp.861-870, 2009.
[47] A. Singh, An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem, *Applied Soft Computing*, vol.9, no.2, pp.652-631, 2009.
[48] L. Ozbakir, A. Baykasoglu and P. Tapkan, Bees algorithms for generalized assignment problem, *Applied Mathematics and Computation*, vol.215, no.11, pp.3782-3795, 2010.
[49] S. N. Omkar, J. Senthilnath, R. Khandelwal, G. N. Naik and S. Gopalakrishnan, Artificial bee colony (ABC) for multi-objective design optimization of composite structures, *Applied Soft Computing*, vol.11, no.1, pp.489-499, 2011.
[50] C. Xu and H. Duan, Artificial bee colony (ABC) optimized edge potential function (EPF) approach to target recognition for low-altitude aircraft, *Pattern Recognition Letters*, vol.31, no.13, pp.1759-1772, 2010.
[51] D. Karaboga and C. Ozturk, A novel clustering approach: Artificial bee colony (ABC) algorithm, *Applied Soft Computing*, vol.11, no.1, pp.652-657, 2011.
[52] R. Ruiz and T. Stützle, A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem, *European Journal Operational Research*, vol.177, no.3, pp.2033-2049, 2007.
[53] L. Davis, Applying adaptive algorithms to Epistatic domains, *Proc. of the 9th International Joint Conference on Artificial Intelligence*, San Francisco, CA, USA, vol.1, pp.162-164, 1985.
[54] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Design*, John Wiley & Sons, 1996.