IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 13, 2018, accepted March 14, 2018, date of publication March 19, 2018, date of current version April 18, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2816959

Decreasing the System Testing Makespan in a Computer Manufacturing Company

SHIH-WEI LIN[1,2,3], CHIEN-YI HUANG[4], KUO-CHING YING[4], AND DAR-LUN CHEN[5]

[1]Department of Information Management, Chang Gung University, Taoyuan 33302, Taiwan
[2]Department of Neurology, Linkuo Chang Gung Memorial Hospital, Taoyuan 33305, Taiwan
[3]Department of Industrial Engineering and Management, Min Chi University of Technology, Taipei 24301, Taiwan
[4]Department of Industrial Engineering and Management, National Taipei University of Technology, Taipei 10608, Taiwan
[5]Department of Quality Assurance, Quanta Computer, Taoyuan 33377, 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, Taiwan, under Grant MOST105-2410-H-182-009-MY2/MOST106-2632-H-182-001, and in part by the Linkou Chang Gung Memorial Hospital under Grant CMRPD3G0011/CARPD3B0012. The work of K.-C. Ying was supported by the Ministry of Science and Technology under Grant MOST106-2221-E-027-085.

ABSTRACT This paper investigates the system testing scheduling problem (STSP), using one of the largest computer manufacturing companies in the world as a case study. A mixed integer linear programming (MILP) model and a restricted simulated annealing (RSA) heuristic which applies two rules to eliminate ineffective job moves to minimize makespan in the STSP are presented. The proposed RSA is empirically evaluated using 188 simulation instances derived from the characteristics of a real technology company. The RSA computational results are compared with those of the traditional simulated annealing (SA) and the artificial bee colony (ABC) algorithms. The statistical results demonstrate that the RSA, SA, and ABC provide much better solutions than the MILP model solved using Gurobi solver for small problems within a reasonable execution time. The RSA offers significant improvements over the SA and ABC algorithms when applied to large problems. The simulation results demonstrate that the proposed RSA heuristic significantly decreases system testing makespan in a computer manufacturing plant at Taiwan.

INDEX TERMS System testing, scheduling, parallel machines.

I. INTRODUCTION

The rapid development of Taiwan's electronics industry began in the 1980s, when companies first started producing personal computers (PCs), which became one of the principal driving elements in the development of essentially all modern technology. In the 2010s, PC products included desktop computers, laptop (or notebook) computers and tablet computers. The manufacturers' excellent research and design (R&D), mass production, logistics and service made Taiwan one of the world's largest PC production centers. In order to enhance competitiveness, PC manufacturers must deliver targeted specifications, achieve cost targets within a set development timetable and realize high-quality products [1]. In the product development phase, system testing is one of the most important aspects in the production of high-quality PCs.

System testing is actually a series of different tests which exercise a full computer-based system. In practice, developing a better schedule to minimize the makespan of the system testing process is the key to improving production performance, shortening the product testing cycle, and even reducing manufacturing costs [2], [3]. The system testing scheduling problem (STSP) considered in this study is a variant of minimizing the makespan of the identical parallel machine scheduling problem (IPMSP) under resource constraints, where each PC represents a machine and each test item corresponds to a job. Since the classic IPMSP is strongly NP-hard, the STSP is clearly strongly NP-hard. Given that IPMSPs arise in a broad range of industrial applications in both manufacturing and service sectors, many researchers and practitioners have addressed it in recent decades [4]–[6]. Classical IPMSPs are commonly solved using exact methods and heuristic algorithms. Heuristic algorithms found in the literature for solving IPMSPs can further be decomposed into two subclasses: constructive heuristics and improvement heuristics. Dispatching rules are examples of construction heuristics, which are simple rules that prioritize all jobs waiting for processing on a machine. Improvement heuristics usually begin with an initial schedule and then iteratively try to find a better schedule. Due to the computational com-

plexity of solving IPMSPs, exact methods in the literature, such as dynamic programming [7], [8], branch-and-bound (B&B) algorithms [9], [10], and mixed integer programming models [11] can only be used to solve very small problems.

In order to meet practical requirements, some priority rules are simply applied with very low computational complexity. Frequently used priority rules found in the literature for solving IPMSPs include the Shortest Processing Time rule (SPT), the Preemptive Shortest Processing Time rule (PSPT), the Weighted Shortest Processing Time rule (WSPT), the Earliest Due Date rule (EDD), the Longest Processing Time rule (LPT), the Critical Path rule (CP), the Largest Number of Successors rule (LNS), the Longest Remaining Processing Time First rule (LRPT), the Longest Remaining Processing Time First-Fastest Machine rule (LRPT-FM), and the Shortest Remaining Processing Time rule (SRPT) [12]. These priority rules, though simple, do not consistently yield optimal results. Therefore, most relevant works focus on the development of heuristic algorithms, which are simple to apply and can be very effective.

The most famous improvement heuristics include a family of rolling horizon procedures, the Tabu search algorithm, the pseudo-polynomial algorithm, the simulated annealing (SA) algorithm, the variable neighborhood search method, the greedy randomized adaptive search procedure and the approximation algorithm [13]. In these improvement heuristics, metaheuristics are the main alternative method of solving IPMSPs. Metaheuristics are upper level general templates that can be used as guiding strategies in designing underlying heuristics to provide ''acceptable'' solutions in a reasonable time for specific hard and complex problems [14]. In addition to being easily understood, metaheuristics also have a greater potential to be employed effectively; they are therefore state-of-the-art methods for solving medium- and large-sized IPMSPs. Since metaheuristics are very simple, fast, robust and flexible for solving large optimization problems [15], this study also designs and implements a metaheuristic to solve the practical STSP. A detailed discussion of heuristic algorithms for solving IPMSPs can be found in the excellent survey conducted by Kaabi and Harrath [12], Mokotoff [16], and Nandagopal *et al.* [17].

This study investigates the STSP under constraints on the number of workforces and machines, using a computer manufacturing plant in Taiwan as a case study. This computer manufacturing plant uses the First Come, First Served (FCFS) method to schedule all test items that involve manual testing and automated testing, which is implemented as follows. Given a sequence of test items, the tests of those items are sequentially scheduled using the machine with the shortest completion time for the currently assigned test items. However, the FCFS method typically generates a schedule that takes a long time to perform the system testing, which is a priority for improvement. To minimize the makespan of the addressed STSP, this study develops a mixed integer linear programming (MILP) model, and a restricted simulated annealing (RSA) heuristic which is applied two rules to eliminate ineffective job moves.

The remainder of this paper is arranged as follows. First, the system testing schedule problem is outlined and formulated. Second, the proposed RSA heuristic is described. Third, the proposed algorithm is empirically evaluated using 188 simulation instances derived from the characteristics of a real PC production company in Taiwan, and its performance is compared with that of the MILP model, simulated annealing (SA), and artificial bee colony (ABC) algorithms. Finally, conclusions are drawn and recommendations for future research are provided.

II. PROBLEM DESCRIPTION

A traditional method used in the development of PCs consists of five steps (see Fig. 1) as follows:

(1) A KICK OFF meeting is held and the product specifications are finalized. All required people and equipment should be identified. Relevant personnel participate in the project meeting to discuss the introduction of the product, the production schedule, and the assignment of tasks and work with departments.

(2) The Engineering Verification Test (EVT) takes six to eight weeks, during which period logic design, layout design and mechanical design are performed, and electronic components are selected. A product verification test is conducted on prototypes to confirm that the design satisfies the desired product specifications and performance requirements. The EVT is composed of basic functional tests, specification verification and parametric measurements.

(3) The Design Verification Test (DVT) involves intensive logic and layout modifications. This intensive testing program delivers objective, comprehensive results that verify all product specifications, original equipment manufacturer requirements, interface standards and diagnostic commands.

(4) The Production Verification Test (PVT) is performed when the product moves to the production phase. The PVT focuses on pre-production or production units. The purpose of the PVT is to confirm that the design has been correctly produced. Approximately 100 units are fabricated for an experiment [18]. The standard operation procedure is drawn up.

(5) Mass Production (MP) is the final phase of the project. The contract with the customer determines whether the manufacturer configures products and packages them, or ships them as semi-finished products to be configured and packaged by the customer [18].

The entire process from EVT to MP takes a very long time, and each stage involves lengthy system testing to ensure product quality. In these stages, the testing of software or hardware is performed using a complete, integrated system, to evaluate the extent to which the system meets its requirements. When errors are found, corrective action must be taken.

FIGURE 1. The procedure of computerized product development.

The STSP considered herein can be formulated as a more complex variation of the IPMSP with resource constraints. In this study, each **TM** is a test machine, each test item corresponds to a job, and the availability of testers and/or test machines represents resource constraints. To formulate the STSP with resource constraints addressed in this study, consider a set $\mathbf{J} = \{1, 2, \ldots, n\}$ of n given test items (jobs) that must be carried out on a set $\mathbf{TM} = \{TM_1, TM_2, \ldots, TM_q\}$ of q identical **TM**s. The test items involve manual testing and automated testing, following the sequence of steps in a written testing proposal, that is $\mathbf{J} = \mathbf{J}_a \cup \mathbf{J}_m$, where $\mathbf{J}_a$ is the set of automated testing items, and $\mathbf{J}_m$ represents the set of manual testing items. The heart of the problem is that the manual testing is executed by one tester and one machine, and must be carried out during working hours, and therefore only on weekdays; the automated testing is performed by one machine only whenever it is available. Any available **TM**s can carry out each test item $j \in \mathbf{J}$ with an identified testing time P_j. The setup time is independent of the sequence of the test item, and is included in the processing time. The lower bound for the number of jobs assigned to a machine is zero. The upper bound for the number of jobs assigned to a machine is restricted by two constraints on resource consumption as follows: Automated testing items can be tested in any period of 24 hours if the required test machines are vacant; Manual testing items can only be tested when the testers are available during working hours. Each TM can test just one test item at a time and is persistently available to deal with all scheduled test items when required. The available working time of each member of staff (manual testing periods) is the same and known in advance. The testing times of test items are non-negative values that are known in advance. Moreover, the STSP considered in this research satisfies the following two assumptions:

(1) Each test item can be tested by exactly one of the test machines.

(2) All test items are available for testing at the beginning of the scheduling period.

The objective is to obtain a feasible testing schedule $\Pi = \{\pi_1, \pi_2, \ldots, \pi_q\}$ for all test items which minimizes the makespan, $C_{\max}$, of the n test items, where $\pi_k (k = 1, 2, \ldots, q)$ is the sequence of test items on TM_k.

III. MILP MODEL

This section presents a formulation of the STSP of interest. The MILP model is constructed herein using a case company. The proposed model can be modified to meet the needs of other companies as required. To simplify this formulation, the following notation is used:

Indices

n : Number of test items
q : Number of TMs
r : Number of manual testing periods
i, j: Index of test item, $i, j \in \{1, 2, \ldots, n\}$
k : Index of machine, $k \in \{1, 2, \ldots, q\}$
t : Index of manual testing period, $t \in \{1, 2, \ldots, r\}$
$\mathbf{J}_a$: Set of automated testing items
$\mathbf{J}_m$: Set of manual testing items

Parameters

O_t : Start time of manual testing period t
W_t : End time of manual testing period t
P_j : Time required to test item j
V : A very large positive number

Decision Variables

$A_{ijk} = 1$, if test item j is processed immediately after test item i on TM_k; 0, otherwise
$A_{0jk} = 1$, if test item j is the first item processed on TM_k; = 0, otherwise
$x_{jt} = 1$, if test item j is processed in manual period t; 0, otherwise
C_j: Completion time of test item j (real variable)
$C_{\max}$: Maximal completion time (makespan) of last test item (real variable)

Let job 0 be a dummy initial job. Now the STSP can be formulated as follows:

$$\text{Min } C_{\max}$$

Subject to

$$\sum_{\substack{i=0 \\ i \neq j}}^{n} \sum_{k=1}^{q} A_{ijk} = 1, \quad \forall j = 1, \ldots, n \tag{1}$$

$$\sum_{\substack{i=0 \\ i \neq h}}^{n} A_{ihk} \geq \sum_{\substack{j=1 \\ j \neq h}}^{n} A_{hjk}, \quad \forall h = 1, \ldots, n, \ \forall k = 1, \ldots, q \tag{2}$$

$$C_j \geq C_i + \sum_{k=1}^{q} A_{ijk} P_j + V \left(\sum_{k=1}^{q} A_{ijk} - 1 \right), \quad \forall i = 0, \ldots, n, \ \forall j = 1, \ldots, n \tag{3}$$

$$\sum_{t=1}^{r} O_t X_{jt} + P_j \leq C_j \leq \sum_{t=1}^{r} W_t X_{jt}, j \in \mathbf{J}_m \tag{4}$$

$$\sum_{t=1}^{r} X_{jt} = 1, \quad j \in \mathbf{J}_m \tag{5}$$

$$\sum_{j=1}^{n} A_{0jk} \leq 1, \quad \forall k = 1, \ldots, q \tag{6}$$

$$C_0 = 0 \tag{7}$$

$$C_{max} \geq C_j, \quad \forall j = 1, \ldots, n \tag{8}$$

$$C_j \geq 0, \forall j = 1, \ldots, n \tag{9}$$

$$A_{ijk} = \{0, 1\}, \quad \forall i = 0, \ldots, n, \quad \forall j = 1, \ldots, n, \forall k = 1, \ldots, q \tag{10}$$

$$X_{jt} = \{0, 1\}, \; j \in \mathbf{J}_m, \quad \forall t = 1, \ldots, r, \tag{11}$$

The objective function is to minimize the makespan. Constraint set (1) ensures that each test item is tested on only one machine. Constraint set (2) ensures each item is tested after one test item (including a dummy test item). Constraint set (3) defines the relations between the completion times of test items j and i that are dependent on whether test item j is processed immediately after test item i on the same machine. For Constraint set (4) the manual testing item will be processed after (or on) the starting time and finished before (or on) the one manual testing period. Constraint set (5) ensures that each manual testing is performed in only one manual testing period. Constraint set (6) ensures that at most one item is the first to be processed on each machine. Constraint set (7) defines the completion time of the testing of the dummy test item as zero. Constraint set (8) defines the makespan. Constraint set (9) specifies the completion time of each job as a non-negative real number. Finally, constraint sets (10) and (11) define the domain of the binary variables. In addition, Assumptions (1) and (2) of the addressed STSP correspond to Constraint sets (1) and (9), respectively.

IV. PROPOSED RESTRICTED SIMULATED ANNEALING ALGORITHM

This section describes the representation of the solution and the calculation of the objective function value for the RSA heuristic, and discusses other parts of the RSA used herein to solve the STSP.

A. BASIC SIMULATED ANNEALING HEURISTIC

The SA heuristic is a commonly used meta-heuristic for solving hard combinatorial optimization problems. SA mimics the annealing process in metallurgy. Annealing is the procedure by which rapid cooling results in poor crystallization, whereas slow cooling of metal results in a uniform and good crystallization. By analogy, the SA heuristic is an optimization procedure that reaches a (near-) global minimum by mimicking the crystallization cooling procedure [14], [15], [19].

In general, the SA heuristic generates an initial solution as a current solution X by random or dispatch rule(s). Then, in the following iteration, the SA heuristic randomly chooses a candidate solution Y from the preset neighborhood solution of X. The objective function value of Y is then compared with that of X. If the objective function value of Y is less than that of X, then it will replace X, after which the search processes will be continued. In order to escape from a local minimum, SA allows a worsening candidate as X, based on a probabilistic criterion derived from the Boltzmann function and controlled by a gradually reduced temperature. Typically, this step is repeated until one of the termination conditions is met.

B. RULES TO ELIMINATE INEFFECTIVE JOB MOVES

In the basic SA heuristic, the set of solutions for neighbor X, $N(X)$, is generated using a swap operation and an insertion operation. That is, $N(X)$ is sampled by randomly choosing two jobs (test items), which are exchanged with each other, or by randomly picking one job and placing it back immediately before another randomly selected job. However, many of the random job moves yield the same objective function value, and these are called ineffective job moves or non-effective job moves [20].

The following two rules can be used to eliminate the ineffective moves in order to increase the search speed in this study; (1) if there are exactly two bottleneck machines, only exchanging one test item from each of them has the probability to improve the current solution, (2) if there is exactly one bottleneck machine, then only exchanging one test item from it has the probability to improve the current solution. However, if there are more than two bottleneck machines, then not exchanging two test items can improve the current solution. Therefore, one test item on any bottleneck machine is chosen, and the second test item is chosen from any one machine [21].

When the most recent test item on a bottleneck machine has been identified, many ineffective job moves can be recognized and excluded from the neighborhood. To limit the generation of unnecessary neighborhood solutions, the neighborhood generation of the SA heuristic is revised by applying the two above rules into the swapping and insertion operations.

C. REPRESENTATION OF SOLUTION AND CALCULATION OF VALUE OF OBJECTIVE FUNCTION

In this work, a solution is represented as a sequence of numbers that comprises a permutation of n test items and $m - 1$ zeros, which split the n test items into m parts. For instance, the solution representation in Figure 2 is interpreted as follows: Twenty test items are to be tested on two machines. The test items are processed on machines 1 and 2 in sequences 4-16-19-13-12-3-1-11-7-10 and 18-17-14-8-15-6-5-20-2-9, respectively.

When a test item is scheduled and no machine is available, it must wait until at least one machine becomes available. If a manual testing item is scheduled to be tested before the manual testing period begins, then the process is delayed until the manual testing period begins. If a manual test cannot be completed before the end of a manual testing period, then it is delayed until a new manual testing period begins.

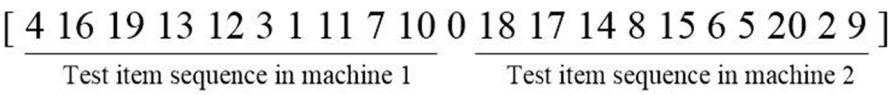

FIGURE 2. An example of solution representation.

D. PARAMETERS USED

The proposed RSA heuristic involves four parameters: I_{iter}, T_0, $N_{non\text{-}improving}$ and α. I_{iter} represents the number of iterations of the search at a specific temperature. T_0 represents the initial temperature. $N_{non\text{-}improving}$ is the maximum admissible number of temperature reductions in which the best objective function value has not improved. Finally, α is a coefficient that controls the cooling schedule.

E. PROCEDURE OF PROPOSED RSA HEURISTIC

The proposed RSA heuristic is the traditional SA plus the two rules introduced in Section IV.B, which is implemented as follows:

(1) Parameter setting: Set the initial temperature, the cooling rate, the number of iterations at a particular temperature, and the termination condition.
(2) Initial state configuration: Produce initial solution randomly and calculate the objective function value of the initial solution.
(3) Neighborhood solution generation: Generate a neighborhood solution as discussed in Section IV.B. and compute the objective function value of a neighborhood solution.
(4) Acceptance test: Let $\Delta = obj(Y) - obj(X)$. If Δ is not less than zero, then X is replaced with Y. Otherwise, the probability of replacing X with Y is $\exp(-\Delta/T)$. X_{best} and F_{best} are the best solution and its objective function value obtained so far.
(4) Temperature reduction check: The current temperature T is reduced I_{iter} iterations after the prior temperature reduction, using the formula $T = \alpha T$, $0 < \alpha < 1$.
(6) Termination condition check: The search procedure is terminated when the current best solution X_{best} has not improved for $N_{non\text{-}improving}$ successive temperature reductions. If the termination condition is not satisfied, then repeat Steps (3) to (6). Following the termination of the search procedure, the best solution is given by X_{best}.

V. EXPERIMENT RESULTS AND DISCUSSION

The proposed RSA heuristic was coded in C and tested on a computer with an Intel Core 2 2.67 GHz CPU. The MILP model for the STSP was solved using Gurobi 7.0 on the same machine.

A. TEST PROBLEMS

To demonstrate the applicability of the RSA heuristic to real situations, 188 real instances obtained from a computer manufacturing plant were used. Based on the real data from this case company, the processing time p_j ($j = 1, \ldots, n$) is an integer generated from the uniform distribution [7, 3840]. To confirm the proposed MILP and to compare it with RSA and SA, another 50 small problem instances were used. Nineteen problem instances were used to calibrate the parameters. One hundred and twenty large problem instances were used to compare the RSA, SA and ABC algorithms. The ABC used in the comparison is based on one of the best-performing existing algorithms for a similar parallel machine scheduling problem proposed by Lin and Ying [21].

For the 19 problem instances used in parameter calibration, the number of test items ranged from 20 to 200. The numbers of machines in these calibration problem instances were two, five and ten. One number of test items (N) corresponds to one problem.

For small problems, the number of test items (N) had five values: 10, 15, 20, 25 and 30, with two machines. For each number of test items, ten instances were randomly generated, yielding a total of 50 test instances.

For large problems, the number of test items had four values, which were 100, 150, 200 and 250, and the number of machines (M) had three values, which were two, five and ten. For each combination of N and M, ten instances were randomly selected. Therefore, 120 instances of the large problem were used.

B. PARAMETER CALIBRATION

The four key parameters of the proposed RSA include the initial temperature (T_0), the number of iterations at a particular temperature (I_{iter}), the cooling rate (α), and the maximal admissible number of temperature reductions in which the best objective function has not improved ($N_{non-improving}$). To determine appropriate values for them, the Taguchi method of design of experiment (DOE) [22] is employed using 19 randomly selected instances. Since each parameter had four levels (see Table 1), the orthogonal array L16(44) was applied. As shown in Table 2, the number of parameter combinations was 16. For each parameter combination, the RSA was run independently 30 times for each of the 19 randomly selected instances. The makespan of the 30 trials for each test instance were recorded and the relative percentage deviation (RPD) was calculated as $RPD = (C^P_{\max} - C^{BKS}_{\max})/C^{BKS}_{\max} \times 100\%$, where $C^P_{\max}$ was the average makespan of the 30 trails obtained using the parameter combination P, and $C^{BKS}_{\max}$ was the minimum makespan among 30 trials obtained using all parameter combinations.

The average RPDs for 19 randomly selected instances obtained using each parameter combination are shown in Table 3 as the response variable. The significance value of

TABLE 1. Levels of the parameters for the RSA.

Parameters	Parameter Level			
	1	2	3	4
T_0	0.5	1.5	2.5	3.5
α	0.93	0.95	0.97	0.99
I_{iter}	2000L*	3000L	4000L	5000L
$N_{non\text{-}improving}$	5	10	15	20

*: L denotes the length of the solution representation

TABLE 2. Orthogonal array and relative percent of deviation (*RPD*).

Experiment No.	T_0	α	I_{iter}	$N_{non\text{-}improving}$	*RPD*	CPU Time (s)
1	0.5	0.93	2000L	5	0.616	4.05
2	0.5	0.95	3000L	10	0.343	11.06
3	0.5	0.97	4000L	15	0.229	21.35
4	0.5	0.99	5000L	20	0.130	33.48
5	1.5	0.93	3000L	15	0.157	19.36
6	1.5	0.95	2000L	20	0.210	16.40
7	1.5	0.97	5000L	5	0.300	9.10
8	1.5	0.99	4000L	10	0.196	14.30
9	2.5	0.93	4000L	20	0.127	34.76
10	2.5	0.95	5000L	15	0.141	30.21
11	2.5	0.97	2000L	10	0.370	7.22
12	2.5	0.99	3000L	5	0.419	5.52
13	3.5	0.93	5000L	10	0.184	21.01
14	3.5	0.95	4000L	5	0.353	7.76
15	3.5	0.97	3000L	20	0.185	23.77
16	3.5	0.99	2000L	15	0.288	10.49

TABLE 3. The average relative percent of deviations for each parameter.

Level	T_0	α	I_{iter}	$N_{non\text{-}improving}$
1	0.3295	0.2708	0.3710	0.4219
2	0.2156	0.2617	0.2761	0.2732
3	0.2641	0.2710	0.2260	0.2036
4	0.2524	0.2582	0.1886	0.1629
Range	0.1139	0.0128	0.1823	0.2590
Rank	3	4	2	1

TABLE 4. ANOVA of parameters.

Source	DF	SS	MS	F	P
T_0	3	15.4	5.1333	9.4837	2.982e-06
α	3	0.3	0.1000	0.1776	0.9116
I_{iter}	3	42.7	14.2333	26.2533	2.2e-16
$N_{non\text{-}improving}$	3	88.7	29.5667	54.5684	2.2e-16
Error	9107	4931.9	0.5416		
Total	9119				

each parameter is also analyzed in Table 3. It can be seen that $N_{non\text{-}improving}$ is the most significant of the four parameters. That is, the more computing time used, the better the solution obtained will be. The Analysis of Variance (ANOVA) of Parameters is shown in Table 4, which reveals that T_0 and I_{iter} also have an impact on the solution quality.

Figure 3 presents the effect of each parameter on solution quality using the average relative percentage deviation (*ARPD*). As presented in Fig. 3, the influences of T_0 and α are not as apparent as those of the other two parameters. In general, if the initial temperature T_0 is too high, then an excessive computational time is required to find better solutions; If T_0 is too low, then the proposed RSA algorithm will be inclined to converge prematurely as a consequence of the low probability of accepting poor solutions; A higher α requires more time to reduce the temperature to the stopping

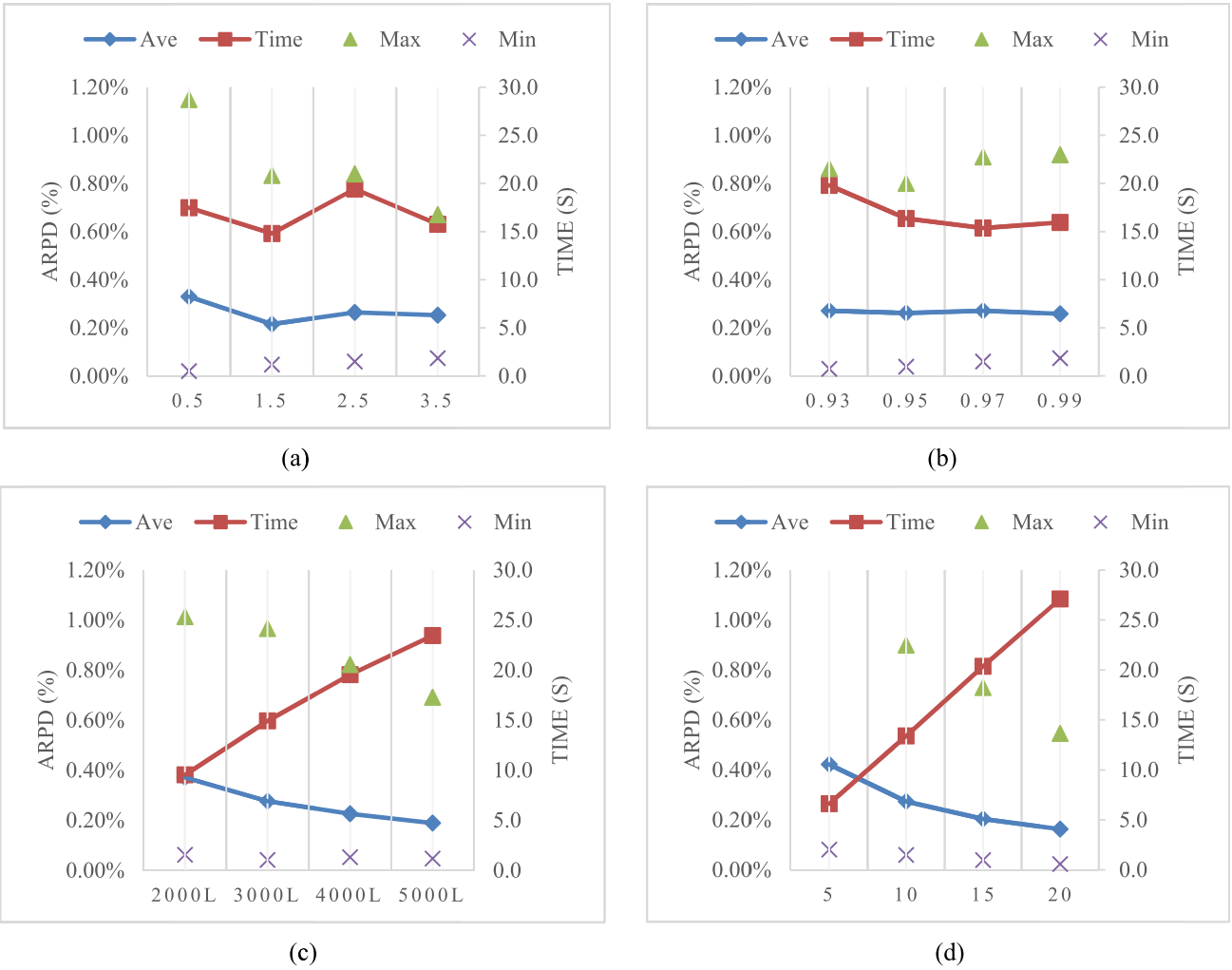

FIGURE 3. The effect of each parameter on solution quality. (a) Parameter T_0. (b) Parameter α. (c) Parameter I_{iter}. (d) Parameter $N_{non\text{-}improving}$.

value if the RSA terminates at a final temperature. However, the termination condition of the proposed RSA heuristic involves the value $N_{non\text{-}improving}$, the ranges of solution quality and computational times are not too wide for various values of T_0 and α, as presented in Figures 3(a) and 3(b). When I_{iter} was increased, better solutions were obtained at the cost of increased computational times, as shown in Fig. 3(c). As shown in Fig. 3(d), increasing $N_{non\text{-}improving}$ improved the quality of the solution at the expense of a higher computational time. As a trade-off between computational time and solution quality, $T_0 = 1.5$, $\alpha = 0.99$, $I_{iter} = 5000L$, and $N_{non-improving} = 15$ are set in RSA, where L denotes the length of the solution representation.

C. COMPARISON OF RESULTS WITH THOSE OBTAINED BY MILP USING THE GUROBI SOLVER

In the comparison of results obtained by the RSA heuristic with those obtained by MILP using the Gurobi solver, each problem is solved by both RSA, SA and ABC for only one run. Gurobi 7.0 was used to solve the model. For small problems with 15 and 20 jobs, it is difficult to solve the MILP model to optimality in a reasonable amount of time using the Gurobi solver. In the pilot experiments, it was found that the incumbent feasible solution obtained after running one hour of the Gurobi solver usually did not change for some time. Therefore, the solver was terminated after one hour if it had not obtained an optimal solution. Table 5 presents solutions to 50 small problem instances and the corresponding computational times. The Gurobi solver finds optimal solutions to ten out of 50 problems, as indicated in bold in Table 5. RSA, SA and ABC also generate optimal solutions to all ten of these problems with known optimal solutions. For the remaining 50 problems without known optimal solutions, RSA, SA and ABC find better solutions than the Gurobi solver in 35 out of 50 problems, respectively, and find solutions equal to that of Gurobi solver in the remaining 15 problems. The computational times of RSA, SA and ABC are all less than three seconds.

Many factors may influence computational time; these include CPU speed, memory size, operating system,

TABLE 5. Computional results for the small size of problem.

N	Gurobi		ABC		SA		RSA		N	Gurobi		ABC		SA		RSA	
	C_{max}	T(s)	C_{max}	T(s)	C_{max}	T(s)	C_{max}	T(s)		C_{max}	T(s)	C_{max}	T(s)	C_{max}	T(s)	C_{max}	T(s)
10	**848**	36.5	**848**	0.1	**848**	0.3	**848**	0.3	20	3465	3600.1	3460	0.7	3460	0.6	3460	0.7
10	**1582**	406.1	**1582**	0.1	**1582**	0.3	**1582**	0.3	20	5075	3600.1	4978	0.7	4978	0.6	4978	0.7
10	**1576**	39.8	**1576**	0.1	**1576**	0.3	**1576**	0.3	20	2264	3600.1	2262	0.7	2263	0.7	2262	0.6
10	**1696**	16.2	**1696**	0.1	**1696**	0.3	**1696**	0.3	20	2322	3600.1	2316	0.8	2316	0.7	2316	0.7
10	**1619**	822.8	**1619**	0.1	**1619**	0.3	**1619**	0.3	20	3171	3600.1	3167	0.8	3167	0.7	3167	0.7
10	**2106**	426.4	**2106**	0.1	**2106**	0.3	**2106**	0.3	25	6515	3600.2	6325	1.2	6325	0.8	6325	0.8
10	**2053**	123.6	**2053**	0.1	**2053**	0.3	**2053**	0.3	25	3102	3600.1	3075	1.2	3075	0.9	3076	0.9
10	**1576**	39.8	**1576**	0.1	**1576**	0.3	**1576**	0.3	25	4292	3600.1	3869	1.2	3869	0.8	3869	0.8
10	**921**	686.9	**921**	0.1	**921**	0.3	**921**	0.3	25	5028	3600.1	4801	1.3	4801	0.8	4801	0.8
10	**902**	0.4	**902**	0.1	**902**	0.3	**902**	0.3	25	5185	3600.1	4992	1.5	4993	0.8	4992	0.9
15	2215	3600.0	2215	0.3	2215	0.5	2215	0.5	25	5152	3600.1	4564	1.1	4564	0.8	4564	0.9
15	2085	3600.0	2028	0.3	2028	0.5	2028	0.5	25	4939	3600.1	4591	1.1	4591	0.8	4591	0.8
15	2220	3600.0	2215	0.4	2216	0.5	2215	0.5	25	4305	3600.1	3878	1.1	3878	0.8	3878	0.8
15	2000	3600.0	2000	0.3	2000	0.5	2000	0.5	25	3128	3600.1	3083	1.1	3083	0.8	3083	0.8
15	2004	3600.0	1998	0.3	1998	0.5	1998	0.5	25	3513	3600.1	3460	1.1	3460	0.8	3460	0.8
15	2126	3600.0	2126	0.4	2126	0.5	2126	0.5	30	5280	3600.1	5129	1.8	5129	1.0	5129	1.0
15	2263	3600.0	2262	0.3	2262	0.5	2262	0.5	30	6367	3600.1	6022	1.7	6022	0.9	6022	1.0
15	2126	3600.0	2125	0.4	2125	0.5	2125	0.5	30	16729	3600.1	8847	2.1	8846	1.1	8846	1.1
15	2183	3600.0	2183	0.4	2183	0.5	2183	0.5	30	10525	3600.1	10189	2.6	10190	1.1	10189	1.2
15	2320	3600.1	2320	0.4	2320	0.5	2320	0.5	30	5749	3600.2	5253	2.0	5253	1.0	5253	1.0
20	3034	3600.1	3019	0.7	3020	0.7	3019	0.7	30	6000	3600.1	5152	1.9	5152	1.0	5152	0.9
20	3456	3600.1	3451	0.7	3451	0.7	3451	0.7	30	3707	3600.1	3533	2.0	3535	1.1	3533	1.1
20	3682	3600.1	3521	0.7	3521	0.6	3521	0.7	30	7199	3600.1	6534	2.4	6535	1.0	6533	1.2
20	5080	3600.1	4942	0.8	4942	0.6	4942	0.6	30	18420	3600.1	9829	1.8	9829	1.0	9829	1.0
20	3661	3600.1	3639	0.7	3637	0.7	3637	0.7	30	3708	3600.1	3679	1.8	3679	1.0	3679	1.0

TABLE 6. Performace comparison for ABC, SA, AND RSA.

N	M	FCFS	ABC			SA			RSA		
		C_{max}	*RPD*	C_{max}	*RPD*	Time (s)	C_{max}	*RPD*	Time (s)	C_{max}	*RPD*	Time (s)
50	2	9352.5	11.25	8379.5	0.020	7.86	8378.0	0.002	3.61	8378.0	0.002	3.5
50	5	4732.5	15.99	4027.5	0.016	5.31	4027.6	0.020	3.49	4027.2	0.007	3.3
50	10	3742.7	21.14	3179.0	0.000	4.66	3179.0	0.000	3.37	3179.0	0.000	3.2
100	2	22652.2	8.13	21019.9	0.024	15.70	21024.1	0.044	15.29	21014.7	0.000	15.7
100	5	10462.2	24.94	8464.5	0.048	11.08	8473.3	0.159	18.29	8462.8	0.033	15.0
100	10	6489.3	44.93	4504.5	0.065	17.84	4507.1	0.116	15.92	4502.4	0.018	14.8
150	2	33851.8	4.80	32324.8	0.091	53.92	32371.3	0.227	39.47	32297.6	0.007	32.9
150	5	14877.6	15.71	12901.9	0.017	32.65	12944.7	0.353	41.03	12904.6	0.037	36.1
150	10	8738.1	34.69	6527.3	0.298	42.09	6527.5	0.330	40.81	6506.6	0.020	36.2
200	2	42691.6	5.06	40654.5	0.012	73.64	40710.4	0.149	57.67	40657.3	0.019	64.3
200	5	17914.7	10.28	16250.4	0.031	71.34	16284.3	0.239	63.56	16253.9	0.052	60.8
200	10	10330.1	24.80	8324.5	0.504	73.66	8358.2	0.909	68.79	8283.6	0.006	68.0
250	2	52997.6	4.53	50720.8	0.014	92.14	50887.6	0.344	85.71	50756.2	0.085	92.0
250	5	22131.7	8.90	20312.6	0.068	92.25	20397.4	0.488	111.39	20317.5	0.089	98.0
250	10	12338.1	18.67	10399.7	0.091	92.30	10439.9	0.477	97.67	10394.2	0.037	92.0
Average			16.96		0.086	45.76		0.257	44.40		0.027	42.5

compiler, coding skill and precision. Generally, the proposed RSA, SA and ABC heuristics take no more than 1.2 seconds to solve small problems, where the Gurobi solver requires significantly more time to solve. To measure the degree of improvement, the overall relative percentage deviation (*RPD*) is used. *RPD* is computed as $[(C_{\max}^{MILP} - C_{\max}^{h})/C_{\max}^{MILP}] \times 100\%$ where $C_{\max}^{MILP}$ and $C_{\max}^{h}$ are the makespan values obtained using MILP and heuristic h, respectively. The *RPDs* of RSA and SA are 4.203 and 4.199, respectively. In summary, the proposed RSA, SA and ABC heuristics obtain

TABLE 7. Wilcoxon rank tests on MIN. *RPD*, MEAN *RPD*, and MAX. *RPD*.

RSA vs.	SA	ABC
Test on Min. *RPD*		
W	18545	12858
P-value	2.2e-16	0.008421
Test on Mean *RPD*		
W	16731	15196
P-value	1.361e-13	7.21e-08
Test on Max. *RPD*		
W	16341	15652
P-value	5.704e-12	2.21e-09

better solutions in less computation time than the Gurobi solver.

D. COMPARING RESULTS OBTAINED USING RSA, SA AND ABC

To validate the performance of the RSA with that of the SA and ABC heuristics, computational experiments were carried out on a large problem set. Given the computational complexity of the STSP, a high-quality solution to a large problem is typically not readily obtainable. Therefore, for each benchmark instance of the large problem set, the *RPD* values of the RSA SA and ABC heuristics were calculated using the best solution found by each, according to the following equation:

$$RPD_h = \frac{C^h_{\max} - C^B_{\max}}{C^B_{\max}} \times 100\% \tag{12}$$

where $C^h_{\max}$ and $C^B_{\max}$ are the makespan value obtained by heuristic h and the best obtained solution, respectively.

Table 6 presents the statistical results of the average makespan ($C_{\max}$), *RPD* and computational times obtained with the large problem set using the RSA, SA and ABC heuristics. As shown in Table 6, RSA outperforms both SA and ABC. Additionally, RSA outperforms SA and ABC on both solutions obtained and the required computational times.

The best, mean and worst makespan values of the solutions to each test problem, based on five runs, obtained using RSA and SA, were used to compute *RPD* values, which were denoted as Min.*RPD*, Mean*RPD* and Max. *RPD*. To determine whether the RSA heuristic outperformed the SA and ABC algorithms, Wilcoxon Rank tests in terms of Min.*RPD*, Mean*RPD* and Max. *RPD* were conducted. The analytical results obtained in Table 7 revealed that, at a confidence level of $\alpha = 0.05$, the proposed RSA heuristic significantly outperformed the SA and ABC in terms of Min. *RPD*, Mean *RPD* and Max. *RPD*. These statistical results verify that the two rules to eliminate ineffective job moves applied to RSA significantly improved the performance of SA and ABC in solving the STSP problem.

VI. CONCLUSIONS AND FUTURE RESEARCH

This study proposes a novel solution to the STSP, which is critical in new product development. The novelty of this work is that the addressed problem with constraints on both the number of workforces and machines has not, to date, been studied in relation to the IPMSP. The contribution of this work is that the investigated problem is not just a purely theoretical model, but can also be applied to many practical manufacturing systems, such as notebook, desktop, laptop and server manufacturers. To reduce the gap between industrial practice and scheduling theory, this work develops an MILP model of the STSP and an RSA heuristic, which applies two rules to eliminate ineffective job moves in order to obtain better neighborhood solutions, for solving it. The proposed RSA heuristic significantly reduces search effort while minimizing makespan in solving the STSP. Experimental results reveal that the RSA yields higher-quality solutions to the STSP than do the SA and the ABC algorithms.

The STSP is a challenging extension of the parallel machine scheduling problem with resource constraints, and has many practical applications. Some suggested directions for future research are based on this work. First, related problems, such as those with various objective functions, can be solved in the future. Second, extensions of the STSP that consider other costs, such as fixed costs of using machines, costs of usage of machines and the costs of tardy jobs, would be an interesting topic. Third, multi-objective STSPs, such as one including the makespan, mean flow time, weighted number of tardy jobs and total setup time for machines, would be complex, but certainly worthy of further research. Finally, future research may also attempt to apply other meta-heuristics or to hybridize other algorithms to solve the STSP.

REFERENCES

[1] C. H. Yeh, J. C. Y. Huang, and C. K. Yu, "Integration of four-phase QFD and TRIZ in product R&D: A notebook case study," *Res. Eng. Des.*, vol. 22, no. 3, pp. 125–141, 2011.
[2] K. Herzig, M. Greiler, J. Czerwonka, and B. Murphy, "The art of testing less without sacrificing quality," in *Proc. 37th Int. Conf. Softw. Eng.*, vol. 1. 2015, pp. 483–493.
[3] S. Gottesman, J. Ramos, and J. Valfre, "Developing built in test to meet the demands of the product test lifecycle," in *Proc. IEEE Autotestcon*, Nov. 2015, pp. 58–64.
[4] S.-W. Lin, Z.-J. Lee, K.-C. Ying, and C.-C. Lu, "Minimization of maximum lateness on parallel machines with sequence-dependent setup times and job release dates," *Comput. Oper. Res.*, vol. 38, no. 5, pp. 809–815, 2011.
[5] S.-W. Lin, K.-C. Ying, Y.-I. Chiang, and W.-J. Wu, "Minimising total weighted earliness and tardiness penalties on identical parallel machines using a fast ruin-and-recreate algorithm," *Int. J. Prod. Res.*, vol. 54, no. 22, pp. 6879–6890, 2016.
[6] K.-C. Ying, "Scheduling identical wafer sorting parallel machines with sequence-dependent setup times using an iterated greedy heuristic," *Int. J. Prod. Res.*, vol. 50, no. 10, pp. 2710–2719, 2012.
[7] A. Doğramaci, "Production scheduling of independent jobs on parallel identical processors," *Int. J. Prod. Res.*, vol. 22, no. 4, pp. 535–548, 1984.
[8] H. Sun and G. Wang, "Parallel machine earliness and tardiness scheduling with proportional weights," *Comput. Oper. Res.*, vol. 30, no. 5, pp. 801–808, 2003.
[9] F. Yalaoui and C. Chu, "Parallel machine scheduling to minimize total tardiness," *Int. J. Prod. Econ.*, vol. 76, no. 3, pp. 265–279, 2002.

[10] S. Tanaka and M. Araki, "A branch-and-bound algorithm with Lagrangian relaxation to minimize total tardiness on identical parallel machines," *Int. J. Prod. Econ.*, vol. 113, no. 1, pp. 446–458, 2008.
[11] S. Özpeynirci, B. Gökgür, and B. Hnich, "Parallel machine scheduling with tool loading," *Appl. Math. Model.*, vol. 40, no. 9, pp. 5660–5671, 2016.
[12] J. Kaabi and Y. Harrath, "A survey of parallel machine scheduling under availability constraints," *Int. J. Comput. Inf. Technol.*, vol. 3, no. 2, pp. 238–245, 2014.
[13] E. Hébrard, M.-J. Huguet, N. Jozefowiez, A. Maillard, C. Pralet, and G. Verfaillie, "Approximation of the parallel machine scheduling problem with additional unit resources," *Discrete Appl. Math.*, vol. 215, pp. 126–135, Dec. 2016.
[14] E.-G. Talbi, *Metaheuristics: From Design to Implementation*. Hoboken, NJ, USA: Wiley, 2009.
[15] S. Nesmachnow, "An overview of metaheuristics: Accurate and efficient methods for optimisation," *Int. J. Metaheuristics*, vol. 3, no. 4, pp. 320–347, 2014.
[16] E. Mokotoff, "Parallel machine scheduling problems: A survey," *Asia–Pacific J. Oper. Res.*, vol. 18, no. 2, pp. 193–242, 2001.
[17] N. Nandagopal, M. Arularasu, B. N. Kumar, and C. N. A. Kumar, "A survey on tardiness parameters in parallel machine scheduling problems," *Int. J. Adv. Manuf. Technol.*, vol. 7, no. 1, pp. 379–383, 2016.
[18] M. Kawakami, "Competing for complementarity: Growth of taiwanese notebook PC manufacturers as ODM suppliers," Competition Cooperation Among Asian Enterprises China, Chosakenkyu-Hokokusho, IDE-JETRO, Chiba, Japan, Interim Rep., 2007.
[19] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
[20] Z.-J. Lee, S.-W. Lin, and K.-C. Ying, "Scheduling jobs on dynamic parallel machines with sequence-dependent setup times," *Int. J. Adv. Manuf. Technol.*, vol. 47, nos. 5–8, pp. 773–781, 2010.
[21] S.-W. Lin and K.-C. Ying, "ABC-based manufacturing scheduling for unrelated parallel machines with machine-dependent and job sequence-dependent setup times," *Comput. Oper. Res.*, vol. 51, pp. 172–181, Nov. 2014.
[22] D. C. Montgomery, *Design and Analysis of Experiments*, 8th ed. New York, NY, USA: Wiley, 2012.

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, Linkuo Chang Gung Memorial Hospital, Taoyuan, Taiwan, and with the Department of Industrial Engineering and Management, Ming Chi University of Technology, Taipei, Taiwan. His current research interests include metaheuristics and data mining. 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*.

CHIEN-YI HUANG received the Ph.D. degree from the State University of New York, Binghamton, in 1996. He served as the Chief Process Technology Head at Wistron Corporation, responsible for new process technology enabling and materials characterization. He is currently a Professor with the National Taipei University of Technology, Taipei, Taiwan. His research interests include process optimization and electronics reliability.

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 has authored or co-authored 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 Innovational 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.

DAR-LUN CHEN received the E.M.B.A. degree from the School of Business, Chang Gung University, in 2016. He currently serves as a Project Manager with the Department of Quality Assurance, Quanta Computer, Taiwan. He has over 20 years working experience in streamlining operations and increasing efficiency. His current research includes project management and scheduling.