IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 21, 2017, accepted July 17, 2017, date of publication August 3, 2017, date of current version August 29, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2735538

Uniform Parallel-Machine Scheduling for Minimizing Total Resource Consumption With a Bounded Makespan

SHIH-WEI LIN[1,2,3] AND KUO-CHING YING[4]

[1]Department of Information Management, Chang Gung University, Taoyuan 33302, Taiwan
[2]Department of Neurology, Linkou Chang Gung Memorial Hospital, Taoyuan 33305, Taiwan
[3]Department of Industrial Engineering and Management, Ming Chi University of Technology, Taipei 24301, Taiwan
[4]Department of Industrial Engineering and Management, National Taipei University of Technology, Taipei 10608, Taiwan

Corresponding author: Kuo-Ching Ying (kcying@ntut.edu.tw)

The work of S.-W. Lin was supported in part by the Ministry of Science and Technology of the Republic of China, Taiwan, under Grant MOST 104-2410-H-182-011, and in part by the Linkou Chang Gung Memorial Hospital under Grant CMRPD3G0011. The work of K.-C. Ying was supported by the Ministry of Science and Technology of the Republic of China, Taiwan, under Grant MOST106-2221-E-027-085.

ABSTRACT This paper examines the uniform parallel-machine scheduling problem in which the objective aims to minimize the total resource consumption (TRC) with a bounded makespan. A matheuristic is proposed to deal with this strongly NP-hard problem. The performance of the proposed matheuristic is compared with that of the state-of-the-art particle swarm optimization (PSO) meta-heuristic and the lower bound (LB) of TRC on a set of benchmark instances. Computational results show that the proposed matheuristic significantly outperforms the PSO meta-heuristic and its solution is very close to the tight LB. Given the critical need for environmental protection, this paper provides an effective and efficient algorithm for diminishing the gap between the theoretical progress of scheduling and the practical need for environmental protection.

INDEX TERMS Operations research, scheduling algorithm, parallel machines, total resource consumption.

I. INTRODUCTION

The parallel-machine scheduling problem (PMSP) is commonly encountered in the machinery, electronics, textiles, transportation, telecommunications, pharmaceuticals, chemicals and service industries [1]. Since the first relevant study by McNaughton [2], various PMSPs have attracted continuing interest among researchers. Based on the characteristics of parallel machines, a classical parallel-machine system (PMS) can be categorized as identical, uniform, or unrelated [3]. In an identical PMS, jobs can be processed on any one machine with the same speed factors; in an uniform PMS, jobs can be processed on any one machine with different speed factors, which are in fixed ratios with each other; while in an unrelated PMS, each job can be processed on some specific but not all machines with different speed factors, which are not in fixed ratios with each other. This paper focuses on minimizing the total resource consumption (TRC) with a bounded makespan (i.e., maximum completion time) in a particular uniform parallel-machine scheduling problem (UPMSP).

The uniform PMS typically arises when factories buy new machines but retain slower and older machines. Owing to its academic and industrial importance, the UPMSP has been extensively investigated in recent decades. Regarding the exact methods, Bulfin and Parker [4] made a variety of modifications to uniform and unrelated PMSPs with two processors to minimize the makespan. Their experimental results showed that relatively large problems can always be handled routinely. Panneerselvam and Kanagalingam [5] proposed a mathematical model for solving the UPMSP while minimizing the makespan, and discussed industrial applications of such a scheduling problem. Azizoglu and Kirca [6] presented a branch-and-bound method aiming at minimizing total weighted flow time in the identical PMSP, and extended the algorithm to the UPMSP. Liao and Lin [7] studied the two-machine UPMSP, the objective is to minimize the makespan. They transformed the UPMSP into a special identical PMSP and then proposed an exact algorithm to solve it optimally. In spite of the fact that the proposed exact algorithm had an exponential time complexity, it is guaranteed to obtain the

optimal schedules in large test instances within a short computational time. Subsequently, Lin and Liao [8] developed two theorems to minimize the makespan in the UPMSP. They developed an approach with exponential time complexity, incorporating the two theorems, for effectively finding the optimal solution. Lin [9] proposed several linear programming (LP) formulations and heuristic algorithms for solving UPMSPs with identical jobs while minimizing several performance measures. The proposed LP models provided insights into the structures of the studied problems and yielded optimal schedules with a speed and efficiency that were required to satisfy practical requirements.

Zou et al. [10] studied several UPMSPs in which the processing times of jobs increased with their starting times. The objective was to minimize the total load on all machines and the total completion time of all jobs. They showed that these UPMSPs to be solvable in polynomial-time while the increasing rates of processing times were identical for all jobs. Zhang and Luo [11] considered the UPMSP, in which the processing time of a job increased with its starting time and jobs could be rejected via giving penalties. They presented an approximation scheme for minimizing the delivery time of all jobs. In view of the fact that most non-preemptive UPMSPs with more than two machines are strongly NP-hard [1], finding optimal solutions to large problems, in particular, is a challenge for researchers and practitioners. For practical purposes, most researchers inevitably develop heuristic algorithms trying to find a near-optimum in polynomial time.

Two categories of heuristic algorithms - constructive heuristics and improvement heuristics - exist for solving UPMSPs. With respect to constructive heuristics, Friesen and Langston [12] examined the UPMSP and proved the worst-case bound on the makespan of a multifit heuristic algorithm is within 1.4 times of the optimal solution. Dobson [13] presented a worst-case bound with respect to the makespan of the longest processing time (LPT) rule that is applied to the UPMSP. He derived tight bounds for the ratio of the solution obtained by the LPT rule to the optimum. Friesen [14] further proved that the worst-case bound that was obtained by the LPT rule for solving this problem was in the interval (1.52, 1.67). Chen [15] evaluated the performance of LPT rule for the same problem under different ratios of the fastest speed to the slowest speed of the system, and presented a worst-case analysis. Mireault et al. [16] dealt with the UPMSP to minimize the makespan when sequencing independent tasks on two machines, and computed the maximum relative error of the LPT rule. Burkard and He [17] combined the multifit heuristic with LPT rule in an incumbent algorithm to solve the same problem, and presented a very tight worst-case bound. Koulamas and Kyparisis [18] extended the EDD rule to solve the UPMSP with the objective of minimize the maximum lateness, which can yield a near-optimal solution.

Many improvement heuristics have been proposed to solve the strongly *NP*-hard UPMSPs more completely and effectively. De and Morton [19] developed an algorithm for solving both identical and uniform PMSPs. Experimental results revealed that the makespan values that were obtained by their heuristics for solving identical and uniform PMSPs are within 1.3% and 5%, respectively, of the best solutions that were obtained using a truncated branch-and-bound procedure. Bulfin and Parker [4] presented a heuristic scheme for solving PMSPs with two identical, uniform, and unrelated processors and precedence constraints to minimize the makespan. Their computational experiments revealed that relatively large problems could be handled routinely. Hochbaum and Shmoys [20] used the dual approximation approach to present a family of polynomial approximation scheme to minimize makespan in UPMSPs such that the suboptimal solution had an error relative to the optimum that was less than that obtained by previously developed algorithms. Burkard et al. [21] presented a linear compound algorithm to minimize makespan on two-machine UPMSPs, which had a worst-case bound of 7/6. Panneerselvam and Kanagalingam [5] presented an algorithm for solving the makespan minimization UPMSP with two processors. Koulamas and Kyparisis [22] developed a heuristic for minimizing the makespan in the UPMSP with specified release times, and presented a worst-case bound for the proposed algorithm.

Agarwal et al. [23] proposed 12 combinations of single-pass heuristics in conjunction with augmented-neural-network (AugNN) formulations for solving UPMSPs, in which the objective was to minimize the makespan. The computational results demonstrated that AugNN considerably improved upon single-pass heuristics. developed an evolutionary algorithm (EA) and a genetic algorithm (GA), respectively, for minimizing the makespan in the UPMSP. The analytical results revealed that the proposed algorithms outperformed other compared meta-heuristics, including particle swarm optimization (PSO), simulated annealing (SA), multi-objective evolutionary algorithm (MOEA), and another GA. Li and Cheng [24] presented a robust variable neighborhood search (RVNS) algorithm to solve the UPMSP with release dates. The effectiveness of the proposed RVNS algorithm was proven in a computational experiment that was performed on 2000 random instances. Elvikis et al. [25] dealt with a two-agent UPMSP that involved sequencing two jobs, each of which required multiple operations to minimize the makespan and cost functions. They provided special properties of the strict Pareto optima set, and polynomial-time algorithms for finding that set. Senthilkumar and Narayanan [26] developed three variations of the SA meta-heuristic to solve the same problem. They compared them using ANOVA and found that no significant differences between the three variations.

Senthilkumar and Narayanan [27] also presented four GA-based meta-heuristics that involved various combinations of crossover methods to minimize the makespan on the UPMSP. The computational results showed that the GA-based meta-heuristic with single point crossover method

outperformed other compared meta-heuristics. Lee et al. [28] proposed PSO and GA meta-heuristics for minimizing the makespan in the UPMSP. The computational results in various scenarios revealed that PSO outperformed GA, which was therefore recommended. Lin [29] studied the UPMSP of scheduling identical jobs on machines with position-based learning effects to minimize the total weighted completion time, total tardiness, total weighted tardiness, and maximum lateness, respectively. Their computational results demonstrated that formulating these problems as assignment problems was more efficient than formulating them as linear programming models. The comprehensive review by Senthilkumar and Narayanan [30] presents more on related theoretical and practical advances, and discussed 17 classes of UPMSPs. An overview of the literature revealed that most studies of UPMSPs assumed that the required resources for processes were unconstrained, at odds with the fact that resources were always limited in practice.

Although various UPMSPs have been extensively investigated in the past decades, the consumption of resources has seldom been considered. As increasing emphasis is being placed on environmental protection, Ji et al. [31] recently introduced a UPMSP whose objective was to minimize the TRC with a bounded makespan. Using the regular 3-tuple notation of Graham et al. [32], this UPMSP can be specified as $Q|C_{\max} \leq \hat{C}|TRC$ Ji et al. [31] proved that the $Q|C_{\max} \leq \hat{C}|TRC$ problem was *NP*-hard in the strong sense, and then derived a very tight lower bound (LB) and developed a PSO meta-heuristic for solving it. Their experimental results revealed that the PSO meta-heuristic performed extremely well with a maximum average relative error rate (RER) of 0.811% compared with the LB among all tested cases. Ji et al. [31] has tried to encourage more research on scheduling problems in the field of environmental protection. In spite of its importance, the $Q|C_{\max} \leq \hat{C}|TRC$ problem has so far received only little research attention.

To reduce the gap between theoretical progress of scheduling and the practical need for environmental protection, this work proposes a matheuristic for solving the $Q|C_{\max} \leq \hat{C}|TRC$ problem. To the best of our knowledge, this study is the first to propose a matheuristic for solving resource-dependent scheduling problems. The rest of this paper is organized as follows. Section 2 formally defines the $Q|C_{\max} \leq \hat{C}|TRC$ problem. Section 3 describes in detail the proposed matheuristic. Using a benchmark problem sets, Section 4 evaluates the performance of the proposed matheuristic by comparing it with that of the PSO meta-heuristic. Section 5 provides conclusions and suggests directions for future research.

II. PROBLEM FORMULATION

To formulate the $Q|C_{\max} \leq \hat{C}|TRC$ problem of interest herein, consider a set of n jobs $\vartheta = \{J_1, J_2, \ldots, J_n\}$ to be processed on a set of m uniform parallel-machines $M = \{U_1, U_2, \ldots, U_m\}$ that are functionally equivalent but differ in terms of processing speed. Each job $J_j \in \vartheta$ requires a single operation on either of the machines with a processing time $p_{ij} = p_j/v_i$ that depends on the absolute processing time p_j of job J_j and the processing speed v_i of machine U_i to which it is assigned. Moreover, processing jobs on machine U_i consumes β_i $(i = 1, \ldots, m)$ of resources per unit time; these resources may be water, electricity usage, or a carbon emission equivalent. Given the above definitions, the objective is to get a schedule $\Pi = \{\pi_1, \pi_2, \ldots, \pi_m\}$ for the n jobs that minimizes the TRC without violating the constraint that all the jobs must be completed before a bound on the makespan $(\hat{C})$: $C_{\max} = \max_{1 \leq j \leq n}\{C_j\} \leq \hat{C}$, where π_k $(k = 1, 2, \ldots, m)$ is the sequence of jobs on machine U_k, and C_j represents the completion time of job J_j. Notably, any job that cannot be completed before $\hat{C}$ using the m uniform parallel machines is processed on a set of outsourcing machines with higher resource consumption rates. If the outsourcing machines are assumed to be identical, then the problem can be reduced to one with a single outsourcing machine U_{m+1}, which consumes β_{m+1} of resources per unit time. This machine is not subject to the constraint that all of the jobs that are assigned to it must be completed before $\hat{C}$.

The $Q|C_{\max} \leq \hat{C}|TRC$ problem that is considered herein satisfies the following basic assumptions.

- The number of machines, the number of jobs, the bound on the makespan, the processing times, and the resources consumed by the jobs are given by deterministic non-negative integers.
- All jobs are prepared for processing at the start of the scheduling horizon, and no precedence constraints among the jobs are imposed.
- Every machine is persistently available for work whenever required, and can process just one job at a time.
- The number of jobs is more than the number of machines to eliminate trivial cases.
- The setup times are sequence independent, and have been integrated in the processing times of those jobs.
- Each job has to be processed without preemption on its assigned machine.

III. PROPOSED MATHEURISTIC

The proposed matheuristic comprises three phases. In the first phase, a simple heuristic rapidly yields an initial solution to the $Q|C_{\max} \leq \hat{C}|TRC$ problem. In the second phase, the initial solution is improved using the SA algorithm to yield a better solution. At last, in the third phase, the obtained better solution is set as the initial feasible solution and the upper bound (UB) of the proposed MILP model of the $Q|C_{\max} \leq \hat{C}|TRC$ problem. Then, the MILP model is solved using the Gurobi optimizer to find an optimal solution. The three phases of the proposed matheuristic are described in detail as follows.

A. PHASE I: USING SIMPLE HEURISTIC TO FIND AN INITIAL SOLUTION

The basic idea of the proposed simple heuristic is to exploit machines with smaller β_i/v_i values as much as possible. Suppose that machines have been sorted according the increasing

```
Procedure Local_Search (Π0)
Begin
 for ( j = 1; j ≤ n − 1; i++ ) {
        for ( k = j + 1; k ≤ n; k++ ) {
        If ( jth job and kth job are not processed in the same machine) {
              Π_temp = solution obtained by exchange the jth and kth jobs of Π0 ;
         If ( TRC(Π_temp) < TRC(Π0) ) {
           Π0 = Π_temp ;
         }
        }
       }
      }
        return Π0 ;
End
```

FIGURE 1. The procedure of the local search.

order of the value β_i/v_i; the initial solution is then generated using the simple heuristic as follows.

Step 1. Arrange the jobs in ascending order of their processing times;
Step 2. Set the starting time of each machine $ST_i = 0(\forall i)$;
Step 3. Find the unscheduled job j with the shortest processing time;
Step 4. Assign jobs j to machine i, where i is the smallest machine index that satisfies $(ST_i + p_j/v_i) \leq \hat{C}$;
Step 5. Update the starting time of machine i by adding p_j/v_i to it;
Step 6. If all jobs have been scheduled, then terminate the assignment procedure; Else go to Step 3;
Step 7. Use the pairwise interchange local search (presented in Fig. 1) to improve the obtained initial solution Π_0. Notably, the local search is designed to find as many feasible neighborhood solutions as possible within the specified period. All infeasible solutions found by the local search will be discarded. Since exchanging the jobs that are processed in the same machine does not improve the solution quality, such cases are not considered.

B. PHASE II: APPLYING THE SA HEURISTIC TO IMPROVE THE INITIAL SOLUTION

In phase II of the proposed metheuristic, an SA algorithm is used to improve the initial solution Π_0 to obtain a better solution. Figure 2 presents the steps for implementing the proposed SA algorithm. Given the initial solution Π_0, the initial temperature T_0, the maximum number of iterations at a specific temperature I_{iter}, the rate of decrease of temperature α, and the maximum permitted number of successive temperature reductions without the improvement of the incumbent solution, $N_{non-improvement}$, the proposed SA algorithm initializes the temperature T_e, the number of temperature reductions N, the incumbent solution Π, and the best solution Π_{best}. In each iteration, a new solution Π_{new} is obtained from the incumbent solution Π by randomly selecting and swapping two jobs in Π. Let $\Delta = TRC(\Pi_{\text{new}}) - TRC(\Pi)$; if $\Delta \leq 0$, then the incumbent solution Π is replaced by the new solution Π_{new}; otherwise, the probability of substituting Π with Π_{new} is $e^{(-\Delta/T_e)}$. Replacement is performed by generating a random number, r, from a uniform distribution U(0,1) and replacing Π with Π_{new} if $r < e^{(-\Delta/T_e)}$. The current temperature T_e is decreased after I_{iter} iterations from the preceding reduce, along with the formula $T_e = \alpha T_e$, where $0 < \alpha < 1$. If the current best solution is not recovered in $N_{non-improvement}$ consecutive temperature reductions, then the process is terminated. Upon termination, the (near-) optimal solution Π_{best} with its penalty cost $TRC(\Pi_{\text{best}})$ are output.

C. PHASE III: OPTIMALLY SOLVING THE MILP MODEL

Let x_{ij} be a binary variable that equals 1 if job J_j is assigned to machine U_i, and 0 otherwise. The $Q|C_{\max} \leq \hat{C}|TRC$ problem that is considered herein can be formulated as an MILP model Ji et al. [31] as follows.

The objective function is as follows.

$$\text{Minimize } TRC = \sum_{j=1}^{n} \sum_{i=1}^{m+1} \beta_i x_{ij} p_j / v_i$$

The constraints are,

$$\sum_{i=1}^{m+1} x_{ij} = 1, \quad \forall j,$$
$$\sum_{j=1}^{n} x_{ij} p_j / v_i \leq \hat{C}, \quad \forall i,$$
$$x_{ij} \in \{0, 1\}, \quad \forall i, j.$$

Procedure *Simulated _Annealing* (Π_0 , T_0 , I_{iter} , α , $N_{non-improving}$)

Setting the parameters $T_e = T_0$; $N = 0$; $\Pi = \Pi_0$; $\Pi_{\text{best}} = \Pi_0$;
WHILE ($N \le N_{\text{non-improvement}}$) {
  FOR ($I = 1; I \le I_{iter}; I{+}{+}$) {
    Generate a new solution Π_{new} from Π by swapping two randomly selected jobs;
    $\Delta = TRC(\Pi_{\text{new}}) - TRC(\Pi)$;
      IF ($\Delta \le 0$), THEN $\Pi = \Pi_{\text{new}}$;
    ELSE {
        $r =$ random U(0,1);
      IF ($r \le e^{(-\Delta/T_e)}$), THEN $\Pi = \Pi_{\text{new}}$;
    }
      IF ($TRC(\Pi) < TRC(\Pi_{\text{best}})$) {
        $\Pi_{\text{best}} = \Pi$;
        $N = 0$;
      }
  }
  $T_e = \alpha T_e$;
  $N = N + 1$;
}
Return Π_{best} and $TRC(\Pi_{\text{best}})$

FIGURE 2. The procedure of the SA algorithm.

The objective function (1) is to minimize the TRC. Constraint set (1) specifies that each job is assigned to exactly one machine. Constraint set (1) ensures that all jobs must be completed before the bound on the makespan $\hat{C}$. Constraint set (4) defines the binary variables.

To shrink the solution space and solve the MILP model more efficiently, the solution that is obtained by the SA algorithm is set as the initial feasible solution and the UB of the model. Then, by using Gurobi (version 7.0), a (near-) optimal solution can be obtained within a predefined maximal computing time. This advantage is significant, especially for large-scale $Q|C_{\max} \le \hat{C}|TRC$ problems.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

This section describes the details of the benchmark problem set of Ji et al. [31], the execution environment, the parameter selection and the computational results of the proposed matheuristic are compared to those of the PSO meta-heuristic, and the LBs of the optimal solutions [31] are discussed.

A. TEST PROBLEMS

In this study, the well-known Ji et al. [31] benchmark problem instances were employed for the performance evaluation. The benchmark problem set of Ji et al. [31] has three data sets. In the first data set, 2400 problems with 11 jobs – a fixed small number - are generated as follows. The number of machines (m) is set to two values, $m \in \{2, 3\}$. The resources consumed per unit time by machine U_i, β_i ($i = 1, \ldots, m$), and the processing speed of machine U_i, v_i ($i = 1, \ldots, m$), are created from uniform distributions U(1, 5) and U(1, 10). The speed and resource consumed per unit time of the outsourcing machine are set to 1 and 100, respectively. The processing times (p_j) are created from discrete uniform distributions U(1, 100), U(100, 200), and U(100, 800). The bound on the makespan ($\hat{C}$) is created from a discrete uniform distribution U($\sum_{j=1}^{n} p_j / \sum_{i=1}^{m} v_i, \sum_{j=1}^{n} p_j / m$). For each combination of parameters, 100 test instances are generated. Therefore, 24 ($2 \times 2 \times 2 \times 3$) combinations of parameters are used, yielding a total of 2400 test instances in the first data set of the computational experiment.

In the second data set of the experiment, 2000 problems with a large number of jobs are created as follows. The number of machines $m \in \{10, 20, 30, 40, 50\}$. The number of jobs $n \in \{100, 200, 500, 1000\}$. The processing times are created from a discrete uniform distribution U(1,100). The resource consumed per unit time by machine U_i, β_i ($i = 1, \ldots, m$), and the processing speed of machine U_i, v_i ($i = 1, \ldots, m$), are created from a uniform distribution U(1, 5). The speed and resource consumed per unit time of the outsourcing machine are set to 1 and 100, respectively. The value of $\hat{C}$ is created from a discrete uniform distribution U($\sum_{j=1}^{n} p_j / \sum_{i=1}^{m} v_i, \sum_{j=1}^{n} p_j / m$). For each combination of parameters, 100 test instances are generated. As a result, 20 (4×5) combinations of parameters are considered,

TABLE 1. The solution quality of proposed matheuristic on test problems with a small number of jobs.

				(PSO-LB)/LB		(PSO-OPT)/OPT		Phase I*		Phase II*		Phase III*		Opt
m	β	v	P	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	
2	U(1,5)	U(1,5)	U(1,100)	0.00045	0.01731	0.00000	0.00000	0.00685	0.44352	0.00000	0.00000	0.00000	0.00000	100
2	U(1,5)	U(1,5)	U(100,200)	0.00081	0.04387	0.00000	0.00000	1.38000	0.52087	0.00000	0.00000	0.00000	0.00000	100
2	U(1,5)	U(1,5)	U(100,800)	0.00106	0.03447	0.00001	0.00070	1.38000	0.15534	0.00000	0.00000	0.00000	0.00000	100
2	U(1,5)	U(1,10)	U(1,100)	0.00082	0.03664	0.00000	0.00000	1.24000	0.09217	0.00000	0.00000	0.00000	0.00000	100
2	U(1,5)	U(1,10)	U(100,200)	0.00120	0.09440	0.00000	0.00000	1.18000	0.10280	0.00000	0.00000	0.00000	0.00000	100
2	U(1,5)	U(1,10)	U(100,800)	0.00026	0.01367	0.00000	0.00000	1.19000	0.02953	0.00000	0.00000	0.00000	0.00000	100
2	U(1,10)	U(1,5)	U(1,100)	0.00154	0.05248	0.00000	0.00000	1.45000	0.05782	0.00000	0.00000	0.00000	0.00000	100
2	U(1,10)	U(1,5)	U(100,200)	0.00483	0.15461	0.00000	0.00000	1.47000	0.89213	0.00000	0.00000	0.00000	0.00000	100
2	U(1,10)	U(1,5)	U(100,800)	0.00138	0.03587	0.00000	0.00000	1.49000	0.10654	0.00000	0.00000	0.00000	0.00000	100
2	U(1,10)	U(1,10)	U(1,100)	0.00102	0.04420	0.00000	0.00000	1.13131	0.10009	0.00000	0.00000	0.00000	0.00000	100
2	U(1,10)	U(1,10)	U(100,200)	0.00129	0.08205	0.00000	0.00000	1.22000	0.09539	0.00000	0.00000	0.00000	0.00000	100
2	U(1,10)	U(1,10)	U(100,800)	0.00236	0.14808	0.00000	0.00000	1.22000	0.08431	0.00000	0.00000	0.00000	0.00000	100
3	U(1,5)	U(1,5)	U(1,100)	0.00174	0.04802	0.00000	0.00000	1.97000	0.19799	0.00001	0.00146	0.00000	0.00000	100
3	U(1,5)	U(1,5)	U(100,200)	0.00323	0.05238	0.00000	0.00000	1.89000	1.12330	0.00000	0.00005	0.00000	0.00000	100
3	U(1,5)	U(1,5)	U(100,800)	0.00172	0.05199	0.00000	0.00000	1.91000	0.26693	0.00000	0.00006	0.00000	0.00000	100
3	U(1,5)	U(1,10)	U(1,100)	0.00175	0.10261	0.00000	0.00000	1.29293	0.98867	0.00000	0.00000	0.00000	0.00000	100
3	U(1,5)	U(1,10)	U(100,200)	0.00160	0.04378	0.00000	0.00000	1.41000	0.34155	0.00000	0.00000	0.00000	0.00000	100
3	U(1,5)	U(1,10)	U(100,800)	0.00039	0.01371	0.00000	0.00000	1.34000	0.14578	0.00000	0.00000	0.00000	0.00000	100
3	U(1,10)	U(1,5)	U(1,100)	0.00228	0.06410	0.00000	0.00035	1.95000	0.10086	0.00000	0.00023	0.00000	0.00000	100
3	U(1,10)	U(1,5)	U(100,200)	0.00887	0.21462	0.00000	0.00000	1.95960	0.68869	0.00000	0.00000	0.00000	0.00000	100
3	U(1,10)	U(1,5)	U(100,800)	0.00256	0.10428	0.00000	0.00017	1.85000	0.18096	0.00000	0.00000	0.00000	0.00000	100
3	U(1,10)	U(1,10)	U(1,100)	0.00077	0.02428	0.00000	0.00000	1.41000	0.03765	0.00000	0.00009	0.00000	0.00000	100
3	U(1,10)	U(1,10)	U(100,200)	0.00460	0.12219	0.00000	0.00000	1.43000	1.19601	0.00000	0.00000	0.00000	0.00000	100
3	U(1,10)	U(1,10)	U(100,800)	0.00191	0.04345	0.00000	0.00000	1.39000	0.05617	0.00000	0.00000	0.00000	0.00000	100

Phase I* denotes the objective function value of the initial solutions.
Phase II* denotes the objective function value obtained by the SA.
Phase III* denotes the objective function value obtained by the MILP.

yielding a total of 2000 problems in the second data set of the computational experiment.

The third data set of the experiment clarifies the change in resource consumption with the value of $\hat{C}$. The test instances are created using $n = 100, m = 5$, and $\hat{C} = (\sum_{j=1}^{n} p_j/m)(b/100)$, where $b = 1, 2, \ldots, 100$ denotes the makespan budget. The processing times (p_j) are created from discrete uniform distributions U(1, 100). The resource consumed per unit time of machine U_i, β_i ($i = 1, \ldots, m$), and the processing speed of machine U_i, v_i ($i = 1, \ldots, m$), are created from continuous uniform distributions U(1, 5). The speed and resource consumed per unit time of the outsourcing machine are set to 1 and 5, respectively. For each value of the parameter b, 20 instances are randomly generated, yielding a total of 2000 (20×100) problems in the third data set of the experiment.

B. EXECUTION ENVIRONMENT AND PARAMETER SELECTION

The algorithm is coded in C and run on a PC with a 2.66GHz Intel Core2 Quad CPU Q9400 and 4 GB RAM, running Windows 10. Since the parameter settings may influence the results, the following parameter values are used to test the proposed SA algorithm; $T_0 = \{5000, 10000, 15000, 20000\}$;$\alpha = \{0.80, 0.85, 0.90, 0.95\}$; $I_{iter} = \{500, 1000, 1500, 2000\}$; $N_{non-improvement} = \{5, 10, 15, 20\}$. Based on pilot tests, the parameter values of the proposed SA algorithm were set to $T_0 = 10000$, $\alpha = 0.90$, $I_{iter} = 1000$ and $N_{non-improvement} = 20$. The maximal computational time is set to 10 s, including all the times for all computations in the three phases.

C. RESULTS AND DISCUSSION

Table 1 presents the results of the computations that were performed on the first data set. Columns 1 to 4 represent the different combinations of parameters. Columns 5 and 6 present the mean and maximum relative percentage deviations, calculated as (PSO−LB)/LB, for each combination of parameters, while columns 7 and 8 present the mean and maximum relative percentage deviations, calculated as (PSO−OPT)/OPT, for each combination of parameters. The LB was obtained by Ji et al. [31], and OPT is the optimal solution that was obtained using MILP. Columns 9-10, 11-12, and 13-14 present the mean and maximum relative percentage deviations, calculated as (Phase 1−OPT)/OPT, (Phase 2−OPT)/OPT, and (Phase 3−OPT)/OPT, for the first, second, and third phases, respectively. Table 1 reveals that the solutions that are obtained in Phase II almost equal those of PSO meta-heuristic. The final column presents the number of optimal solutions that are getting by the proposed matheuristic. As revealed in Table 1, the proposed matheuristic yields optimal solutions to all test problems with a small number of jobs.

Table 2 presents the quality of the solutions to the test problems with a large number of jobs that are obtained by PSO and the three phases of the proposed matheuristic.

TABLE 2. The solution quality of the proposed matheuristic on test problems with a large number of jobs.

		(PSO-LB)/LB		(Phase I-LB)/LB		(Phase II-LB)/LB		(Phase III-LB)/LB	
m	n	Mean	Max	Mean	Max	Mean	Max	Mean	Max
10	100	0.00039	0.00607	0.00047	0.00474	0.00045	0.00474	**0.00031***	0.00338
10	200	0.00012	0.00123	0.00012	0.00099	0.00012	0.00099	**0.00011**	0.00099
10	500	0.00006	0.00039	0.00006	0.00039	0.00006	0.00039	0.00006	0.00039
10	1000	0.00002	0.00016	0.00002	0.00016	0.00002	0.00016	0.00002	0.00016
20	100	0.00105	0.02093	**0.00096**	0.01114	**0.00096**	0.01114	**0.00059**	0.00661
20	200	0.00042	0.00602	**0.00036**	0.00498	**0.00036**	0.00498	**0.00031**	0.00303
20	500	0.00010	0.00061	0.00010	0.00052	0.00010	0.00052	0.00010	0.00052
20	1000	0.00005	0.00034	0.00006	0.00048	0.00006	0.00048	0.00005	0.00034
30	100	0.00272	0.02630	**0.00183**	0.01397	**0.00183**	0.01397	**0.00123**	0.01057
30	200	0.00068	0.00744	**0.00046**	0.00338	**0.00046**	0.00338	**0.00041**	0.00322
30	500	0.00020	0.00229	**0.00018**	0.00152	**0.00018**	0.00152	**0.00018**	0.00152
30	1000	0.00008	0.00063	0.00008	0.00063	0.00008	0.00063	0.00008	0.00063
40	100	0.00811	0.10570	**0.00290**	0.02142	**0.00290**	0.02142	**0.00218**	0.02048
40	200	0.00131	0.02232	**0.00071**	0.00609	**0.00071**	0.00609	**0.00062**	0.00571
40	500	0.00025	0.00246	**0.00022**	0.00155	**0.00022**	0.00155	**0.00021**	0.00155
40	1000	0.00010	0.00070	0.00010	0.00064	0.00010	0.00064	0.00010	0.00064
50	100	0.00766	0.16308	**0.00264**	0.01669	**0.00264**	0.01669	**0.00220**	0.01669
50	200	0.00190	0.03298	**0.00090**	0.00964	**0.00090**	0.00964	**0.00082**	0.00919
50	500	0.00044	0.00463	**0.00033**	0.00254	**0.00033**	0.00254	**0.00032**	0.00254
50	1000	0.00016	0.00205	0.00016	0.00177	0.00016	0.00177	0.00016	0.00177

*Bold value denotes the RPD is better than that of PSO.

TABLE 3. The computing time of the proposed matheuristic on test problems with a large number of jobs.

		PSO		Phase I		Phase II		Phase III	
m	n	Mean	Max	Mean	Max	Mean	Max	Mean	Max
10	100	0.83945	0.98300	0.02304	0.08600	0.08389	0.19300	0.19241	5.07600
10	200	1.75046	1.95000	0.02234	0.07400	0.15128	0.30400	0.17059	0.80100
10	500	4.88774	5.44400	0.03827	0.06000	0.35673	0.71600	0.36948	0.65300
10	1000	10.45724	11.56000	0.06788	0.09900	0.71485	1.34500	0.73036	1.07900
20	100	0.93510	1.10800	0.01788	0.05000	0.08130	0.11200	0.97615	5.15900
20	200	2.00901	2.41800	0.02529	0.09300	0.13309	0.18700	0.46643	2.60800
20	500	5.37611	6.20900	0.04272	0.09300	0.28669	0.31200	0.82333	1.78700
20	1000	11.70929	13.93100	0.06782	0.10900	0.56549	0.57800	1.66132	3.82100
30	100	1.03490	1.34200	0.01587	0.04600	0.07043	0.09300	2.26340	5.28300
30	200	2.17865	2.80800	0.02423	0.04700	0.14387	0.21900	1.19916	5.39900
30	500	6.02144	7.31600	0.04450	0.06200	0.34731	0.43900	1.43908	5.68100
30	1000	12.64781	15.86600	0.07116	0.10900	0.66055	0.85800	2.43954	4.99700
40	100	1.15531	1.52900	0.01851	0.04600	0.09288	0.12500	3.59384	5.24900
40	200	2.38855	3.18300	0.02681	0.07800	0.13358	0.21800	2.07156	5.53000
40	500	6.54459	8.51800	0.05343	0.07800	0.29199	0.31200	2.45620	6.06300
40	1000	13.74363	18.34500	0.08830	0.12400	0.58361	0.84300	3.49927	7.11800
50	100	1.21654	1.65400	0.02187	0.04600	0.09511	0.12500	4.58622	5.41100
50	200	2.63671	3.49500	0.03295	0.04600	0.16892	0.23400	3.30239	5.63000
50	500	6.99731	9.68800	0.04773	0.09300	0.36981	0.48400	3.22061	6.22800
50	1000	14.76697	19.81200	0.08166	0.10900	0.57025	0.59300	4.73032	7.45400

Columns 1 and 2 in Table 2 present the number of machines and the number of jobs for various combinations of parameters. Columns 3-4 present the mean and the maximum relative percentage deviations, calculated as (PSO−LB)/LB, for each combination of m and n. Columns 5-6, Columns 7-8, and Columns 9-10 provide corresponding information for

solutions that were obtained in Phase 1, Phase 2, and Phase3, respectively. As revealed in Table 2, all of the solutions that were obtained in the three phases are better or equal than those obtained using PSO and very close to the tight LB.

Table 3 presents the computational time (in seconds) of PSO and the three phases of the proposed matheuristic for the test problems with a large number of jobs. Columns 1 and 2 in Table 3 present the numbers of machines and jobs for various combinations of parameters. Columns 3-4 present the mean and maximum computational times, obtained by PSO, for each combination of m and n. Columns 5-6, Columns 7-8, and Columns 9-10 present corresponding information obtained in Phase 1, Phase 2, and Phase3, respectively. As revealed in Table 3, the average computational times of the three phases of the proposed matheuristic are all shorter than that of PSO. In summary, Phase I, Phase II, and Phase III of the proposed matheuristic find solutions better than those found by PSO and is close to the tight LB, in less computational time. The computational efficiencies could not be directly compared because it vary with the hardware, software and programming skill. However, the average computational times, presented in Table 3, demonstrated that the proposed matheuristic outperformed the PSO meta-heuristic within a reasonable computing time.

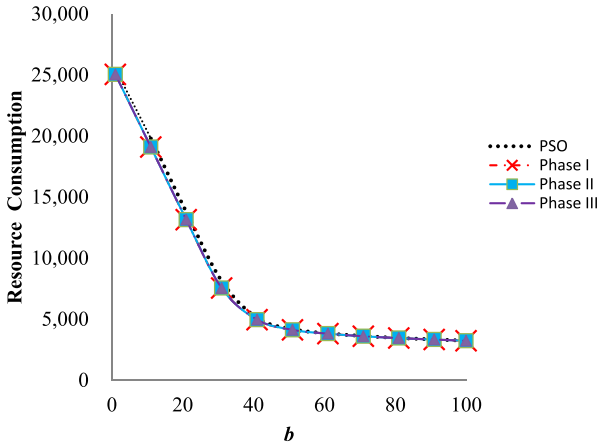

FIGURE 3. The resource consumption versus the bound of makespan.

Figures 3 and 4 present the computational results obtained using the third data set. Figure 3 plots b versus the TRC, while Fig. 4 plots b versus computational time. As shown in Fig. 3, the total resource consumption decreased with b when PSO, Phase I, Phase II, or Phase III is used to solve the problem. Since a larger b represents a larger makespan budget, a more efficient machine can process more jobs, yielding a lower value of the total resource consumption. This result is in agreement with the findings of Ji et al. [31]. The TRC that is used by Phase I, Phase II, or Phase III of the proposed matheuristic is slightly lower than that used by PSO. Figure 4 reveals that the computational time that is required for the PSO meta-heuristic decreases as b increases. Interestingly, the computational time that is required by Phase I,

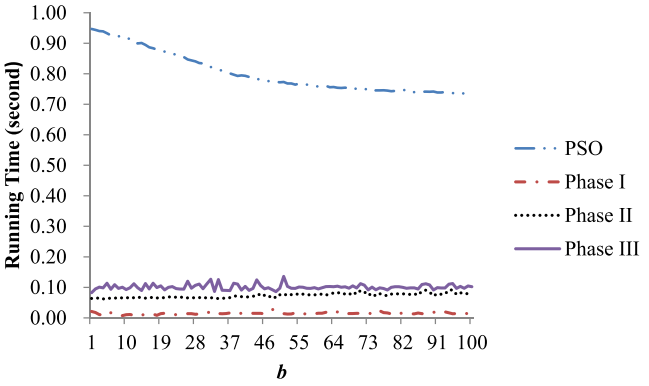

FIGURE 4. The executive time of the algorithms versus the bound of makespan.

Phase II, or Phase III varies only little with b, indicating the proposed matheuristic is not greatly affected by the makespan budget. Additionally, as revealed in Fig. 4, the computational times that are used by Phase I, Phase II, and Phase III of the proposed matheuristic are shorter than that used by PSO. In summary, the Phase I, Phase II, and Phase III of the proposed matheuristic require less computational time to find better solutions than found by PSO, and those solutions are very close to the tight LB, when the third data set is used. This fact confirms again that the proposed matheuristic outperforms the state-of-the-art PSO meta-heuristic.

V. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE RESEARCH

The $Q|C_{\max} \leq \hat{C}|TRC$ problem is an important production scheduling problem for industries that consume very large amounts of natural resources. This work develops a matheuristic for solving this strongly NP-hard problem, subject to the desired makespan, reducing the excess use of natural resources. The computational results reveal that the proposed matheuristic outperforms the best algorithm to date and the solution is very close to the tight LB. Since little theoretical progress has been made on this problem, this work represents a significant development that should encourage the application of the proposed matheuristic to real-world $Q|C_{\max} \leq \hat{C}|TRC$ problem.

Given the little available research on the $Q|C_{\max} \leq \hat{C}|TRC$ problem, much research is yet to be done on scheduling problems that are associated with environmental protection. First, additional meta-heuristic algorithms for solving the $Q|C_{\max} \leq \hat{C}|TRC$ problem are worthy of research. Second, the proposed matheuristic can be applied to solve the UPMSP whose objective is to minimize the total resource consumption with a bound on other significant performance metrics. Third, the UPMSP in which the objective is to minimize the makespan with a bound on the resource consumption warrants further study. Fourth, mathematical programming methods to find optimal solutions of the $Q|C_{\max} \leq \hat{C}|TRC$ problem should be pursued, given that the problem is NP-hard in the strong sense. Finally, theoretical research

should go beyond the static $Q|C_{\max} \leq \hat{C}|TRC$ problem to address corresponding dynamic problems.

REFERENCES

[1] 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," *Int. J. Adv. Manuf. Technol.*, vol. 53, nos. 1–4, pp. 353–361, 2011.

[2] R. McNaughton, "Scheduling with deadlines and loss functions," *Manage. Sci.*, vol. 6, no. 3, pp. 4–12, 1959.

[3] T. C. E. Cheng and C. C. S. Sin, "A state-of-the-art review of parallel-machine scheduling research," *Eur. J. Oper. Res.*, vol. 47, no. 3, pp. 292–294, 1990.

[4] R. L. Bulfin and R. G. Parker, "Scheduling jobs on two facilities to minimize makespan," *Manage. Sci.*, vol. 26, no. 3, pp. 202–214, 1980.

[5] R. Panneerselvam and S. Kanagalingam, "Modelling parallel processors with different processing speeds of single machine scheduling problem to minimize makespan," *Ind. Eng. J.*, vol. 17, no. 3, pp. 4–19, 1998.

[6] M. Azizoglu and O. Kirca, "On the minimization of total weighted flow time with identical and uniform parallel machines," *Eur. J. Oper. Res.*, vol. 113, no. 3, pp. 100–104, 1999.

[7] C.-J. Liao and C.-H. Lin, "Makespan minimization for two uniform parallel machines," *Int. J. Prod. Econ.*, vol. 84, no. 2, pp. 205–214, 2003.

[8] C.-H. Lin and C.-J. Liao, "Makespan minimization for multiple uniform machines," *Comput. Ind. Eng.*, vol. 54, no. 3, pp. 992–994, 2008.

[9] Y.-K. Lin, "Fast LP models and algorithms for identical jobs on uniform parallel machines," *Appl. Math. Model.*, vol. 37, no. 3, pp. 3436–3448, 2013.

[10] J. Zou, Y. Zhang, and C. Miao, "Uniform parallel-machine scheduling with time dependent processing times," *J. Oper. Res. Soc. China*, vol. 1, no. 2, pp. 239–252, 2013.

[11] Q. Zhang and C. X. Luo, "Uniform parallel-machine scheduling with deteriorating jobs and rejection," *Appl. Mechan. Mater.*, vols. 644–650, pp. 2030–2033, Sep. 2014.

[12] D. K. Friesen and M. A. Langston, "Bounds for multifit scheduling on uniform processors," *SIAM J. Comput.*, vol. 12, no. 3, pp. 70–74, 1983.

[13] G. Dobson, "Scheduling independent tasks on uniform processors," *SIAM J. Comput.*, vol. 13, no. 3, pp. 4–716, 1984.

[14] D. K. Friesen, "Tighter bounds for LPT scheduling on uniform processors," *SIAM J. Comput.*, vol. 16, no. 3, pp. 560–564, 1987.

[15] C. Bo, "Parametric bounds for LPT scheduling on uniform processors," *Acta Math. Appl. Sinica*, vol. 7, no. 1, pp. 67–73, 1991.

[16] P. Mireault, J. B. Orlin, and R. V. Vohra, "A parametric worst case analysis of the LPT heuristic for two uniform machines," *Oper. Res.*, vol. 45, no. 3, pp. 116–125, 1997.

[17] R. E. Burkard and Y. He, "A note on MULTIFIT scheduling for uniform machines," *Computing*, vol. 61, no. 3, pp. 277–283, 1998.

[18] C. Koulamas and G. J. Kyparisis, "Scheduling on uniform parallel machines to minimize maximum lateness," *Oper. Res. Lett.*, vol. 26, no. 4, pp. 175–179, 2000.

[19] P. De and T. E. Morton, "Scheduling to minimize makespan on unequal parallel processors," *Decision Sci.*, vol. 11, no. 4, pp. 586–602, 1980.

[20] D. S. Hochbaum and D. B. Shmoys, "A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach," *SIAM J. Comput.*, vol. 17, no. 3, pp. 539–551, 1988.

[21] R. E. Burkard, Y. He, and H. Kellerer, "A linear compound algorithm for uniform machine scheduling," *Computing*, vol. 61, no. 1, pp. 1–9, 1998.

[22] C. Koulamas and G. J. Kyparisis, "Makespan minimization on uniform parallel machines with release times," *Eur. J. Oper. Res.*, vol. 157, no. 1, pp. 262–266, 2004.

[23] A. Agarwal, S. Colak, V. S. Jacob, and H. Pirkul, "Heuristics and augmented neural networks for task scheduling with non-identical machines," *Eur. J. Oper. Res.*, vol. 175, no. 1, pp. 296–317, 2006.

[24] K. Li and B.-Y. Cheng, "Variable neighborhood search for uniform parallel machine makespan scheduling problem with release dates," in *Proc. Int. Symp. Comput. Intell. Des. (ISCID)*, 2010, pp. 43–46.

[25] D. Elvikis, H. W. Hamacher, and V. T'kindt, "Scheduling two agents on uniform parallel machines with makespan and cost functions," *J. Scheduling*, vol. 14, no. 3, pp. 471–481, 2011.

[26] P. Senthilkumar and S. Narayanan, "Simulated annealing algorithm to ninimize makespanin single machine scheduling problem with uniform parallel machines," *Intell. Inf. Manage.*, vol. 3, no. 1, pp. 22–31, 2011.

[27] P. Senthilkumar and S. Narayanan, "GA based heuristic to minimize makespan in single machine scheduling problem with iniform parallel machines," *Intell. Inf. Manage.*, vol. 3, pp. 204–214, Sep. 2011.

[28] W.-C. Lee, M.-C. Chuang, and W.-C. Yeh, "Uniform parallel-machine scheduling to minimize makespan with position-based learning curves," *Comput. Ind. Eng.*, vol. 63, no. 3, pp. 814–818, 2012.

[29] Y.-K. Lin, "Scheduling identical jobs on uniform parallel machines under position-based learning effects," *Arabian J. Sci. Eng.*, vol. 39, no. 8, pp. 6567–6574, 2014.

[30] P. Senthilkumar and S. Narayanan, "Literature review of single machine scheduling problem with uniform parallel machines," *Intell. Inf. Manage.*, vol. 2, p. 457, Aug. 2010.

[31] 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, Dec. 2013.

[32] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: A survey," *Ann. Discrete Math.*, vol. 5, pp. 287–326, Aug. 1979.

SHIH-WEI LIN received the bachelor's, master's, and Ph.D. degrees in industrial management from the National Taiwan University of Science and Technology, Taiwan, in 1996, 1998, and 2000, respectively. He is currently a Professor with the Department of Information Management, Chang Gung University, Taiwan. He is also with the Department of Neurology, Linkou Chang Gung Memorial Hospital, Taoyuan, Taiwan, and with the Department of Industrial Engineering and Management, Ming Chi University of Technology, Taipei, Taiwan. His papers have appeared in *Computers and Operations Research*, the *European Journal of Operational Research*, the *Journal of the Operational Research Society*, the *European Journal of Industrial Engineering*, the *International Journal of Production Research*, the *International Journal of Advanced Manufacturing Technology*, *Knowledge and Information Systems*, *Applied Soft Computing*, *Applied Intelligence*, and *Expert Systems with Applications*. His current research interests include meta-heuristics and data mining.

KUO-CHING YING joined the National Taipei University of Technology in 2009, where he is currently a Distinguished Professor with the Department of Industrial Engineering and Management. He is the author or co-author of over 100 academic papers, some of which have been published in international journals, such as *Applied Intelligence*, *Applied Soft Computing*, *Computers and Operations Research*, *Computers and Industrial Engineering*, the *European Journal of Industrial Engineering*, the *European Journal of Operational Research*, the *International Journal of Production Economics*, the *International Journal of Advanced Manufacturing Technology*, the *International Journal of Innovative Computing, Information and Control*, the *International Journal of Production Research*, the *Journal of the Operational Research Society*, *OMEGA-The International Journal of Management Sciences*, *Production Planning and Control*, and *Transportation Research Part E: Logistics and Transport Review*, among others. His research is focused on the operations scheduling.