Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 1856734, 9 pages
https://doi.org/10.1155/2021/1856734

Research Article

Tabu Search Algorithm Based on Lower Bound and Exact Algorithm Solutions for Minimizing the Makespan in Non-Identical Parallel Machines Scheduling

Mohammed A. Noman,[1] **Moath Alatefi**,[1] **Abdulrahman M. Al-Ahmari**,[1,2] **and Tamer Ali**[1]

[1]*Industrial Engineering Department, College of Engineering, King Saud University, Riyadh-11421, Saudi Arabia*
[2]*Raytheon Chair for Systems Engineering (RCSE Chair), King Saud University, P.O. Box 800, Riyadh 11421, Saudi Arabia*

Correspondence should be addressed to Mohammed A. Noman; mmohammed1@ksu.edu.sa and Moath Alatefi; malatefi@ksu.edu.sa

Received 26 October 2021; Revised 10 November 2021; Accepted 11 November 2021; Published 25 December 2021

Academic Editor: Jiafu Su

Recently, several heuristics have been interested in scheduling problems, especially those that are difficult to solve via traditional methods, and these are called NP-hard problems. As a result, many methods have been proposed to solve the difficult scheduling problems; among those, effective methods are the tabu search algorithm (TS), which is characterized by its high ability to adapt to problems of the large size scale and ease of implementation and gives solution closest to the optimum, but even though those difficult problems are common in many industries, there are only a few numbers of previous studies interested in the scheduling of jobs on unrelated parallel machines. In this paper, a developed TS algorithm based on lower bound (LB) and exact algorithm (EA) solutions is proposed with the objective of minimizing the total completion time (makespan) of jobs on nonidentical parallel machines. The given solution via EA was suggested to enhance and assess the solution obtained from TS. Moreover, the LB algorithm was developed to evaluate the quality of the solution that is supposed to be obtained by the developed TS algorithm and, in addition, to reduce the period for searching for the optimal solution. Two numerical examples from previous studies from the literature have been solved using the developed TS algorithm. Findings show that the developed TS algorithm proved its superiority and speed in giving it the best solution compared to those solutions previously obtained from the literature.

1. Introduction

Scheduling plays an essential role in the management of production. The scheduling dictates what will be done, where, and with what resources [1]. Production scheduling is an important decision-making process at the operational level and is a difficult problem depending on the number of calculations needed to get a schedule that will maximize the criterion chosen. Many scheduling issues exist in modern manufacturing environments. There are many shops in the factory, depending on the machine configuration and workflow [2]. A flow shop is a shop in which machines are organized in series; jobs start processing on an initial machine, cross several intermediate machines, and end in a final machine. In a job shop, jobs can be done in any order on machines. In other words, no unique restriction of machine order is enforced.

Machine scheduling is one of the most widely researched fields in the research literature on operations, and it covers a wide range of issues [3]. The issue discussed in this paper is that independent jobs should be scheduled on nonidentical parallel machines to minimize the total completion time of the job "makespan." There are n separate jobs, each with its own processing time and can be processed on any m parallel machine that is nonidentical. Each of these unrelated machines has a different speed as S_i. Makespan ($C_{\max}$) is the

completion time for the last job leaving the system for scheduling issues. Typically, small C_{max} means a high use. Reducing C_{max} should also result in a higher performance level [4, 5].

There are m machines and n jobs which need to be handled in an organized way. Jobs may not need all m machines and may need to be visited by those machines more than once. Only one machine is able to perform a specific task. In reality, many copies of the same machine are always present (parallel machine). There are also several machines that can perform the work. Scheduling determines the number of jobs completed. The requisite inputs are a list of operations that must be performed, the time it takes for each operation, and a list of any prior restrictions that define the operations that must be followed by others. If jobs have no precedence constraints, it is named as an open shop. In a general job shop, there are two scheduling types: for the first one, for a single machine and n jobs, there are $n!$ choices of possible schedules. For the second type, if there are m machines and n jobs, the number of possible schedules is $(n!)$ m [2]. Therefore, the scheduling problem is more difficult to be solved via dispatching rules (LPT, SPT, EDD, etc.), and these are called NP-hard problems [6, 7]. With traditional optimization approaches, it is very difficult to achieve an optimum solution. Mathematical optimizations can provide an optimal solution for a reasonable size problem, but their implementation is limited in the case of a large-scale problem. Dispatch rules can be only adapted for small-scale problems, and there are no good results in different problems with a single dispatching rule [8]. Consequently, many previous studies' effort centered on heuristic methods to solve complex problems.

In this study, the tabu search (TS) algorithm has been developed to solve like these complex problems based on the exact (EA) and lower bound (LB) algorithms. The reason for presenting TS for minimizing the makespan in nonidentical parallel machines' scheduling in this paper is that the solutions of the proposed method depend on the value of the lower bound to stop, unlike the previous TS methods, in which the solution iterations are predetermined to stop, even if they do not find the optimum solution. Also, it is known that the exact algorithm (EA) cannot solve the large scheduling problems. Unlike the proposed approach, it can solve big scheduling problems quickly.

2. Literature Review

In this section, prior research studies related to the scheduling of nonidentical parallel machines and related observations are summarized. Fanjul-Peyro and Ruiz [9] proposed a set of simple iterated greedy local search-based metaheuristics for nonidentical parallel machines' scheduling with the objective to minimize the makespan, which produce solutions of very good quality in a very short amount of time. In the study of Vallada and Ruiz [10], a genetic algorithm (GA) was introduced to the nonrelated parallel scheduling machines with considered machine and job sequence setup times. The proposed GA includes a simple local search and a crossover operator enhanced with local search. The proposed method produced good results as a model for large-scale and small-scale experiments. Fanjul-Peyro and Ruiz [11] proposed mixed-integer programming mathematical formulations via two generalizations for scheduling the nonidentical parallel machines' problem for makespan minimization: the first is a condition in which not every parallel machine available should be used and only a subset of parallel machines should be used; the second is no responsibility to manage all jobs available. Comprehensive tests show that the two suggested algorithms boost the results substantially. Balin [12] proposed GA and fuzzy processing time simulation for scheduling a set of jobs on unrelated parallel machines. The results were compared with those obtained from when using the longest processing time (LPT) rule. The proposed approach showed good results and achieved them easily and multiple times in one sprint. Joo and Kim [13] proposed two metaheuristics; these are GA and a new evolutionary metaheuristic population-based algorithm called self-evolution to assess job allocation and scheduling of nonidentical parallel machines to reduce the makespan. The efficiency of the metaheuristic algorithms is evaluated by using randomly generated several examples to compare with optimal solutions.

Fleszar et al. [14] developed a variable neighborhood descent search algorithm hybridized with mathematical programming elements for minimizing the makespan for scheduling jobs on unrelated parallel machines. The study of Joo and Kim [15] proposed hybrid GA with dispatching rules for scheduling a set of jobs on nonidentical parallel machines with sequence- and machine-dependent setup times and machine-dependent processing times for minimizing the makespan. Geyik and Elibal [16] proposed a fuzzy linguistic approach for considering the learning effect for solving nonidentical parallel processor scheduling under uncertain processing times. Arroyo and Leung [17] developed a lower bound for the problem and proposed a mixed-integer programming model and several heuristics based on first-fit and best-fit earliest job-ready time rules to minimize the makespan in unrelated parallel batch-processing machines for scheduling nonidentical job sizes and unequal ready times. Tan et al. [18] proposed a hybrid GA and greedy strategy for green batch scheduling on unrelated parallel machines to minimize total electricity costs in production.

Liu et al. [19] developed a mixed-variable differentiate evolution (MVDE) as the scheduling algorithm for coordinated charging scheduling of electric vehicles. Zhou et al. [20] proposed a self-adaptive differential evolution algorithm for scheduling a single batch-processing machine with arbitrary job sizes and release times. The study goal is to assign jobs into batches without breaking the machine capacity constraint and then sort the batches to minimize the makespan. Zhao et al. [21] proposed a self-learning discrete Jaya algorithm for multiobjective energy-efficient distributed no-idle flow-shop scheduling problem in the heterogeneous factory system. Zhao et al. [22] proposed a two-stage cooperative evolutionary algorithm with problem-specific knowledge for energy-efficient scheduling of the no-wait flow-shop problem.

Many heuristics have been proposed such as branch and bound (B&B), simulated annealing (SA), genetic algorithm (GA), tabu search (TS), and neural network (NN). Among these different approaches to various scheduling problems, lately, there is an increasing interest in applying TS because of its high ability to give a solution that is very close to the optimal solution and its ease of implementation. Glover [23] implemented the TS algorithm, and since then, it has become very popular in solving the optimization issues. Many studies such as [24–26] have shown that the TS will yield higher timetables than others. TS approach uses the memory to store search-related information that helps avoid cycles (previous solutions that were visited could be reselected). Tabu list is the name of the memory used. The tabu list is updated for every TS iteration. Due to the limitations in the tabu list, the risk of rejecting solutions not yet produced may occur. Tabu solutions for a condition that is called aspiration criteria are then reviewed. If the tabu solution has a better objective value than the best objective value so far found, it is then acknowledged and excluded from the tabu list. This tabu solution will be removed from the tabu list [27]. TS algorithm has different elements that must be tuned before the search process is initiated. The primary solution, tabu list, neighboring structure, aspiration criteria, and stop criteria are the principal features of the TS. The initial TS solution was developed by using the LPT rule in which jobs in the nonincreasing order of processing time are sequenced. The two classic and useful operators for constructing promising neighborhood structures are parity interchange and reverse reintegration [28]. The pairwise interchange operator was implemented in the proposed TS algorithm for generating new candidate solutions. The importance of the metaheuristic parameters has an effect on its efficiency. In general, the main difficulties in TS include tabu list design, list management mechanism, and nonprohibited selection of the move [29]. This is not an easy job for certain optimization issues. In addition, TS may be very sensitive to certain parameters, such as tabu list size [27]. There are many studies that have good applications of TS which can be found in [23, 30–37].

3. Problem Description

In this study, a number of n jobs on m nonidentical parallel machines are to be processed. Each machine (i) can process just one job (j) at a time. Each job should be handled continuously without any interruption. Put $P(i, j)$ as the processing time of job j on machine i. The objective is to make the maximum makespan as minimized as possible.

$x(i, j)$ is the variable which determines whether job j is processed by machine i (if $x(i, j) = 1$) or not (if $x(i, j) = 0$). The matrix X is composed of variables $x(i, j)$, and it has the following properties [38]:

(1) All elements are equal to "0" or "1," $x(i, j) \in \{0, 1\}$.

(2) Each column has only one element valued "1."

$$\sum_{i=1}^{m} x(i, j) = 1, \quad j = 1, \ldots, n. \tag{1}$$

(3) The number of elements valued "1" is n.

$$\sum_{j=1}^{n} \sum_{i=1}^{m} x(i, j) = n. \tag{2}$$

In equation (3), the processing time j on different machines is expressed as follows:

$$P(i_1, j) \times S_1 = P(i_2, j) \times S_2, \tag{3}$$

where S_1 is the speed of machine 1 and S_2 is the speed of machine 2.

Equation (4) expresses on the makespan of the jobs:

$$C_{\max} = \max_{i=1}^{m} \left\{ \sum_{j=1}^{n} x(i, j) \times P(i, j) \right\}. \tag{4}$$

Thus, the objective function may be defined as follows:

$$\min C_{\max}. \tag{5}$$

4. The Proposed Methodology

The proposed methodology of this study consists of three phases. In the first phase, we proposed an exact algorithm to be used as a base of comparison to measure the effectiveness of the next two phases. Then, the second phase established a lower bound that judges the optimality of the next heuristic algorithm. Also, it helps in identifying the stopping criteria of the search. Finally, tabu search algorithm is used to search for the best and optimal solution depending on the proposed lower bound.

4.1. Scheduling Using the Exact Algorithm (EA). This section discusses the exact algorithm that is intended to calculate optimal total completion time as the following:

(1) Decision variables:

$$x(i, j) = \begin{cases} 1 & \text{if job } j \text{ is processed on machine } i, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

(2) Objective function:

$$\text{Min} C_{\max}. \tag{7}$$

(3) Constraints:

$$C_{max} \geq \sum_{j=1}^{n} P(i, j) \times x(i, j), \quad \forall i \in 1, \ldots, m, \tag{8}$$

$$\sum_{i=1}^{m} x(i, j) = 1, \quad \forall j \in 1, \ldots, n, \tag{9}$$

Put $LB_3 \leftarrow$ (smallest overall processing time of the jobs/total machines).
$LB_2 \leftarrow LB_3$
While $LB_2 \leq LB_3$ **do**
Remove jobs that have the lowest $\varepsilon(j)$ percentage with preemption from machines that have makespan greater than LB_2 to their second-fastest machines.
$LB_3 \leftarrow$ (new minimum overall processing time of the jobs/total machines).
$LB_2 = LB_2 + 1$
end while
$LB_2 = LB_2 - 1$

Algorithm 1: The algorithm to calculate LB_2.

Choose the initial solution using LPT rule s_0
Initialize memory structures
Repeat
Generate a set "A"of nontabu solutions $s \in N(s_0)$
s = best solution of A
Update memory structures
If $f(s) < f(s_0)$, then $s_0 = s$
Until stopping criterion true
s_0 is an approximate solution to the optimal solution

Algorithm 2: TS algorithm template.

$$x(i, j) \in [0, 1]. \tag{10}$$

The objective function shown in equation (7) is to minimize the maximum completion time for all the jobs. Constraint (8) calculates the completion time for each job. Constraint (9) guarantees that each job is assigned precisely to one machine. Constraint (10) is a binary decision variable.

4.2. Lower Bound (LB). In this section, the lower bounds LB_1 and LB_2 that are used to evaluate the quality of the heuristic method that is used to find the optimal makespan are proposed. Set, for $1 \leq j \leq n$, $D_j = \min_i = 1, \ldots, m\{p_{ij}\}$, and LB_1 is defined as a lower bound for the problem, $LB_1 = \sum_{j=1}^{n} D_j/n$, where D_j is the smallest processing time of each job. It is worth noting that every machine is busy processing some jobs. Next, define $\varepsilon(j)$ as an efficiency of the second smallest processing time of a job (j); therefore, $\varepsilon(j) =$ second smallest processing time
c of job j/smallest processing time of job j. Then, the lower bound LB_2 is iteratively calculated in order to improve the lower bound LB_1 as described in Algorithm 1 [39].

The objective is to increase the overall processing time from the likely minimum value step by step. LB_3 is the new minimum total processing time of the jobs over the total machines, which is intuitively a lower bound. With this algorithm, the total processing time is increased each time; this increase is the minimum possible. While $LB_2 \leq LB_3$, the value of LB_2 will remain as a lower bound because it is less than the possible minimum makespan at that point of time. Only when $LB_2 > LB_3$, the last LB_2 value will take as the lower bound.

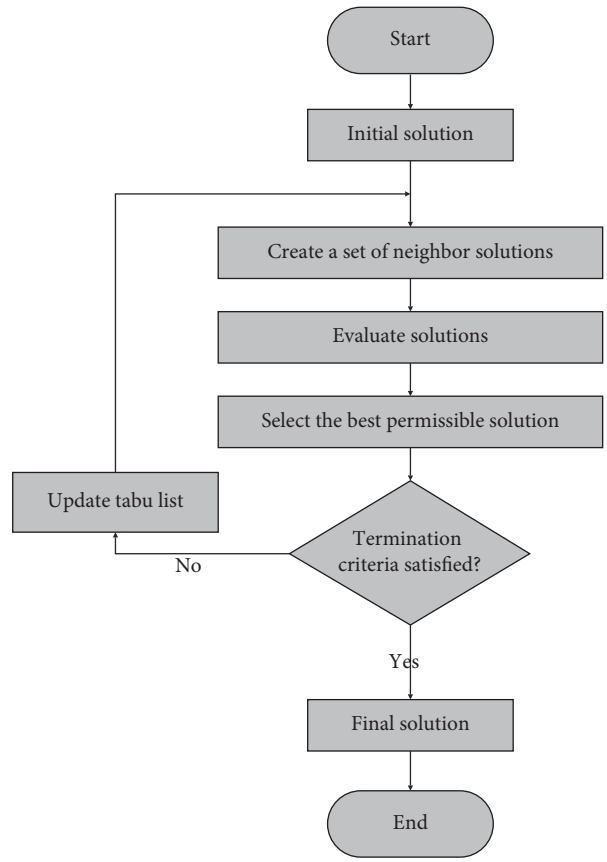

Figure 1: General structure of TS.

Table 1: Nine jobs and four nonidentical parallel machines' scheduling [5].

Job	Machine			
	M1	M2	M3	M4
Job 1	18	9	4.5	3.6
Job 2	14	7	3.5	2.8
Job 3	24	12	6	4.8
Job 4	30	15	7.5	6
Job 5	16	8	4	3.2
Job 6	20	10	5	4
Job 7	22	11	5.5	4.4
Job 8	26	13	6.5	5.2
Job 9	14	7	3.5	2.8

Table 2: Scheduling with GA [5].

Machines	Scheduled jobs	C_i (min.)
M1	Job 2	14
M2	Job 1, job 9	16
M3	Job 5, job 7, job 3	15.5
M4	Job 6, job 8, job 4	15.2

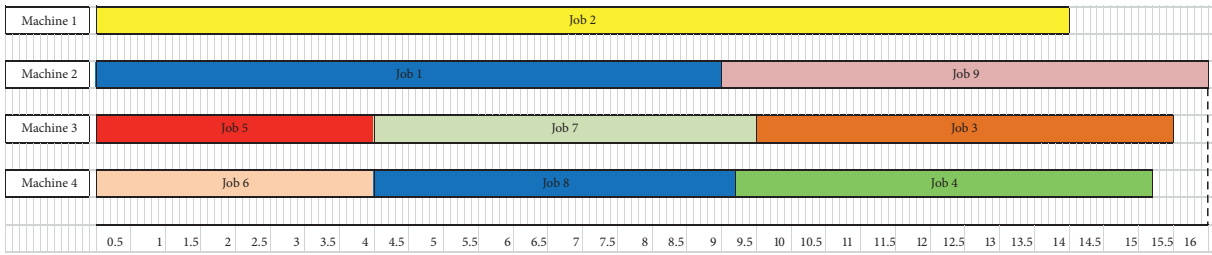

Figure 2: Jobs' scheduling using GA [5].

4.3. Scheduling Using Tabu Search (TS). This section discusses the TS algorithm that is intended to calculate optimal total completion time and shows how to select the parameter values for the TS algorithm.

Parameter tuning is an important problem in the design of metaheuristics. To this end, test runs for a wide range of parameter values for the TS algorithm have been performed, and the best configuration has been selected and described as follows:

(1) Based on preliminary experiments, the number of candidates produced was 70

(2) There were three elements of the tabu list size, and an update to the tabu list was performed using the first-in-first-out (FIFO) technique

(3) The iterations have been designed to be 5000; the iterations stop if the tabu solutions equal with the lower bound

(4) The criteria for ambition have been defined as described previously (if a step leads to a better solution than the best one ever)

In general, the TS algorithm can be defined as described in Algorithm 2.

Firstly, the general structure of TS shows an initial solution of the problem using the LPT method and then creates a set of neighbor solutions; after that, the evaluation of the solutions is done using LB, and the best permissible solution is selected as described in Figure 1.

5. Results and Discussion

In this section, the scheduling problem of n jobs on m nonidentical machines is solved using EA and TS algorithms described above. As a numerical example in the study of Balin [5], nine jobs are scheduled on four nonidentical machines. The objective is to minimize makespan ($C_{\max}$). Problem data are summarized in Table 1.

Table 2 shows the scheduling of the nine jobs on the four nonidentical machines problem using GA proposed by Balin [5] and shows that the makespan obtained is 16 minutes. Gantt chart for scheduling every job on the machines is shown in Figure 2.

Tables 3 and 4 show the same scheduling solution of the same problem proposed by Balin [5] and show the best makespan using EA and TS which is only 15.6 minutes.

Gantt chart for job scheduling every job on the machines using EA or TS is shown in Figure 3. It is observed that the

Table 3: Scheduling with EA.

Machines	Scheduled jobs	C_i (min.)
M1	Job 2	14
M2	Job 4	15
M3	Job 1, job 3, job 6	15.5
M4	Job 5, job 7, job 8, job 9	15.6

Table 4: Scheduling with TS.

Machines	Scheduled jobs	C_i (min.)
M1	Job 2	14
M2	Job 4	15
M3	Job 1, job 3, job 6	15.5
M4	Job 5, job 7, job 8, job 9	15.6

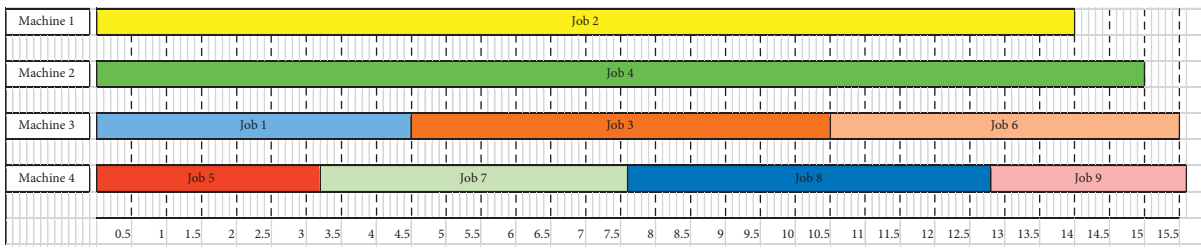

Figure 3: Jobs' scheduling using EA and TS.

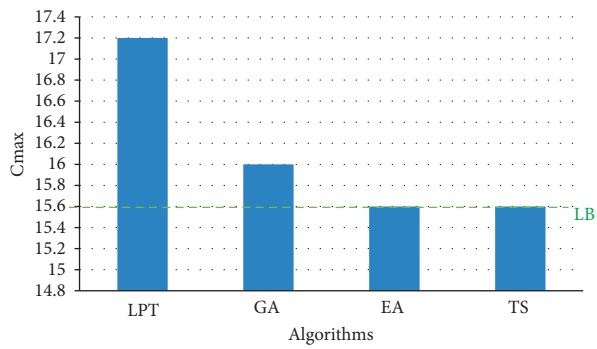

Figure 4: Histogram of averages of makespan means for $n = 9$ and $m = 4$.

Table 5: Thirty jobs and five nonidentical parallel machines' scheduling [40].

Job	Machine					Job	Machine				
	M1	M2	M3	M4	M5		M1	M2	M3	M4	M5
1	50	38	43	49	47	16	50	42	30	42	50
2	46	42	30	49	50	17	45	32	48	34	33
3	44	35	43	36	37	18	47	40	35	38	46
4	38	43	44	48	48	19	48	45	44	34	44
5	39	39	47	36	49	20	43	41	33	48	44
6	43	42	35	50	38	21	33	44	49	36	33
7	38	42	38	39	30	22	48	42	47	35	30
8	34	35	37	34	33	23	32	47	32	34	42
9	47	48	41	39	42	24	42	32	42	45	36
10	32	30	34	38	31	25	34	48	35	39	41
11	41	38	37	48	33	26	38	39	43	50	49
12	37	38	48	49	31	27	38	46	45	40	31
13	40	41	44	45	45	28	37	36	44	40	48
14	43	40	47	37	40	29	31	32	42	39	44
15	50	42	44	39	40	30	35	46	44	38	33

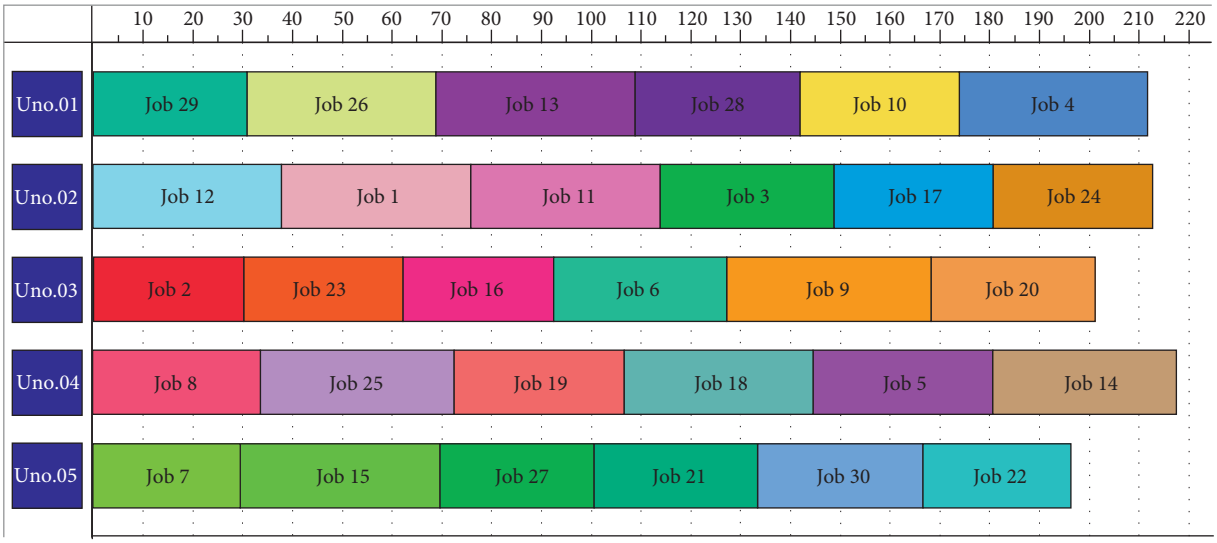

Figure 5: Gantt chart for job scheduling using ABC [40].

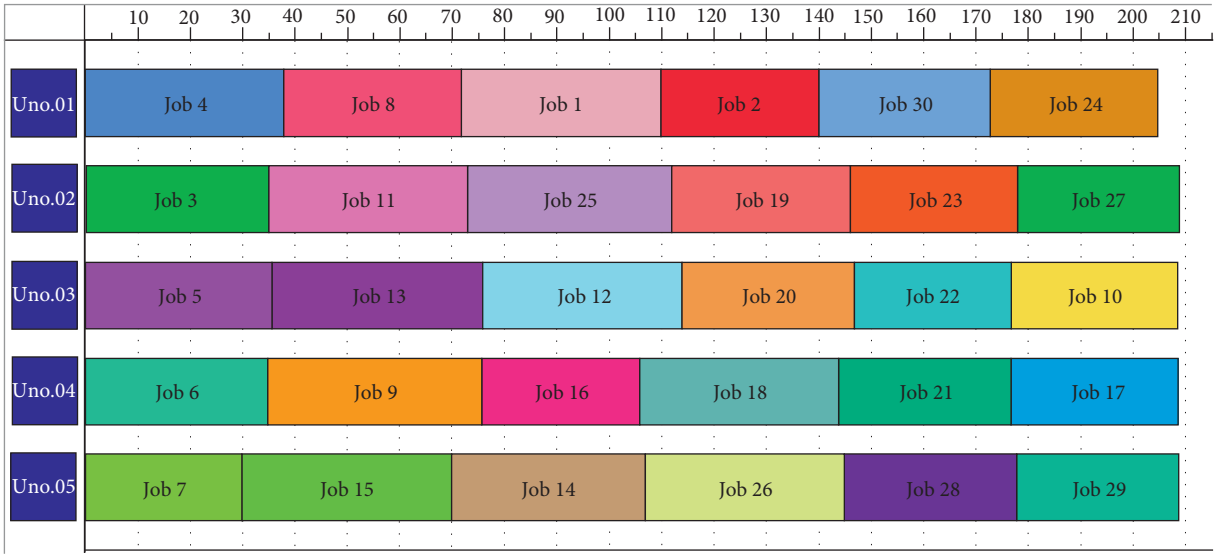

Figure 6: Gantt chart for job scheduling using EA and TS.

makespan obtained using EA and TS is 15.6 minutes which is best obtained from GA.

To show the effect of every algorithm on the makespan, Figure 4 compares between the makespan solutions obtained from LPT and GA algorithms given by Balin [5] and the developed EA and TS algorithm solutions.

To enhance the effect of the developed algorithms on minimizing the makespan, another big example from the study of Lei and Liu [40] is studied. The example consists of thirty jobs which are scheduled in five nonidentical parallel machines as shown in Table 5. The developed algorithms have given the best makespan over the artificial bee colony (ABC) algorithm which was used by Lei and Liu [40].

Gantt charts for job scheduling the problem studied by Lei and Liu [40] using ABC and (EA or TS) are shown in Figures 5 and 6, respectively. It is observed that the makespan obtained using EA and TS is 209 minutes which is best than that obtained from ABC, which is 218 minutes.

To assess the quality of EA and TS algorithm solutions on minimizing the makespan, Figure 7 compares between the ABC algorithm solution proposed by Lei and Liu [40] and the developed algorithm solutions. It is observed that the

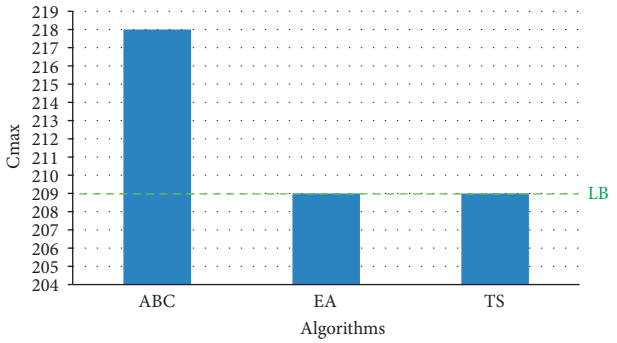

Figure 7: Histogram of averages of makespan means for $n = 30$ and $m = 5$.

solution obtained from the developed algorithms is better than that obtained from the ABC algorithm. Additionally, it always equals the lower bound.

Based on the value of the lower bound, the TS method was devised. There are no preset endpoints to the solution iterations, unlike the prior TS techniques. The EA is well known to be unable to address huge scheduling issues. In contrast to the proposed method, it can swiftly address large scheduling issues.

6. Conclusion

Many scheduling problems are NP-hard. Consequently, it is impossible to solve these problems efficiently via polynomial-time algorithms. Mathematical optimization techniques are suitable only for small-scale problems; therefore, they have limited applications in the case of a large-scale problem. Thus, the study efforts have focused on heuristic approaches. Among these approaches, TS outperforms others because of its high ability to give a solution that is very close to the optimum solution and its ease of implementation.

This paper proposed a developed TS algorithm for scheduling a set of jobs on nonidentical parallel machines with the intention of studying this algorithm's effect on minimizing the makespan. EA solution was proposed to enhance and evaluate the solution obtained from TS. Moreover, lower bound was developed to assess the quality of the proposed algorithm solution. The proposed algorithm solution is compared with the solutions obtained from other algorithms from the literature. Computational results showed that the solution obtained from the developed TS algorithm outperforms all the algorithms in the literature. The essential difference between TS and other heuristics is that TS is one of the most efficient heuristic techniques in the sense that it finds quality solutions in relatively short running time. Moreover, the developed TS algorithm always gives a solution that is equal or close to the lower bound.

Notations

j: Index of a job
n: Total number of jobs
i: Index of a machine
m: Total number of machines
$P(i, j)$: Processing time of job j on machine i
$C_{\max}$: Total completion time (makespan).

Data Availability

The raw data are included within the manuscript in Tables 1 and 5, so researchers can verify the results of the manuscript, replicate the analysis, and conduct secondary analyses.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors are grateful to the Raytheon Chair for Systems Engineering for funding this study.

References

[1] A. Cardon, T. Galinho, and J.-P. J. R. Vacher, "Genetic algorithms using multi-objectives in a multi-agent system," *Robotics and Autonomous Systems*, vol. 33, no. 2-3, pp. 179–190, 2000.

[2] C. Moon, M. Lee, Y. Seo, and Y. H. Lee, "Integrated machine tool selection and operation sequencing with capacity and precedence constraints using genetic algorithm," *Computers & Industrial Engineering*, vol. 43, no. 3, pp. 605–621, 2002.

[3] I. Muter, "Exact algorithms to minimize makespan on single and parallel batch processing machines," *European Journal of Operational Research*, vol. 285, 2020.

[4] A. H. Kashan, B. Karimi, and M. Jenabi, "A hybrid genetic heuristic for scheduling parallel batch processing machines with arbitrary job sizes," *Computers & Operations Research*, vol. 35, no. 4, pp. 1084–1098, 2008.

[5] S. Balin, "Non-identical parallel machine scheduling using genetic algorithm," *Expert Systems with Applications*, vol. 38, no. 6, pp. 6814–6821, 2011.

[6] J. K. Lenstra and A. H. G. Rinnooy Kan, "Complexity of scheduling under precedence constraints," *Operations Research*, vol. 26, no. 1, pp. 22–35, 1978.

[7] R. Sethi, "On the complexity of mean flow time scheduling," *Mathematics of Operations Research*, vol. 2, no. 4, pp. 320–330, 1977.

[8] L. Min and W. Cheng, "A genetic algorithm for minimizing the makespan in the case of scheduling identical parallel machines," *Artificial Intelligence in Engineering*, vol. 13, no. 4, pp. 399–403, 1999.

[9] L. Fanjul-Peyro and R. Ruiz, "Iterated greedy local search methods for unrelated parallel machine scheduling," *European Journal of Operational Research*, vol. 207, no. 1, pp. 55–69, 2010.

[10] E. Vallada and R. Ruiz, "A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times," *European Journal of Operational Research*, vol. 211, no. 3, pp. 612–622, 2011.

[11] L. Fanjul-Peyro and R. Ruiz, "Scheduling unrelated parallel machines with optional machines and jobs selection," *Computers & Operations Research*, vol. 39, no. 7, pp. 1745–1753, 2012.

[12] S. Balin, "Non-identical parallel machine scheduling with fuzzy processing times using genetic algorithm and

simulation," *The International Journal of Advanced Manufacturing Technology*, vol. 61, no. 9-12, pp. 1115–1127, 2012.

[13] C.-M. Joo and B.-S. Kim, "Non-identical parallel machine scheduling with sequence and machine dependent setup times using meta-heuristic algorithms," *Industrial Engineering and Management Systems*, vol. 11, no. 1, pp. 114–122, 2012.

[14] 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," *Journal of Intelligent Manufacturing*, vol. 23, no. 5, pp. 1949–1958, 2012.

[15] C. M. Joo and B. S. Kim, "Hybrid genetic algorithms with dispatching rules for unrelated parallel machine scheduling with setup time and production availability," *Computers & Industrial Engineering*, vol. 85, pp. 102–109, 2015.

[16] F. Geyik and K. Elibal, "A linguistic approach to non-identical parallel processor scheduling with fuzzy processing times," *Applied Soft Computing*, vol. 55, pp. 63–71, 2017.

[17] J. E. C. Arroyo and J. Y.-T. Leung, "Scheduling unrelated parallel batch processing machines with non-identical job sizes and unequal ready times," *Computers & Operations Research*, vol. 78, pp. 117–128, 2017.

[18] M. Tan, H.-L. Yang, and Y.-X. Su, "Genetic algorithms with greedy strategy for green batch scheduling on non-identical parallel machines," *Memetic Computing*, vol. 11, no. 4, pp. 439–452, 2019.

[19] W.-L. Liu, Y.-J. Gong, W.-N. Chen, Z. Liu, H. Wang, and J. Zhang, "Coordinated charging scheduling of electric vehicles: a mixed-variable differential evolution approach," *IEEE Transactions on Intelligent Transportation Systems,*vol. 21, no. 12, pp. 5094–5109, 2019.

[20] S. Zhou, L. Xing, X. Zheng, N. Du, L. Wang, and Q. Zhang, "A self-adaptive differential evolution algorithm for scheduling a single batch-processing machine with arbitrary job sizes and release times," *IEEE Transactions on Cybernetics*, vol. 51, 2019.

[21] F. Zhao, R. Ma, and L. Wang, "A self-learning discrete jaya algorithm for multiobjective energy-efficient distributed No-idle flow-shop scheduling problem in heterogeneous factory system," *IEEE Transactions on Cybernetics*, pp. 1–12, 2021.

[22] F. Zhao, X. He, and L. Wang, "A two-stage cooperative evolutionary algorithm with problem-specific knowledge for energy-efficient scheduling of No-wait flow-shop problem," *IEEE Transactions on Cybernetics*, vol. 51, 2020.

[23] F. Glover, "Tabu search-Part I," *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.

[24] S. Chu and H. Fang, "Genetic algorithms vs. tabu search in timetable scheduling," in *Proceedings of the 1999 Third International Conference on Knowledge-Based Intelligent Information Engineering Systems. Proceedings (Cat. No. 99TH8410)*, pp. 492–495, IEEE, Adelaide, South Australia, August 1999.

[25] F. Glover, J. P. Kelly, and M. Laguna, "Genetic algorithms and tabu search: hybrids for optimization," *Computers & Operations Research*, vol. 22, no. 1, pp. 111–134, 1995.

[26] A. Rathore, A. Bohara, R. G. Prashil, T. Prashanth, and P. R. Srivastava, "Application of genetic algorithm and tabu search in software testing," in *Proceedings of the Fourth Annual ACM Bangalore Conference*, p. 23, March 2011.

[27] E.-G. Talbi, *Metaheuristics: From Design to Implementation*, John Wiley & Sons, Hoboken, NJ, USA, 2009.

[28] F. Della Croce, V. Narayan, and R. Tadei, "The two-machine total completion time flow shop problem," *European Journal of Operational Research*, vol. 90, no. 2, pp. 227–237, 1996.

[29] G. Reinelt, *The Traveling Salesman: Computational Solutions for TSP Applications*, Springer-Verlag, Berlin, Germany, 1994.

[30] F. Glover and M. Laguna, "Tabu search," in *Handbook of Combinatorial Optimization*, pp. 2093–2229, Springer, Berlin, Germany, 1998.

[31] F. Glover, "Tabu search-Part II," *ORSA Journal on Computing*, vol. 2, no. 1, pp. 4–32, 1990.

[32] F. Glover, "Tabu search: a tutorial," *Interfaces*, vol. 20, no. 4, pp. 74–94, 1990.

[33] F. Glover, E. Taillard, and E. Taillard, "A user's guide to tabu search," *Annals of Operations Research*, vol. 41, no. 1, pp. 1–28, 1993.

[34] D. de Werra and A. Hertz, "Tabu search techniques," *Operations-Research-Spektrum*, vol. 11, no. 3, pp. 131–141, 1989.

[35] M. Gendreau and J.-Y. Potvin, "Tabu Search," in *Search Methodologies*, pp. 165–186, Springer, Berlin, Germany, 2005.

[36] M. Alshibli, A. El Sayed, E. Kongar, T. M. Sobh, and S. M. Gupta, "Disassembly sequencing using tabu search," *Journal of Intelligent and Robotic Systems*, vol. 82, no. 1, pp. 69–79, 2016.

[37] M. Abdel-Basset, G. Manogaran, D. El-Shahat, and S. Mirjalili, "Integrating the whale algorithm with tabu search for quadratic assignment problem: a new approach for locating hospital departments," *Applied Soft Computing*, vol. 73, pp. 530–546, 2018.

[38] M. Liu and C. Wu, "Scheduling algorithm based on evolutionary computing in identical parallel machine production line," *Robotics and Computer-Integrated Manufacturing*, vol. 19, no. 5, pp. 401–407, 2003.

[39] Y. Guo, A. Lim, B. Rodrigues, and L. Yang, "Minimizing the makespan for unrelated parallel machines," *International Journal on Artificial Intelligence Tools*, vol. 16, no. 3, pp. 399–415, 2007.

[40] D. Lei and M. Liu, "An artificial bee colony with division for distributed unrelated parallel machine scheduling with preventive maintenance," *Computers & Industrial Engineering*, vol. 141, Article ID 106320, 2020.