IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 6, 2019, accepted November 18, 2019, date of publication November 26, 2019, date of current version December 11, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2955975

Optimize Unrelated Parallel Machines Scheduling Problems With Multiple Limited Additional Resources, Sequence-Dependent Setup Times and Release Date Constraints

IBRAHIM M. AL-HARKAN AND AMMAR A. QAMHAN
Department of Industrial Engineering, King Saud University, Riyadh 11451, Saudi Arabia
Corresponding author: Ammar A. Qamhan (435108378@student.ksu.edu.sa)

This work was supported by the Deanship of Scientific Research at King Saud University through the Initiative of DSR Graduate Students Research Support (GSR).

ABSTRACT This paper considers the problem of scheduling a set of jobs on unrelated parallel machines subject to several constraints which are non-zero arbitrary release dates, limited additional resources, and non-anticipatory sequence-dependent setup times. The objective function is to minimize the maximum completion time. In order to find an optimal solution for this problem, a new mixed-integer linear programming model (MILP) is presented. Moreover, a two-stage hybrid metaheuristic based on variable neighborhood search hybrid and simulated annealing (TVNS_SA) is proposed. In the first stage, a developed heuristic is used to find an initial solution with good quality. At the second stage, the obtained initial solution is used as the first neighborhood structures in the proposed metaheuristic, for further progress different neighborhood structures and effective resolution schemes are also presented. The computational results indicate that the proposed metaheuristic is capable of obtaining optimal solutions for most of the instances when compared to the solution obtained by the developed mixed-integer linear programming model. In addition, the metaheuristic dominated the MILP with respect to computing time. The overall evaluation of the proposed algorithm shows its efficiency and effectiveness when compared with other algorithms. Finally, in order to obtain rigorous and fair conclusions, a paired t-test has been conducted to test the significant differences between the five variants of the TVNS_SA.

INDEX TERMS Scheduling, parallel machines, renewable resources, variable neighborhood search.

NOMENCLATURE

MILP	Mixed integer linear programming model
TSVNS-SA	Two stage hybrid Variable Neighborhood Search with Simulated Annealing
VNS	Variable Neighborhood Search
PMSP	Parallel machine problems
SPT	Short Process Time first
LPT	longest Process Time first
ERD	Earlier release date first
ARPD	Average relative percentage deviation
Rmax	Maximum number of rejection
Vmax	Maximum number of neighborhood structures

The associate editor coordinating the review of this manuscript and approving it for publication was Yanbo Chen.

I. INTRODUCTION

Production scheduling problems are the most probably related problem to the industrial world since it is a question of reconciling, optimally, limited resources with activities in time. In addition to the industrial field, there are other areas that benefited from scheduling such as education, agriculture, transportation or health research. The addressed scheduling problem in this study is often found in manufacturing processes such as painting, metalworking [1], shipyard [2], and semiconductor manufacturing [3]. This study considers a scheduling problem of unrelated parallel machines under several constraints with the objective of minimizing the maximum completion time. The constraints considered listed are as follow:

- The release time for some jobs or all jobs is a non-zero unit of time.

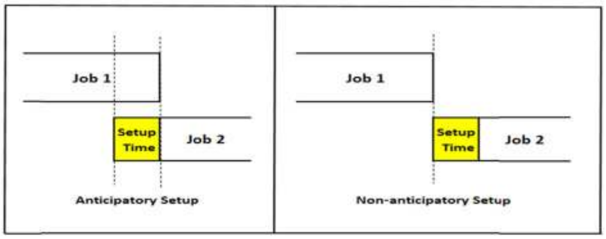


FIGURE 1. **Difference between anticipatory and non-anticipatory setups.**

- The setup time for any job depends on the job itself and its immediately preceding job, that is, the setup time for a job i after job j it should not equal the setup time for the same job i after any other job. Furthermore, setup times are non-anticipatory, that is the setup procedures for a corresponding job cannot be started until the allocated resources are available as shown in Figure.1.
- The machines are not the only restricted resource; there also are other resources with limited capacity at any unit of time. Moreover, the resources are renewable which means it returned after its usage such as industrial robots, machine operators, equipment, tools, …etc.

In this context, this study develops an exact methodology and metaheuristic to solve different size problems, which made the contribution of this paper different from the existing work as it will be shown in the literature reviewed and to the best of our knowledge. This work led to the development of solving methods for the scheduling problem in the type of workshop that is described previously. Also, it is clear from the literature reviewed that just a few papers have been reported on the problem addressed in this paper. The only paper in the literature that addressed a similar problem to the one considered [4].

The main difference to the addressed problems is:

- The nature of the setups: while anticipatory setups are considered in their study, non-anticipatory setups are considered in this study.
- The mathematical programming: while a nonlinear pure integer model is provided in their study, a mixed-integer linear programming model is presented in this study.

The rest of the paper is organized as follows: Section 2, presents the considered problem background and literature review. Section 3, defines the problem and introduces the formulation of the developed mixed-integer linear programming model (MILP); Section 4, presents the developed metaheuristics; Section 5, provides experimental results and computational analysis, and finally, Section 6 draws conclusions.

II. LITERATURE REVIEW

Parallel machine scheduling (PMSP) is an environment that classified according to three categories of parallel machines which are the identical parallel machines, the uniform parallel machines, and the unrelated machines [5]. An unrelated parallel machine problem is a generalization of the other parallel machine and it more general and complex to deal with. The PMSP has been widely investigated in the past few decades. Lenstra *et al.* [6] construct an approximation method based on the rounding of a solution obtained by linear programming. The authors show that when the unit of processing times are in the set $\{1, 2\}$, the scheduling problems with unrelated parallel machines to minimize the makespan is polynomial. In any other case where the processing times are in a set of $\{p, q\}$ where p is an integer less than q and $2p \neq q$ the problem is NP-difficult. For general reviews and applications of the unrelated parallel scheduling problems, the readers can refer to books such as Pinedo [7] and Blazewicz *et al.* [8].

Generally, in most of the existing parallel machine scheduling studies considered the machines as the only restricted resource. In practical manufacturing environments, some resources are required with the assigned machine to perform a certain job. According to resources renewability, the resources are classified into three major sections [9]. A renewable resource is limited and fixed at any unit of time and could to be reused for another job such as industrial robots, machine operators, equipment, or tools. A non-renewable resource is consumed by jobs such as raw material, energy, or money. A resource is both renewable and non-renewable. A superior comprehensive survey present by Edis *et al.* [9] to discuss scheduling problems. In which an additional resource on five main categories which are the machine environment, additional resource, objective functions, complexity results, solution methods, and other important issues. For the past few years, studying scheduling problems with additional resources constraints have received considerable attention. Studying the complexity of the scheduling problem with non-renewable resources is provided in [10]–[14]. Considering one common renewable resource to minimize makespan in [5], and to minimize makespan and total carbon emission (TCE) in [15]. A modified fruit fly optimization algorithm and a mixed-integer linear programming model are presented in their works. The MILP of the work [5] has some deficiencies which were discussed and corrected in [16]. Integer mathematical programming model (ILP) and two genetic algorithms were proposed in [17] to optimize makespan in unrelated parallel machine scheduling problems with renewable resource-constrained and machine eligibility restrictions. A uniform scheduling problem is studied by Li *et al.* [18] with resource-dependent release dates. A variable neighborhood search algorithm and a simulated annealing algorithm Scheduling problems were also proposed. Villa *et al.* [19] proposed several heuristics based on resources and assignments to minimizing the makespan on unrelated parallel machines with one renewable additional resource. Abdeljaoued *et al.* [20] provide two new heuristics and a simulated annealing metaheuristic for the parallel machine scheduling problem with a set of renewable resources. A two mixed-integer programming approach and tabu search algorithm were provided in [21] to study the problem of scheduling the operations on parallel machines with their required tools. For the same problem, the study of Özpeynirci *et al.* [21] presented three constraint programming models. Multiprocessor scheduling with additional resources is studied by Furugyan [22] where the interruptions are allowed and the task execution may be switched from one processor to another. Dosa *et al.* [23] studied the parallel machine scheduling with job assignment restrictions and renewable constrained resource. Wang *et al.* [24] presented A two-stage heuristic for the constrained parallel machine scheduling. The authors addressed the problem of selecting the proper cutting conditions for various jobs under the condition that power consumption never exceeds the electricity load limit. Labbi *et al.* [25] provide some heuristic algorithms to optimize two identical machines makspane with preparation requires renewable resource constraints. Li *et al.* [26] addressed the case steelmaking scheduling problems with multiple constrained resources, a discrete artificial bee colony and several heuristics that developed for the considered problem.

In addition, there are numerous works have been carried out for scheduling problems with setup time. Allahverdi *et al.* [27] present a comprehensive survey on scheduling problems with setup. The authors classify scheduling problems into sequence-independent and sequence-dependent setup. In addition, they categorize the literature according to the workshop environments (single machine, parallel machines, flow shops, and job shops). This survey is updated in [28] and [29], the authors introduced the distinction between anticipatory and non-anticipatory setups. A setup is anticipatory when the setup procedures for a corresponding job can be started regardless the machine is available or not; otherwise, the setup is said to be non-anticipatory. Koulamas and Kyparisis [30] treat the case of sequence-dependent setups time for single machine using several objective functions, which are makespan (Cmax), the total completion time (TC), the total absolute differences in completion times (TADC), and a bi-criteria combination (BC). The authors show that the problem can be solved in O (n log n) time by a sorting procedure. Vélez-Gallego *et al.* [1] studied the case of a single machine by considering two constraints non-zero release dates and sequence-dependent setup. They proposed mixed-integer linear programming and beam search heuristic time to minimize the total completion time. Concerning problems with parallel machines, Gendreau *et al.* [31], and Mendes *et al.* [32] treat the case of identical parallel machines, with setups depending on the sequence in which the makespan was minimized. Gendreau et al. propose lower bounds and a merge heuristic. Mendes et al. implement two metaheuristics, tabu search, and memetic algorithm. Lin and Ying [33] proposed a metaheuristic based on a hybrid artificial bee colony to minimize the makespan on unrelated parallel machines scheduling with sequence-dependent setup times. Ezugwu and Akutsah [34] deal with the same problem described above, they propose a metaheuristic based on a firefly algorithm to minimize the makespan. A part of the work of Meng *et al.* [35] deals with the hybrid flow shop scheduling problem with unrelated parallel machines and sequence-dependent setup times. In addition, the work of Naderi *et al.* [36] provides three mathematical modeling and hybrid particle swarm optimization algorithm with local search algorithm to solve the problem of hybrid flow shop scheduling. Afzalirad and Rezaeian [2] proposed two metaheuristics for minimizing the total mean flow time on an unrelated parallel machine scheduling problem with sequence-dependent setup times, release dates, machine eligibility, and precedence constraints. Weng *et al.* [37] study the impact of seven heuristics to minimize the weighted mean completion time with sequence-dependent setup. Lin and Hsieh [38] proposed a mixed integer programming model, a heuristic, and iterated hybrid metaheuristic to minimize the total weighted tardiness for an unrelated parallel scheduling problem with ready times and sequence- and machine-dependent setup times. They show that the iterated hybrid metaheuristic outperforms tabu search and ant colony optimization in terms of total weighted tardiness. Emami *et al.* [39] considered the scheduling problem to maximize the profits of order processing on a non-identical parallel machines with sequence-dependent setup. The profits computed based on the revenue, and unit tardiness penalty cost for each order. In their study they present a MILP and a Benders decomposition approach for solving the studied problem. Obeid *et al.* [40] proposed two Mixed Integer Linear Programming models to solve a problem with different parallel machines with setup time dependent on a family of the preceding task. In this case, the setup times do not depend on the sequence. Zeidi and Mohammad Hosseini [41] formulate a new mathematical model to minimize the total cost of tardiness and earliness on unrelated parallel machines with considering sequence-dependent setup times constraints. They also propose an integrated metaheuristic and evaluated it under the relevant existing benchmark test data. Rabadi *et al.* [42] propose a metaheuristic for unrelated PMSP with machine-dependent and sequence-dependent setup times to minimize the makespan. The results obtained show the effectiveness for the metaheuristic when compared to Partitioning Heuristic outcomes. Hamzadayi and Yildiz [43] proposed a genetic algorithm and a simulated annealing for solving the scheduling problem of m identical parallel machines with sequence dependent setup times. They assume that the setup is carried out by a common server which does not belong to the set of parallel machines The proposed algorithms evaluated by the results of mixed integer linear programing model for small sized problem and the results of basic dispatching rules for all sized problem. A MILP, tabu Search, and simulated annealing algorithms were presented in the study of Bektur and Saraç [44] to minimize the total weighted tardiness of a scheduling problem of unrelated parallel machine with a common server and sequence dependent setup times.

III. PROBLEM FORMULATION

A. PROBLEM DEFINITIONS

In this study, the considered problem can be summarized as follows: there are n jobs $(J_1, J_2, \ldots, J_n)$ which are

TABLE 1. MILP notations.

MILP Indices	
i, j	the index of job, $i, j = 1,2,\ldots,n$;
k	the index of machines, $k = 1,2,\ldots,m$;
v	the index of resources, $v = 1,2,\ldots,R$;
T	the index of time, $T = 1,2,\ldots,T_{max}$ "a large enough number" ;
MILP Binary Decision variables	
x_{ik}	x_{ik} equal 1 if J_i is assigned to M_k; otherwise x_{ik} zero.
y_{ijk}	y_{ijk} equal 1 if job J_j is processed immediately after job J_i and is being processed on M_k; otherwise y_{ijk} is zero.
z_{ikT}	z_{ikT} equal 1 if job i is being processed on M_k at time T; otherwise z_{ikT} is zero.
MILP Real Decision variables	
t_i	the starting time of job i
f_i	the completion time of job i
C_{max}	Maximum completion time
MILP Parameters	
p_{ik}	processing time of job i on machine k
s_{jik}	setup time of job i dependent on job j and is being processed on machine k
r_i	release date of job j
Res_{iv}	amount of resource v required by job i
Av_Res_v	Available resource of type v

processed on m unrelated parallel machines ($M_1, M_2, \ldots, M_m$) which are always available over the planning horizon. Each machine M_k can process only one job J_i at a time by (p_{ik}) units of processing time. Each Jobs J_i has a known release date ($r_1, r_2, \ldots, r_n$). Every job J_i has a setup time s_{ij} depends on the job itself and its immediately preceding job J_i. There are v types of resources ($R_1, R_2, \ldots, R_v$), each resource has a limited number of units at any point of time ($Av_{R1}, Av_{R2}, \ldots, Av_{Rv}$). Each job J_i requires a specific amount (Res_{iv}) of resource R_v per unit of time. Each job J_i cannot be processed until all allocated resource beside the assigned machine is available. The objective is to minimize the maximum completion time of all jobs (the makespan).

B. MIXED INTEGER LINEAR PROGRAMMING MODEL (MILP) FORMULATION

In order to find the optimal solution for the problem considered in this paper, a Mixed Integer Linear Programming (MILP) model is developed. Table 1 present the notations of the developed MILP.

The MILP model can be formulated as follows:

$$\textbf{Min } \boldsymbol{C_{max}} \tag{1}$$

Subject to

$$f_i - C_{max} \le 0, \quad i = 1, 2, \ldots, n \tag{2}$$

$$\sum_{k=1}^{m} x_{ik} = 1, \quad i = 1, 2, \ldots, n \tag{3}$$

$$\sum_{(q=1, i\neq j)}^{n} y_{qjk} = x_{jk}, \quad j=1, 2, \ldots, n; \; k=1, 2, \ldots, m \tag{4}$$

$$\sum_{(j=1, i\neq j)}^{n} y_{ijk} \le x_{ik}, \quad i=1, 2, \ldots, n; \; k=1, 2, \ldots, m \tag{5}$$

$$\sum_{(j=1)}^{n} y_{0jk} = 1, \quad k = 1, 2, \ldots, m \tag{6}$$

$$f_i = t_i + \sum_{k=1}^{m} p_{ik}\, x_{ik} + \sum_{q=0}^{n}\sum_{k=1}^{m} s_{qik} y_{qik}, \quad i = 1, 2, \ldots, n \tag{7}$$

$$f_0 = 0 \tag{8}$$

$$f_q - t_j + M\left(\sum_{k=1}^{m} y_{qjk} - 1\right) \le 0, \quad j = 1, 2, \ldots, n; \quad q = 0, 1, \ldots, n \tag{9}$$

$$r_i - t_i \le 0, \quad i = 1, 2, \ldots, n \tag{10}$$

$$\sum_{k=1}^{m}\sum_{T=0}^{T_{max}} z_{ikT} = f_i - t_i, \quad i=1, 2, \ldots, n \tag{11}$$

$$\sum_{T=0}^{Tmax} z_{ikT} \le M x_{ik}, \quad i=1, 2, \ldots, n; \; k=1, 2, \ldots, m \tag{12}$$

$$f_i \ge T \sum_{k=1}^{m} z_{ikT}, \quad i=1, 2, \ldots, n; \; T=1, 2, \ldots, T_{max} \tag{13}$$

$$t_i \le T \sum_{k=1}^{m} z_{ikT} + M\left(1 - \sum_{k=1}^{m} z_{ikT}\right), \quad i=1, 2, \ldots, n; \quad T = 1, 2, \ldots, T_{max} \tag{14}$$

$$\sum_{i=0}^{n}\sum_{k=1}^{m} Res_{iv}\, z_{ikT} \le Av_{Resv}, \quad v = 1, 2, \ldots, R; \quad T = 1, 2, \ldots, T_{max} \tag{15}$$

$$x_{ik} \in \{0, 1\}, \quad i = 1, 2, \ldots, n; \; k = 1, 2, \ldots, m \tag{16}$$

$$y_{ijk} \in \{0, 1\}, \quad i = 1, 2, \ldots, n; \; j = 0, 1, \ldots, n; \quad k = 1, 2, \ldots, m \tag{17}$$

$$z_{ikT} \in \{0, 1\}, \quad i = 1, 2, \ldots, n; \; k = 1, 2, \ldots, m; \quad T = 1, 2, \ldots, T_{max} \tag{18}$$

$$t_i, f_i \ge 0, \quad i = 1, 2, \ldots, n(20) \tag{19}$$

The objective function (1) is to minimize the maximum completion time of all the jobs. Constraint set (2) calculates the maximum completion time. Constraint set (3) ensures that each job is assigned exactly to one machine. Constraint sets (4) until (6) are to choose one binary variable for the setup time for each job on one machine with considering the immediate predecessor job. Constraint set (7) calculates the completion time of each job. Constraint set (8) fix the dummy job as the first job in the schedule. Constraint set (9) guarantee the job cannot start before its predecessor job. Constraint set (10) guarantee the job cannot start before their release time. Constraint set (11) ensures that the total number of units of decision variable 'z_{ikT}' for each job are

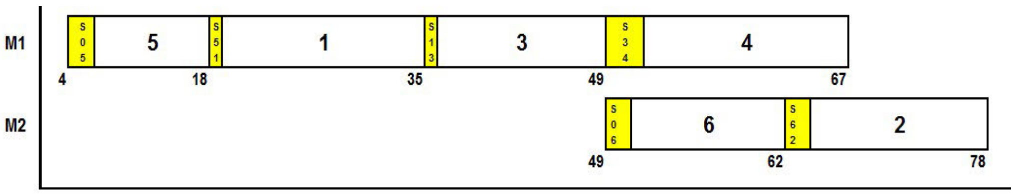


FIGURE 2. Optimal schedule for the numerical example.

TABLE 2. Data for the parallel machine scheduling.

Job	1	2	3	4	5	6
P_{i1}	16	16	13	14	11	19
P_{i2}	20	14	16	13	19	11
R_i	17	13	26	24	4	15
Res_{i1}	2	2	3	1	2	2
Res_{i2}	0	1	0	1	1	1
	$Av\_Res_1 = 3$			$Av\_Res_2 = 2$		

TABLE 3. Setup times matrix on machine 1.

Job	1	2	3	4	5	6
0	4	1	3	2	3	4
1	-	3	1	2	2	4
2	4	-	3	2	1	2
3	3	2	-	4	1	2
4	2	3	3	-	3	3
5	1	3	3	1	-	2
6	4	3	3	1	2	-

exactly equals its running time. Constraint set (12) ensures that all units of decision variable 'z_{ikT}' for each job are allocated to their accurate machine. Constraints set (13) and (14) ensure that all units of decision variable 'z_{ikT}' for each job are allocated between its starting time and completion time. Constraint set (15) imposes the condition that the total amount of resource assigned to a job at any point in the time horizon is less than or equal to the available amount of that resource. Constraint sets (16) until (18) are to force integrality for binary decision variables. Constraint (20) imposes non-negativity for real decision variables.

To evaluate the effectiveness of the proposed optimization model. An example instance consists of six jobs (n = 6), two machines (m = 2) and two additional resources (R = 2). The input parameters related to the example of each job on each machine are listed in Table 2, Table 3, and Table 4. The optimal solution obtained by using CPLEX solver under GAMS software version 24.1.2. The Gantt chart for the optimal schedule is given in Figure 2.

To achieve the best performance with respect to the CPU computation time for the GAMS software, the default lower bound was changed from zero to the obtained value of lower bound that will be discussed in the following section.

C. LOWER BOUND

To assess the performance and robustness of the proposed algorithms, three efficient lower bounds are developed and presented in this study. First, the problem is simplified to be

TABLE 4. Setup times matrix on machine 2.

Job	1	2	3	4	5	6
0	3	2	4	3	3	2
1	-	3	2	3	2	3
2	2	-	4	3	3	2
3	3	3	-	1	2	3
4	4	2	4	-	2	3
5	4	3	1	4	-	2
6	3	2	3	3	3	-

similar to a machine's problem by selecting the minimum processing time and setup time for each job such that

$$P_i = \min_{k \in m} (p_{ik}), \quad \forall i \in n$$

$$S_i = \min_{k \in m} \left(\min_{j \in \{0,n\}} s_{jik} \right), \quad \forall i \in n$$

The considered lower bounds are presented as follows:

1) FIRST LOWER BOUND

This lower bound is presented in the work of Afzalirad and Rezaeian [4]. The property depends on the capacity of each resource and given as follows.

$$LB_1 = \grave{LB}_1 + \left\lfloor \min_{i \in n} r_i + S_i/m \right\rfloor$$

$$\text{Where } \grave{LB}_1 = \max_{v \in R} \left(\sum_{Res_{iv} > \frac{AR_v}{2}} P_i + \frac{1}{2} \sum_{Res_{iv} = \frac{AR_v}{2}} P_i \right)$$

The $\grave{LB}_1$ is obtained the maximum expected time for each type of additional resources with considering only the jobs that occupy half of the resource capacity or more.

2) SECOND LOWER BOUND

This lower bound is presented in the work of Qamhan *et al.* [45]. The property depends on the latest release date and given as follows

For any feasible schedule s, $\max_i (r_i + P_i + S_i) \le C_{max}$

Note: The lower bound of the problem is estimated by the maximum value between these three properties.

IV. METAHEURISTICS

Metaheuristics algorithms represent an alternative way of dealing with large-sized problems or combinatorial optimization problems. Despite the currently available technologies,

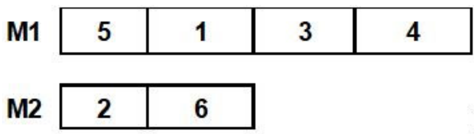


FIGURE 3. First representation for the example.

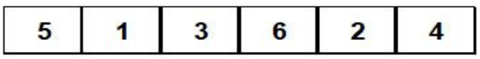


FIGURE 4. Second representation for the example.

the computation time for the exact methods for most of large-size problems in the literature is very long and it is unrealizable. As a result, exact methods become ineffective for large-size problems. Metaheuristics approaches can significantly reduce the computation time without necessarily leading to the optimal solution. In this study, our main interest in the hybrid variable neighborhood search with simulated annealing. In the following sections a detailed description of developed metaheuristics is given.

A. SOLUTION REPRESENTATION

A good representation of the solution should be simple, easy to understanding, and also make the algorithm work effectively. To assign a certain sequence of jobs in their proper machine by considering the limited resources at any point in time, it needs a suitable representation with two schemes. The first representation shows in Figure 3 an example of a feasible solution to the problem in Tables 2 to 4. The second representation is for the priority of doing a certain job in the timeline schedule horizon and it showed in Figure 4.

Because of the limited resources, it cannot perform the job 5 on machine1 and job 2 on machine 2 in Figure 3 at the same point of time, so the job with the higher priority will be processed first, and the final result shows in Figure 2.

B. A HEURISTIC FOR AN INITIAL SOLUTION (FIRST STAGE)

According to the solution representation (Figures 3&4), it needs two approaches to order the heuristic clearly. Firstly, the heuristic determines the loading of the machine by assign jobs to the candidates' machines as in the first representation in Figure 3, and then it obtains the job priority sequence like the second representation in Figure 4. For the machine loading, we used a very well-known dispatching rule by assigning each job to the fastest machine, and to the second part of the heuristic that concern with the job priority sequence, three basic dispatching rule has been tested: - Short Process Time (SPT), longest Process Time (LPT), and Earlier release date first(ERD).

To evaluate the performance of the initial heuristic based on the three dispatching rules, a set of computational instances was generated by the parameters in Table 5 where: number of jobs n= {5, 6}, number of machines m= {2, 3, 4} and number of additional resources R= {1, 2, 3, 4}. The three algorithms were tested on 96 problem instances (4 instances for each combination n∗m∗R) and evaluated by using the Average relative percentage deviation (ARPD) of each combination, where the RPD is calculated as follows:

TABLE 5. The generating conditions of test problems.

Group	Release dates	Processing times	Setup times	Available amount of each resource	Resource requirements
Small size	(1, 30)	(10, 20)	(1, 4)	(1, 2)	(0, AR)
Large size	(1, 200)	(20, 50)	(4, 10)	(1, 5)	(0, AR)

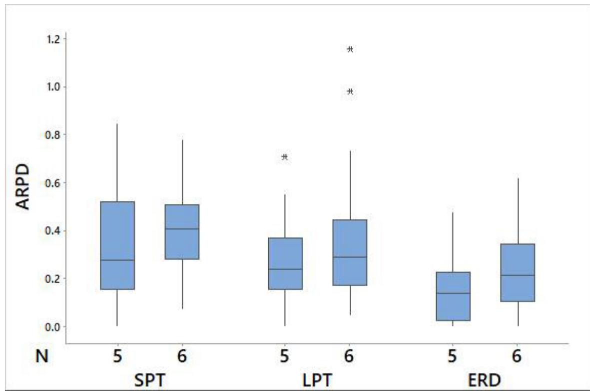


FIGURE 5. Box Plot ARPD values versus a number of jobs.

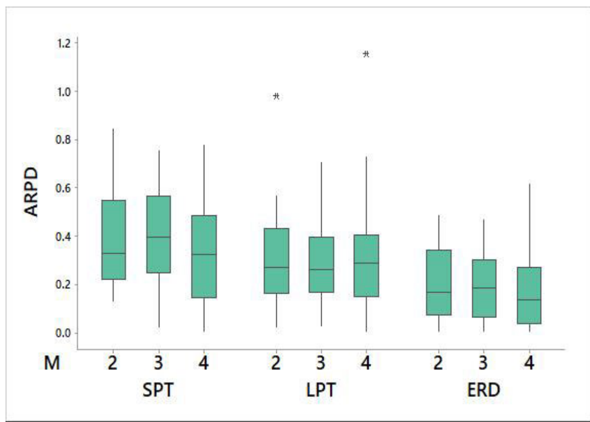


FIGURE 6. Box Plot ARPD values versus a number of machines.

$$RPD = \frac{|Method_{sol} - Best_{sol}|}{Best_{sol}} x100 \tag{20}$$

where '$Method_{sol}$' is the maximum completion time obtained from the different algorithms and $Best_{sol}$ is the optimal completion time obtained by CPLEX solver.

The three algorithms are coded in C and run on a PC including Intel(R) Core(TM) i3 CPU with 2.53 GHz speed and 3GB of RAM.

Figures 5–7 show the Box Plot of the ARPD values versus the number of jobs (n), number of machines(m) and the number of resources (R), respectively. It seems from Figures 5–7

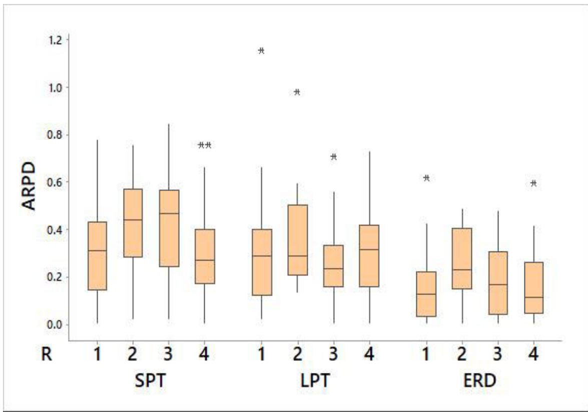


FIGURE 7. Box Plot ARPD values versus a number of resources.

that the initial heuristic based on the Earliest Release Date has a better performance than the other candidates.

C. HYBRID VARIABLE NEIGHBORHOOD SEARCH WITH SIMULATED ANNEALING (SECOND STAGE)

Our attention has focused more particularly on a very well-known and used metaheuristic Variable neighborhood search that firstly developed by [46] for solving hard combinatorial problems. This method has two main phases: A local search phase in which the method consists of exploring the current neighborhood to find a local optimum, and a shaking phase put in place to refresh and reiterating to avoid being trapped in local optimum by considering more than one neighborhood starting from an initial solution.

A variable neighborhood search has been used to solve many optimization problems in various domains. For unrelated parallel machine scheduling problems, we can cite the works of Charalambous *et al.* [47] which they proposed a variable neighborhood descent hybrid with mathematical programming for the unrelated parallel machines scheduling with the objective of minimizing the complication time. They extend their problem to consider setup times constraints in [48] and used the same algorithm as a solution method. VNS approaches improve decomposition schemes is presented in [49] and an investigation of the relationship between the shaking step and the local search phase in a basic VNS approach was provided. Cruz-Chávez *et al.* [50] present a comparative review of different neighborhood structures that consider the variable neighborhood search to optimize well-known benchmarks Unrelated Parallel Machines Problems. Yazdani *et al.* [51] proposed a modified variable neighborhood search for the problem of a single machine scheduling problem with multiple unavailability constraints. Sevkli and Sevilgen [52] proposed a hybrid method combining Reduced Variable Neighborhood Search (RVNS) edited with a particle swarm algorithm. In the following subsections is a brief description of the components of VNS in the proposed algorithm.

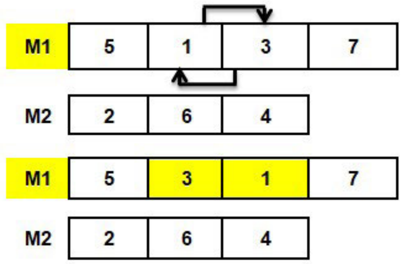


FIGURE 8. Local search structure I.

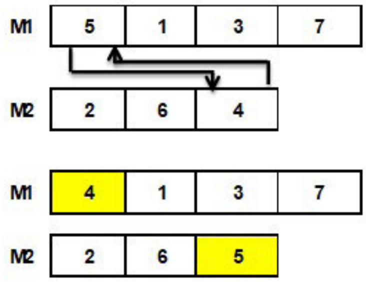


FIGURE 9. Local search structure II.

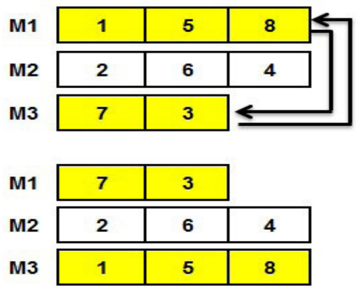


FIGURE 10. Local search structure III.

1) NEIGHBORHOOD STRUCTURES

The main function of the neighborhood structure is to generate a new solution from the neighbors of the current solution. We have used six neighborhood structures. LNI to LNV is used for the local search and SNI is used for the shaking procedure by considering a different swapping and insertion.

These structures will be defined in the following

a. Local search structure I

The procedure of this method consists in swap two jobs selected randomly in the same machine for a possible improvement of the objective function. The steps of this method are as follows (see Figure 8):

b. Local search structure II

The procedure of this method consists of swap two jobs selected randomly in the different machines for a possible improvement of objective function The steps for this method are as follows (see Figure 9):

c. Local search structure III

The steps of this method are as follows (see Figure 10):

Step 1: Select two machines at random.

Step 2: Swap the machines loading of jobs between the selected machines.

TABLE 6. Comparisons TVNS-SA with CPLEX and initial heuristic.

No	Problem size	Average RPD				Min RPD				Max RPD			
		Stage one			Final Stage	Stage one			Final Stage	Stage one			Final Stage
		Initial Heuristic Based on			Hybrid VNS with SA	Initial Heuristic Based on			Hybrid VNS with SA	Initial Heuristic Based on			Hybrid VNS with SA
	N x M x R	SPT	LPT	ERD		SPT	LPT	ERD		SPT	LPT	ERD	
1	5 x 2 x 1	29%	15%	13%	0%	16%	2%	8%	0%	43%	37%	17%	0%
2	5 x 2 x 2	39%	33%	18%	0%	17%	13%	6%	0%	40%	38%	17%	0%
3	5 x 2 x 3	54%	29%	20%	0%	23%	24%	0%	0%	85%	35%	48%	0%
4	5 x 2 x 4	36%	32%	17%	0%	14%	20%	0%	0%	76%	47%	42%	0%
5	5 x 3 x 1	22%	26%	4%	0%	2%	5%	0%	0%	29%	53%	5%	0%
6	5 x 3 x 2	38%	26%	16%	0%	2%	16%	0%	0%	67%	53%	27%	0%
7	5 x 3 x 3	41%	35%	24%	0%	10%	19%	16%	0%	60%	27%	28%	0%
8	5 x 3 x 4	33%	22%	17%	0%	16%	2%	2%	0%	76%	36%	35%	0%
9	5 x 4 x 1	6%	30%	4%	0%	0%	18%	0%	0%	16%	41%	13%	0%
10	5 x 4 x 2	28%	25%	22%	0%	13%	15%	15%	0%	49%	24%	39%	0%
11	5 x 4 x 3	35%	21%	13%	0%	2%	0%	0%	0%	54%	44%	46%	0%
12	5 x 4 x 4	32%	21%	14%	0%	0%	0%	4%	0%	67%	49%	21%	0%
13	6 x 2 x 1	43%	33%	32%	0%	27%	10%	17%	0%	58%	45%	35%	0%
14	6 x 2 x 2	58%	56%	32%	0%	33%	16%	20%	0%	76%	98%	49%	0%
15	6 x 2 x 3	29%	17%	22%	1%	15%	11%	7%	0%	51%	21%	46%	4%
16	6 x 2 x 4	25%	30%	8%	0%	13%	5%	4%	0%	41%	36%	12%	0%
17	6 x 3 x 1	43%	32%	19%	0%	35%	13%	3%	0%	56%	67%	31%	0%
18	6 x 3 x 2	46%	31%	35%	0%	24%	22%	7%	0%	45%	45%	43%	0%
19	6 x 3 x 3	55%	29%	18%	0%	38%	9%	6%	0%	75%	56%	31%	0%
20	6 x 3 x 4	39%	30%	22%	0%	33%	13%	11%	0%	45%	41%	34%	2%
21	6 x 4 x 1	40%	46%	22%	0%	14%	9%	0%	0%	78%	116%	62%	0%
22	6 x 4 x 2	48%	37%	31%	0%	39%	29%	13%	0%	51%	60%	42%	0%
23	6 x 4 x 3	42%	26%	19%	0%	17%	12%	1%	0%	65%	34%	27%	1%
24	6 x 4 x 4	21%	44%	20%	1%	7%	15%	2%	0%	24%	73%	60%	4%

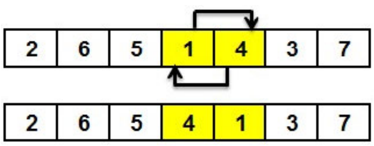


FIGURE 11. Local search structure IV.

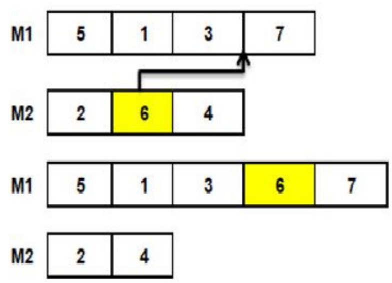


FIGURE 12. Local search structure V.

Step 3: Test and update if there is any improvement in the objective function.

Step 4: If the new adjustments reject "i+1".

Step 5: Go to step 1 until i > Rmax.

d. Local search structure IV

This method consists in swap the priority for two jobs selected randomly for a possible improvement of the objective function, and if the selected jobs load in the same machine the swapping will be also in the job sequence on the machine (see Figure 11).

e. Local search structure V

This method consists in to remove any job selected randomly and then relocated to another randomly selected position.

f. Neighborhood shaking structure

This method consists of creating randomly a new neighborhood structure for all machines with keeping the same job priority sequence for the current best solution.

2) STOPPING CRITERIA

The stopping condition for the most algorithms depends on the application like correspond to a maximum number of iterations or an algorithmic time allocated …etc. For our study, the proposed algorithm takes two parameters as stopping condition: the maximum number of rejection (Rmax) for the local search phase, and the maximum number of neighborhood structures (Vmax) for the shaking phase. The algorithm will explore each neighborhood closely, starting with the first whenever a better solution can be found and is stopped for exploring the neighborhood when the number of rejection reaches the maximum number of non-improvement in local search (Rmax), and the algorithm stopped after exploring all the neighborhood structures (Vmax).

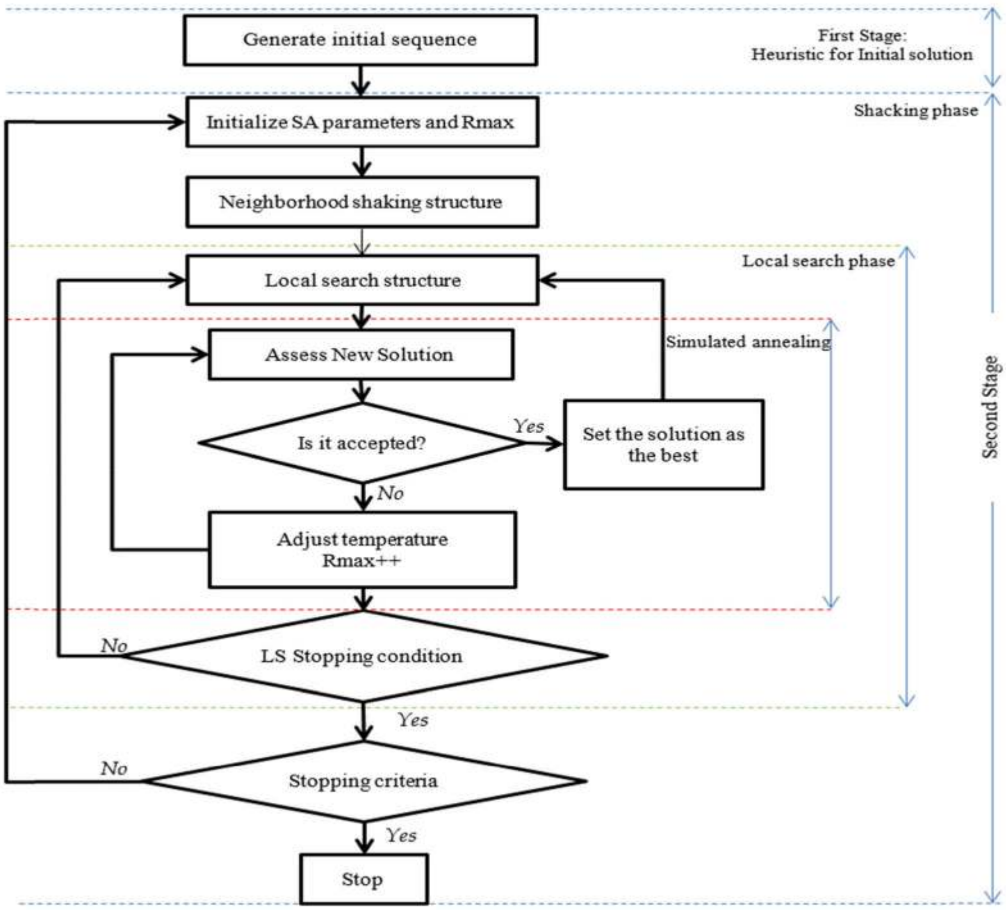


FIGURE 13. Flowchart of the TVNS-SA.

D. REPRESENTATION AND DECODING SCHEME

The structure of the proposed algorithm 'Two-stage hybrid Variable Neighborhood Search with Simulated Annealing (TVNS-SA)' is shown in Figure 13.

V. COMPUTATIONAL RESULTS AND PERFORMANCE EVALUATION

The TVNS-SA algorithm is coded in C and run on a PC including Intel(R) Core(TM) i3 CPU with 2.53 GHz speed and 3GB of RAM.

A. DATA GENERATION

As a wide set of benchmark instances is not available for the addressed problem, a set of test instances are produced randomly. About 288 test instances are generated using integer uniform distribution which had been used from the literature and adopted from the study of Afzalirad and Rezaeian [4]. Table 5 demonstrates the minimum and maximum values of the random integer uniform distribution generator for the instance input parameter such as processing time, release dates, …etc.

Four instances for each combination n∗m∗R was generated where: n number of jobs = {5, 6} for small size and n= {40, 50, 60} for large size, m number of machines = {2, 3, 4} for small size and n= {2, 4, 6, 8} for large size, and R number of additional resources = {2, 4, 6, 8}.

B. EXPERIMENTAL RESULTS

The computational results are closely explored in this section. At first, the proposed MILP model is validated through the small-scaled test instances. Then, the obtained results from the MILP are compared with the obtained results from the TVNS-SA algorithm. The computational results for the small-scaled instances are summarized in Table 6 and Table 7. Where: 'Average RPD 'is the average relative percentage deviation of the solution from the optimal solutions, 'Min RPD' is the minimum RPD, and 'Max RPD' is the maximum RPD.

The RPD is calculated based on the formula (20). As Table 7 indicates, the proposed TVNS-SA is able to obtain exact solutions in reasonable CPU times for all instances while CPLEX solver has solved the problems in extremely longer CPU times. Therefore, it can be concluded that TVNS-SA is effective and efficient for solving most of the problem instances. To assess the algorithm's most important and significant neighborhoods under different

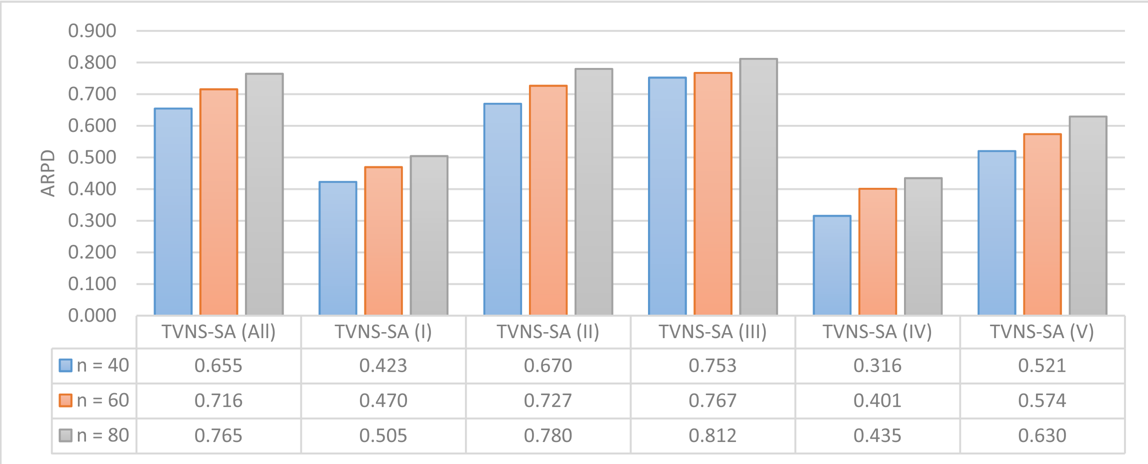


FIGURE 14. Interaction between ARPD values versus number of Jobs.

TABLE 7. Comparisons CPU Time of TVNS-SA with CPLEX.

No	Problem size N x M x R	*CPLEX* Ave.	Min.	Max.	Stdv.	*TVNS-SA* Ave.	Min.	Max.	Stdv.
1	5 x 2 x 1	14.21	9.05	23.00	6.07	1.59	1.54	1.67	0.06
2	5 x 2 x 2	111.46	13.81	351.86	160.87	1.72	1.67	1.80	0.06
3	5 x 2 x 3	11.90	5.23	18.30	5.43	1.93	1.77	2.27	0.23
4	5 x 2 x 4	14.91	9.50	24.02	6.31	1.96	1.92	1.99	0.04
5	5 x 3 x 1	9.23	5.02	18.58	6.39	2.52	2.44	2.65	0.10
6	5 x 3 x 2	16.10	7.44	23.63	6.91	2.75	2.58	3.05	0.21
7	5 x 3 x 3	14.41	9.84	24.25	6.76	2.85	2.77	2.92	0.06
8	5 x 3 x 4	17.44	8.69	35.09	12.09	3.14	3.03	3.39	0.17
9	5 x 4 x 1	8.56	4.62	10.64	2.70	3.57	3.43	3.78	0.15
10	5 x 4 x 2	11.88	4.92	19.96	6.27	4.01	3.90	4.15	0.10
11	5 x 4 x 3	56.63	7.52	139.05	61.52	4.21	4.11	4.31	0.09
12	5 x 4 x 4	9.11	4.33	13.53	4.11	4.29	4.15	4.51	0.15
13	6 x 2 x 1	56.16	40.06	71.09	16.58	2.01	1.86	2.14	0.12
14	6 x 2 x 2	42.89	23.25	86.21	29.25	2.11	2.06	2.15	0.05
15	6 x 2 x 3	97.55	25.27	245.63	101.22	2.25	2.19	2.29	0.05
16	6 x 2 x 4	418.13	25.77	1364.98	635.58	2.33	2.18	2.44	0.11
17	6 x 3 x 1	16.38	9.93	22.92	5.59	2.94	2.88	3.07	0.08
18	6 x 3 x 2	83.03	22.62	161.76	67.33	3.34	3.11	3.56	0.19
19	6 x 3 x 3	218.06	49.88	645.34	285.44	3.69	3.66	3.74	0.04
20	6 x 3 x 4	189.11	106.46	286.77	74.13	4.09	3.62	4.38	0.33
21	6 x 4 x 1	593.16	4.77	2316.35	1148.93	4.28	3.97	4.51	0.26
22	6 x 4 x 2	126.01	6.87	369.74	166.77	4.54	4.38	4.77	0.17
23	6 x 4 x 3	285.01	84.60	825.53	360.65	4.88	4.75	5.04	0.13
24	6 x 4 x 4	125.92	12.08	306.94	126.54	5.30	5.04	5.49	0.19

circumstances, Figure 14 to Figure 16 demonstrates the interaction between ARPD values for the large-scaled instances versus the number of jobs (n), number of machines (m) and number of resources (R). Where: 'TVNS-SA (All), TVNS-SA (I), TVNS-SA (II), TVNS-SA (III), TVNS-SA (IV) and TVNS-SA (V)' is the proposed algorithm with using all, first, second, Third, Fourth and Fifth neighborhoods structures for the local search, respectively. The TVNS-SA(IV) has better performance than the other Algorithms.

The RPD for the large-scaled instances is calculated as follows:

$$RPD = \frac{|Method_{sol} - Lower\ bound|}{Lower\ bound} x100$$

where '$Method_{sol}$' is the maximum completion time obtained from the different algorithms and *lower bound* is the maximum value between the two priorities in section 4.3.

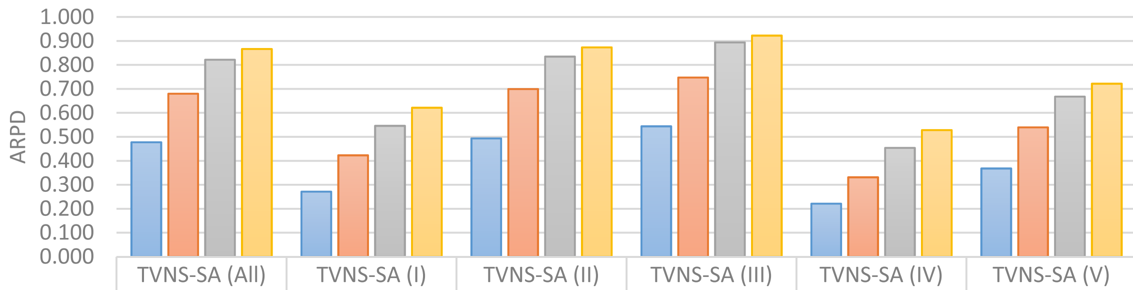
From, Figure 14 to Figure 16 the following conclusions can be drawn:

TABLE 8. Comparisons TVNS-SA (IV) with VNS (IV) for large-scaled instances.

NO	n	m	R	RPD VNS(IV) max	min	avg	std	TVNS(IV) max	min	avg	std	Time (s) VNS(IV)	TVNS(IV)
1	40	2	2	0.68	0.22	0.37	0.21	0.38	0.13	0.21	0.11	16.2	15.0
2	40	2	4	0.51	0.43	0.47	0.04	0.31	0.18	0.26	0.06	23.7	22.1
3	40	2	6	0.63	0.32	0.48	0.13	0.33	0.21	0.28	0.06	30.1	28.0
4	40	2	8	0.62	0.57	0.59	0.03	0.39	0.29	0.33	0.04	37.1	34.5
5	40	4	2	0.78	0.39	0.54	0.17	0.43	0.11	0.23	0.13	36.3	31.3
6	40	4	4	0.78	0.41	0.58	0.18	0.36	0.15	0.26	0.10	50.2	45.0
7	40	4	6	0.88	0.47	0.70	0.18	0.49	0.25	0.37	0.11	70.2	59.8
8	40	4	8	0.71	0.58	0.66	0.06	0.34	0.23	0.28	0.04	98.0	79.7
9	40	6	2	0.58	0.46	0.55	0.06	0.20	0.18	0.19	0.01	55.5	49.6
10	40	6	4	0.92	0.61	0.77	0.14	0.50	0.18	0.34	0.13	98.7	73.6
11	40	6	6	1.02	0.76	0.85	0.12	0.53	0.29	0.40	0.11	111.0	85.3
12	40	6	8	1.13	0.75	0.92	0.16	0.58	0.26	0.44	0.13	139.6	104.1
13	40	8	2	0.77	0.55	0.67	0.11	0.31	0.16	0.24	0.06	80.4	65.6
14	40	8	4	0.92	0.48	0.73	0.21	0.48	0.11	0.28	0.17	129.5	104.6
15	40	8	6	1.04	0.96	0.99	0.04	0.52	0.40	0.45	0.06	177.4	128.4
16	40	8	8	1.33	0.83	1.06	0.21	0.58	0.37	0.47	0.09	200.0	164.9
17	50	2	2	0.49	0.20	0.36	0.12	0.20	0.14	0.16	0.03	22.2	20.8
18	50	2	4	0.48	0.41	0.44	0.03	0.25	0.14	0.20	0.05	35.7	30.1
19	50	2	6	0.78	0.41	0.57	0.16	0.51	0.23	0.39	0.12	47.0	42.6
20	50	2	8	0.97	0.65	0.83	0.14	0.67	0.56	0.62	0.05	54.5	53.4
21	50	4	2	0.72	0.33	0.58	0.18	0.31	0.11	0.24	0.09	47.5	43.6
22	50	4	4	0.75	0.68	0.72	0.03	0.39	0.28	0.33	0.05	79.0	72.2
23	50	4	6	1.00	0.58	0.84	0.19	0.60	0.29	0.48	0.14	106.1	90.5
24	50	4	8	1.06	0.83	0.98	0.10	0.58	0.47	0.54	0.05	129.4	103.7
25	50	6	2	0.81	0.56	0.66	0.11	0.36	0.18	0.24	0.08	79.4	65.5
26	50	6	4	0.90	0.65	0.79	0.11	0.49	0.25	0.35	0.12	121.8	106.4
27	50	6	6	1.36	0.71	1.08	0.28	0.68	0.33	0.53	0.15	169.5	137.2
28	50	6	8	1.62	1.02	1.26	0.27	0.77	0.52	0.61	0.11	211.6	158.9
29	50	8	2	0.92	0.60	0.70	0.15	0.38	0.14	0.24	0.10	131.8	94.8
30	50	8	4	1.00	0.91	0.95	0.04	0.40	0.35	0.38	0.02	190.8	149.4
31	50	8	6	1.22	0.90	1.06	0.15	0.63	0.33	0.47	0.14	239.2	199.9
32	50	8	8	1.35	0.96	1.22	0.18	0.70	0.45	0.63	0.12	331.7	229.7
33	60	2	2	0.54	0.39	0.47	0.06	0.40	0.21	0.27	0.09	31.8	29.3
34	60	2	4	0.60	0.45	0.52	0.06	0.40	0.32	0.35	0.04	54.0	46.4
35	60	2	6	0.77	0.71	0.74	0.03	0.56	0.40	0.48	0.08	65.1	64.8
36	60	2	8	0.88	0.71	0.78	0.07	0.68	0.51	0.59	0.07	87.1	82.2
37	60	4	2	0.64	0.41	0.51	0.11	0.26	0.17	0.22	0.03	69.0	54.3
38	60	4	4	0.81	0.59	0.72	0.10	0.50	0.30	0.38	0.09	104.8	89.5
39	60	4	6	1.06	0.72	0.83	0.16	0.58	0.35	0.44	0.10	155.6	132.5
40	60	4	8	1.17	0.94	1.06	0.12	0.63	0.49	0.55	0.06	173.3	146.6
41	60	6	2	0.76	0.56	0.65	0.08	0.26	0.18	0.23	0.04	102.6	94.5
42	60	6	4	1.15	0.76	0.91	0.17	0.52	0.29	0.40	0.09	193.9	149.2
43	60	6	6	1.27	1.02	1.16	0.11	0.69	0.53	0.60	0.07	272.3	204.7
44	60	6	8	1.37	1.04	1.18	0.14	0.69	0.51	0.61	0.08	288.5	241.5
45	60	8	2	0.69	0.65	0.67	0.02	0.25	0.14	0.18	0.05	156.9	129.7
46	60	8	4	1.11	0.95	1.06	0.08	0.49	0.35	0.44	0.06	244.0	200.7
47	60	8	6	1.35	1.03	1.17	0.16	0.67	0.48	0.55	0.09	369.0	283.6
48	60	8	8	1.42	1.21	1.35	0.10	0.74	0.62	0.67	0.06	442.4	333.3

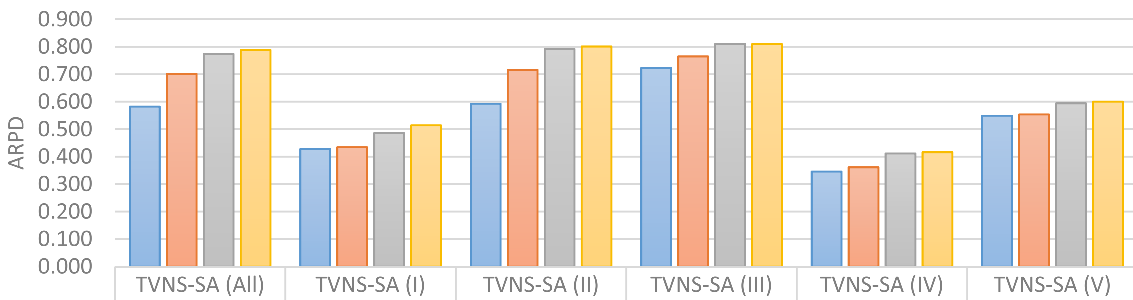
TABLE 9. **Paired t tests with 95% confidence on the makespan for all problem instances.**

paired t tests	Mean Difference (MD)	Std. Deviation	95 CI on MD	T state	Two Tailed p
TVNS-SA(All vs I)	119.94	62.83	(104.24; 135.63)	15.27	< 0.00001
TVNS-SA(All vs II)	-8.91	27.07	(-15.67; -2.14)	-2.63	0.011
TVNS-SA(All vs III)	-58.38	60.53	(-73.49; -43.26)	-7.72	< 0.00001
TVNS-SA(All vs IV)	181.25	68.74	(164.08; 198.42)	21.09	< 0.00001
TVNS-SA(All vs V)	68.25	58.72	(53.58; 82.92)	9.30	< 0.00001
TVNS-SA(I vs II)	-128.84	56.06	(-142.85; -114.84)	-18.39	< 0.00001
TVNS-SA(I vs III)	-178.31	78.39	(-197.89; -158.73)	-18.20	< 0.00001
TVNS-SA(I vs IV)	61.31	51.62	(48.42; 74.21)	9.50	< 0.00001
TVNS-SA(I vs V)	-51.69	49.80	(-64.13; -39.25)	-8.30	< 0.00001
TVNS-SA(II vs III)	-49.47	58.64	(-64.12; -34.82)	-6.75	< 0.00001
TVNS-SA(II vs IV)	190.16	67.14	(173.38; 206.93)	22.66	< 0.00001
TVNS-SA(II vs V)	77.16	53.02	(63.91; 90.40)	11.64	< 0.00001
TVNS-SA(III vs IV)	239.6	87.6	(217.7; 261.5)	21.88	< 0.00001
TVNS-SA(III vs V)	126.63	64.24	(110.58; 142.67)	15.77	< 0.00001
TVNS-SA(IV vs V)	-113.00	42.89	(-123.71; -102.29)	-21.08	< 0.00001
TVNS-SA(All vs I)	194.55	64.57	(178.42; 210.68)	24.10	< 0.00001
TVNS-SA(All vs II)	-13.70	31.77	(-21.64; -5.77)	-3.45	0.001
TVNS-SA(All vs III)	-48.88	50.42	(-61.47; -36.28)	-7.76	< 0.00001
TVNS-SA(All vs IV)	251.17	65.78	(234.74; 267.60)	30.55	< 0.00001
TVNS-SA(All vs V)	110.72	57.28	(96.41; 125.03)	15.46	< 0.00001
TVNS-SA(I vs II)	-208.25	66.76	(-224.93; -191.57)	-24.95	< 0.00001
TVNS-SA(I vs III)	-243.42	66.60	(-260.06; -226.79)	-29.24	< 0.00001
TVNS-SA(I vs IV)	56.63	33.67	(48.21; 65.04)	13.45	< 0.00001
TVNS-SA(I vs V)	-83.83	41.46	(-94.18; -73.47)	-16.18	< 0.00001
TVNS-SA(II vs III)	-35.17	50.95	(-47.90; -22.44)	-5.52	< 0.00001
TVNS-SA(II vs IV)	264.88	65.97	(248.40; 281.35)	32.12	< 0.00001
TVNS-SA(II vs V)	124.42	59.98	(109.44; 139.40)	16.60	< 0.00001
TVNS-SA(III vs IV)	300.05	64.62	(283.90; 316.19)	37.14	< 0.00001
TVNS-SA(III vs V)	159.59	49.71	(147.18; 172.01)	25.68	< 0.00001
TVNS-SA(IV vs V)	-140.45	39.90	(-150.42; -130.49)	-28.16	< 0.00001
TVNS-SA(All vs I)	256.80	66.65	(240.15; 273.45)	30.82	< 0.00001
TVNS-SA(All vs II)	-15.19	31.45	(-23.04; -7.33)	-3.86	< 0.00001
TVNS-SA(All vs III)	-48.64	68.34	(-65.71; -31.57)	-5.69	< 0.00001
TVNS-SA(All vs IV)	326.83	63.32	(311.01; 342.64)	41.29	< 0.00001
TVNS-SA(All vs V)	124.98	70.48	(107.38; 142.59)	14.19	< 0.00001
TVNS-SA(I vs II)	-271.98	67.71	(-288.90; -255.07)	-32.14	< 0.00001
TVNS-SA(I vs III)	-305.4	82.3	(-326.0; -284.9)	-29.68	< 0.00001
TVNS-SA(I vs IV)	70.03	48.09	(58.02; 82.04)	11.65	< 0.00001
TVNS-SA(I vs V)	-131.81	56.33	(-145.88; -117.74)	-18.72	< 0.00001
TVNS-SA(II vs III)	-33.45	63.91	(-49.42; -17.49)	-4.19	< 0.00001
TVNS-SA(II vs IV)	342.02	68.23	(324.97; 359.06)	40.10	< 0.00001
TVNS-SA(II vs V)	140.17	69.22	(122.88; 157.46)	16.20	< 0.00001
TVNS-SA(III vs IV)	375.5	90.5	(352.9; 398.1)	33.21	< 0.00001
TVNS-SA(III vs V)	173.63	64.99	(157.39; 189.86)	21.37	< 0.00001
TVNS-SA(IV vs V)	-201.84	68.66	(-219.00; -184.69)	-23.52	< 0.00001



	TVNS-SA (All)	TVNS-SA (I)	TVNS-SA (II)	TVNS-SA (III)	TVNS-SA (IV)	TVNS-SA (V)
R = 2	0.478	0.272	0.494	0.544	0.221	0.368
R = 4	0.680	0.423	0.700	0.748	0.332	0.540
R = 6	0.822	0.546	0.835	0.894	0.454	0.668
R = 8	0.867	0.622	0.874	0.923	0.529	0.722

FIGURE 15. Interaction between ARPD values versus number of Resources.



	TVNS-SA (All)	TVNS-SA (I)	TVNS-SA (II)	TVNS-SA (III)	TVNS-SA (IV)	TVNS-SA (V)
m = 2	0.583	0.428	0.593	0.723	0.346	0.549
m = 4	0.702	0.435	0.716	0.765	0.362	0.554
m = 6	0.774	0.486	0.792	0.811	0.412	0.594
m = 8	0.789	0.514	0.801	0.810	0.416	0.601

FIGURE 16. Interaction between ARPD values versus number of Machines.

- TVNS-SA(IV) significantly outperforms the other methods with respect to solution quality (ARPD).
- An increasing number of resources have a clearer impact on the solution quality than other parameters which are number of jobs and number of machines.

In addition, to demonstrate the effectiveness of using the first stage, the obtained results from the most significant neighborhood structure algorithm TVNS_SA(IV) were compared with the obtained results from the same algorithm after eliminate the first stage VNS_SA(IV). In Table 8 we report the average, maximum, minimum, and stander deviation of the RPD (avg, max, min, and std respectively) as well as CPU times in seconds consumed by certain methods to solve certain test instances. As Table 8 indicates, the TVNS-SA(IV) outperforms the VNS(IV) regarding both solution quality and average CPU time for almost all the instances that have been tested. That is when starting the algorithm with a good initial solution it helps to identify promising areas in the large solution space. Thus, it can be concluded that the initial heuristic improves the algorithm performance.

C. STATISTICAL ANALYSIS

Once the experimentation results have been presented, a statistical test has been made to test rigorously and fairness of the performances of the five variants of the TVNS-SA. The significant differences between the five variants of the TVNS_SA can be detected using paired t-tests.

The results of the paired t-tests with the individual error rate (0.05) were summarized in Table 9.

In summary, paired t-tests results reveal a significant difference with $p < 0.00001$ for all the comparisons made among the performance of the fife variants of the TVNS-SA.

VI. CONCLUSION

This paper addressed the unrelated parallel machines scheduling problem subject to multiple limited renewable resources, sequence-dependent setup times and release date constraints. This study presents a new MILP to minimize the maximum completion time (makespan). Despite the currently available technologies, the computation time for the MILP for most of the small-scaled instances is long and it is unacceptable. As a result, MILP becomes ineffective for large-size problems. Approximation approaches can significantly reduce the computation time without necessarily leading to the optimal solution. This paper introduced a different variant of TVNS-SA metaheuristic approach to achieve that goal. At the first stage, an initial solution was obtained based on two basic dispatching rules start with assigning each job to the fastest machine then arrange the sequence for each machine according to the earliest release date. At the second stage, hybrid variable neighborhood search with simulated annealing are designed.

To generate diverse solutions in the quickest and easiest way possible, five types of local search techniques were constructed and used in the TVNS-SA. Random instances were generated to test the efficiency and effectiveness of TVNS-SA. The performances of TVNS-SA algorithm compared with CPLEX and Variable Neighborhood Search (VNS) algorithms. The results revealed that the proposed algorithm obtained the optimal solution for most of the small-scaled instances and it's outperformed CPLEX on all small-scaled instances with regard to CPU time. Regarding the most effective local search technique for the addressed problem, it was found that the developed TVNS-SA (IV) outperformed other variants of TVNS-SA with respect to the conducted ARPD and CPU time. Finally, paired t-tests with individual error rate (0.05) were applied to test the rigorous and the fairness of the five variants of the TVNS-SA performances.

Although the results presented in this work have demonstrated the effectiveness of TVNS-SA. However, there remain several research directions worth a thorough investigation. Despite the good results achieved by the TVNS-SA, it is still possible to improve results by using optimization methods with adopting some useful knowledge in the algorithms. It is interesting to study the addressed problem by taking maintenance intervention dates or machine unavailability constraints into consideration. Consider green manufacturing constraints will be challenging.

REFERENCES

[1] M. C. Vélez-Gallego, J. Maya, and J. R. Montoya-Torres, "A beam search heuristic for scheduling a single machine with release dates and sequence dependent setup times to minimize the makespan," *Comput. Oper. Res.*, vol. 73, pp. 132–140, Sep. 2016.

[2] M. Afzalirad and J. Rezaein, "A realistic variant of bi-objective unrelated parallel machine scheduling problem: NSGA-II and MOACO approaches," *Appl. Soft Comput.*, vol. 50, pp. 109–123, Jan. 2017.

[3] A. Bitar, S. Dauzère-Pérès, C. Yugma, and R. Roussel, "A memetic algorithm to solve an unrelated parallel machine scheduling problem with auxiliary resources in semiconductor manufacturing," *J. Scheduling*, vol. 19, no. 4, pp. 367–376, 2016.

[4] M. Afzalirad and J. Rezaeian, "Resource-constrained unrelated parallel machine scheduling problem with sequence dependent setup times, precedence constraints and machine eligibility restrictions," *Comput. Ind. Eng.*, vol. 98, pp. 40–52, Aug. 2016.

[5] X.-L. Zheng and L. Wang, "A two-stage adaptive fruit fly optimization algorithm for unrelated parallel machine scheduling problem with additional resource constraints," *Expert Syst. Appl.*, vol. 65, pp. 28–39, Dec. 2016.

[6] J. K. Lenstra, D. B. Shmoys, and É. Tardos, "Approximation algorithms for scheduling unrelated parallel machines," *Math. Program.*, vol. 46, nos. 1–3, pp. 259–271, 1990.

[7] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*. New York, NY, USA: Springer-Verlag, 2012.

[8] J. Blazewicz, K. H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz, *Scheduling Computer and Manufacturing Processes*. Springer, 2013.

[9] E. B. Edis, C. Oguz, and I. Ozkarahan, "Parallel machine scheduling with additional resources: Notation, classification, models and solution methods," *Eur. J. Oper. Res.*, vol. 230, no. 3, pp. 449–463, 2013.

[10] P. Györgyi and T. Kis, "Approximability of scheduling problems with resource consuming jobs," *Ann. Oper. Res.*, vol. 235, no. 1, pp. 319–336, 2015.

[11] T. Kis, "Approximability of total weighted completion time with resource consuming jobs," *Oper. Res. Lett.*, vol. 43, no. 6, pp. 595–598, 2015.

[12] E. Hebrard, 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.

[13] P. Györgyi and T. Kis, "Approximation schemes for parallel machine scheduling with non-renewable resources," *Eur. J. Oper. Res.*, vol. 258, no. 1, pp. 113–123, 2017.

[14] P. Györgyi, "A PTAS for a resource scheduling problem with arbitrary number of parallel machines," *Oper. Res. Lett.*, vol. 45, no. 6, pp. 604–609, 2017.

[15] X.-L. Zheng and L. Wang, "A collaborative multiobjective fruit fly optimization algorithm for the resource constrained unrelated parallel machine green scheduling problem," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 5, pp. 790–800, May 2018.

[16] A. A. Qamhan and I. M. Alharkan, "Note on 'A two-stage adaptive fruit fly optimization algorithm for unrelated parallel machine scheduling problem with additional resource constraints,'" *Expert Syst. Appl.*, vol. 128, pp. 81–83, Aug. 2019.

[17] M. Afzalirad and M. Shafipour, "Design of an efficient genetic algorithm for resource-constrained unrelated parallel machine scheduling problem with machine eligibility restrictions," *J. Intell. Manuf.*, vol. 29, no. 2, pp. 423–437, 2018.

[18] K. Li, S.-L. Yang, J. Y.-T. Leung, and B.-Y. Cheng, "Effective metaheuristics for scheduling on uniform machines with resource-dependent release dates," *Int. J. Prod. Res.*, vol. 53, no. 19, pp. 5857–5872, 2015.

[19] F. Villa, E. Vallada, and L. Fanjul-Peyro, "Heuristic algorithms for the unrelated parallel machine scheduling problem with one scarce additional resource," *Expert Syst. Appl.*, vol. 93, pp. 28–38, Mar. 2018.

[20] M. A. Abdeljaoued, N. El Houda Saadani, and Z. Bahroun, "Heuristic and metaheuristic approaches for parallel machine scheduling under resource constraints," *Oper. Res.*, pp. 1–24, Jun. 2018, doi: 10.1007/s12351-018-0412-3.

[21] S. Özpeynirci, B. Gökgür, and B. Hnich, "Parallel machine scheduling with tool loading," *Appl. Math. Model.*, vol. 40, no. 9, pp. 5660–5671, 2016.

[22] M. G. Furugyan, "Optimal correction of execution intervals for multiprocessor scheduling with additional resource," *J. Comput. Syst. Sci. Int.*, vol. 54, no. 2, pp. 268–277, 2015.

[23] G. Dosa, H. Kellerer, and Z. Tuza, "Restricted assignment scheduling with resource constraints," *Theor. Comput. Sci.*, vol. 760, pp. 72–87, Feb. 2019.

[24] Y.-C. Wang, M.-J. Wang, and S.-C. Lin, "Selection of cutting conditions for power constrained parallel machine scheduling," *Robot. Comput.-Integr. Manuf.*, vol. 43, pp. 105–110, Feb. 2017.

[25] W. Labbi, M. Boudhar, and A. Oulamara, "Scheduling two identical parallel machines with preparation constraints," *Int. J. Prod. Res.*, vol. 55, no. 6, pp. 1531–1548, 2017.

[26] J. Li, P. Duan, H. Sang, S. Wang, Z. Liu, and P. Duan, "An efficient optimization algorithm for resource-constrained steelmaking scheduling problems," *IEEE Access*, vol. 6, pp. 33883–33894, 2018.

[27] A. Allahverdi, J. N. Gupta, and T. Aldowaisan, "A review of scheduling research involving setup considerations," *Omega*, vol. 27, pp. 219–239, Apr. 1999.

[28] A. Allahverdi, C. T. Ng, T. C. E. Cheng, and M. Y. Kovalyov, "A survey of scheduling problems with setup times or costs," *Eur. J. Oper. Res.*, vol. 187, no. 3, pp. 985–1032, 2008.

[29] A. Allahverdi, "The third comprehensive survey on scheduling problems with setup times/costs," *Eur. J. Oper. Res.*, vol. 246, no. 2, pp. 345–378, 2015.

[30] C. Koulamas and G. J. Kyparisis, "Single-machine scheduling problems with past-sequence-dependent setup times," *Eur. J. Oper. Res.*, vol. 187, no. 3, pp. 1045–1049, 2008.

[31] M. Gendreau, G. Laporte, and E. M. Guimarães, "A divide and merge heuristic for the multiprocessor scheduling problem with sequence dependent setup times," *Eur. J. Oper. Res.*, vol. 133, no. 1, pp. 183–189, 2001.

[32] A. S. Mendes, F. M. Müller, P. M. França, and P. Moscato, "Comparing meta-heuristic approaches for parallel machine scheduling problems," *Prod. Planning Control*, vol. 13, no. 2, pp. 143–154, 2002.

[33] 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.

[34] A. E. Ezugwu and F. Akutsah, "An improved firefly algorithm for the unrelated parallel machines scheduling problem With sequence-dependent setup times," *IEEE Access*, vol. 6, pp. 54459–54478, 2018.

[35] L. Meng, C. Zhang, X. Shao, B. Zhang, Y. Ren, and W. Lin, "More MILP models for hybrid flow shop scheduling problem and its extended problems," *Int. J. Prod. Res.*, 2019, doi: 10.1080/00207543.2019.1636324.

[36] B. Naderi, S. Gohari, and M. Yazdani, "Hybrid flexible flowshop problems: Models and solution methods," *Appl. Math. Model.*, vol. 38, pp. 5767–5780, Dec. 2014.

[37] M. X. Weng, J. Lu, and H. Ren, "Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective," *Int. J. Prod. Econ.*, vol. 70, no. 3, pp. 215–226, 2001.

[38] Y.-K. Lin and F.-Y. Hsieh, "Unrelated parallel machine scheduling with setup times and ready times," *Int. J. Prod. Res.*, vol. 52, no. 4, pp. 1200–1214, 2014.

[39] S. Emami, G. Moslehi, and M. Sabbagh, "A Benders decomposition approach for order acceptance and scheduling problem: A robust optimization approach," *Comput. Appl. Math.*, vol. 36, no. 4, pp. 1471–1515, 2017.

[40] A. Obeid, S. Dauzère-Pérès, and C. Yugma, "Scheduling job families on non-identical parallel machines with time constraints," *Ann. Oper. Res.*, vol. 213, no. 1, pp. 221–234, Feb. 2014.

[41] J. R. Zeidi and S. MohammadHosseini, "Scheduling unrelated parallel machines with sequence-dependent setup times," *Int. J. Adv. Manuf. Technol.*, vol. 81, nos. 9–12, pp. 1487–1496, 2015.

[42] G. Rabadi, R. J. Moraga, and A. Al-Salem, "Heuristics for the unrelated parallel machine scheduling problem with setup times," *J. Intell. Manuf.*, vol. 17, no. 1, pp. 85–97, Feb. 2006.

[43] A. Hamzadayi and G. Yildiz, "Modeling and solving static *m* identical parallel machines scheduling problem with a common server and sequence dependent setup times," *Comput. Ind. Eng.*, vol. 106, pp. 287–298, Apr. 2017.

[44] G. Bektur and T. Sarac, "A mathematical model and heuristic algorithms for an unrelated parallel machine scheduling problem with sequence-dependent setup times, machine eligibility restrictions and a common server," *Comput. Oper. Res.*, vol. 103, pp. 46–63, Mar. 2019.

[45] M. A. Qamhan, A. A. Qamhan, I. M. Al-Harkan, and Y. A. Alotaibi, "Mathematical modeling and discrete firefly algorithm to optimize scheduling problem with release date, sequence-dependent setup time, and periodic maintenance," *Math. Problems Eng.*, vol. 2019, Aug. 2019, Art. no. 8028759.

[46] N. Mladenović and P. Hansen, "Variable neighborhood search," *Comput. Oper. Res.*, vol. 24, no. 11, pp. 1097–1100, Nov. 1997.

[47] C. Charalambous, K. Fleszar, and K. S. Hindi, "A hybrid searching method for the unrelated parallel machine scheduling problem," in *Proc. IFIP Int. Conf. Artif. Intell. Appl. Innov. (AICT)*, vol. 339, 2010, pp. 230–237.

[48] K. Fleszar, C. Charalambous, and K. S. Hindi, "A variable neighborhood descent heuristic for the problem of makespan minimisation on unrelated parallel machines with setup times," *J. Intell. Manuf.*, vol. 23, no. 5, pp. 1949–1958, 2012.

[49] A. Bilyk and L. Mönch, "A variable neighborhood search approach for planning and scheduling of jobs on unrelated parallel machines," *J. Intell. Manuf.*, vol. 23, no. 5, pp. 1621–1635, 2012.

[50] M. A. Cruz-Chávez, A. Martínez-Oropeza, J. del Carmen Peralta-Abarca, M. H. Cruz-Rosales, and M. Martínez-Rangel, "Variable neighborhood search for non-deterministic problems," in *Artificial Intelligence and Soft Computing* (Lecture Notes in Computer Science), vol. 8468. Cham, Switzerland: Springer, 2014, pp. 468–478.

[51] M. Yazdani, S. M. Khalili, M. Babagolzadeh, and F. Jolai, "A single-machine scheduling problem with multiple unavailability constraints: A mathematical model and an enhanced variable neighborhood search approach," *J. Comput. Des. Eng.*, vol. 4, no. 1, pp. 46–59, 2017.

[52] Z. Sevkli and F. E. Sevilgen, "A Hybrid Particle Swarm Optimization Algorithm for Function Optimization," in *Proc. Workshops Appl. Evol. Comput.* Berlin, Germany: Springer, 2008, pp. 585–595.

IBRAHIM M. AL-HARKAN received the B.S. degree in industrial engineering from King Saud University, Saudi Arabia, and the M.S. and Ph.D. degrees in industrial engineering from The University of Oklahoma, USA. He was the Chairman for the Mechanical Engineering Department for three and half years, also the Chairman for the Industrial Engineering Department for two years. He was the Dean for graduate studies for five years. He was also the Vice-Rector Assistant for Graduate Studies and Scientific Research for Knowledge Exchange and Technology Transfer, for one year. He was the General Director of the External Joint Supervision Program (EJSP), for ten years. He is also is the General Director of the Entrepreneurship Institute. He is currently one of the founder of the Saudi Industrial Engineering Society (SIES) and also the Vice President of the society, one of the founders of the Industrial Engineering Council, Saudi Council of Engineers, one of the founders of the Saudi Chapter of the Institute of Industrial engineers, and established the Department of Industrial Engineering, College of Engineering, King Saud University, where he is also an Associate Professor of industrial engineering. His specialized areas are production planning and control, modeling and simulation of industrial and service systems, and applied operations research.

AMMAR A. QAMHAN received the B.Sc. degree in industrial engineering and manufacturing system from the Faculty of Engineering and Information Technology, Taiz University, in 2011. He is currently pursuing the M.Sc. degree in industrial engineering with King Saud University. He was a former Engineer with Hayel Saeed Anam (HSA) Head Office–Technical Auditing Yemen Region, from 2012 to 2014. His main research interests include optimization and scheduling.