Hindawi
Mathematical Problems in Engineering
Volume 2023, Article ID 8353962, 11 pages
https://doi.org/10.1155/2023/8353962

Research Article

A Solution-Based Tabu Search Algorithm for the Resource-Constrained Project Scheduling Problem with Step Deterioration

Huafeng Dai,[1,2] **Lei Li**,[3] **Run Mao**,[3,4] **Xingyue Liu**,[4,5] **and Kun Zhou**[5]

[1]*School of Mechanical Engineering, Chengdu University, Chengdu 610105, China*
[2]*Technology and Equipment of Rail Transit Operation and Maintenance Key Laboratory of Sichuan Province, Chengdu 610031, China*
[3]*CNPC National Engineering Research Center for Oil & Gas Drilling Equipment Co., Ltd., Chengdu 610052, China*
[4]*School of Information Science and Engineering, Chengdu University, Chengdu 610105, China*
[5]*Key Laboratory of Intelligent Manufacturing Quality Big Data Tracing and Analysis of Zhejiang Province, China Jiliang University, Hangzhou 310018, China*

Correspondence should be addressed to Run Mao; maorun@cdu.edu.cn

Received 8 March 2022; Revised 26 October 2022; Accepted 12 April 2023; Published 10 May 2023

Academic Editor: Antonio M. Lopes

This paper integrates the classical resource-constrained project scheduling problem with step deterioration and accordingly presents a solution-based tabu search (SB-TS) algorithm to solve it with challenging computations. First, we choose successor list size as the priority rule to generate the initial solution of the presented methods. Next, the corresponding solution decoding and calculation ways are defined to update the related configuration at each iteration. Compared with the state-of-the-art methods, in terms of the known datasets, the proposed SB-TS finds the optimal solution for each instance in J30 and shows strongly competitive performance in J60, J90, and J120. Then, on newly generated datasets, where the step deterioration is attached to, the comparison experimental data state that the SB-TS with high-quality solution is superior to the move-based tabu search (MB-TS) algorithm. In addition, two key components are investigated to emphasize their attributions to the proposed algorithm.

1. Introduction

In the field of scheduling, resource-constrained project scheduling problem (RCPSP) is undoubtedly one of the focal points and received growing attention for several decades. The classical RCPSP, which is proved to be NP-hard [1], takes account of precedence constraints as well as limited resource. The RCPSP can be described as follows: n tasks and r renewable resources are given. The amount of resource $R_k, k = 1, \ldots, r$ is a constant at any time. To finish the task $i = 1, \ldots, n$, a fixed amount of r_{ik} units of resource k will be occupied and a constant period of time d_i should be consumed. In addition, precedence relationships are regulated between tasks. Finally, the goal is to decide the starting time τ_i of task $i = 1, \ldots, n$ in such a way that the maximum completion time $F_{\max} = \max F_i, i = 1, \ldots, n$, where $F_i = \tau_i + d_i$ represents the finishing time of task i, is minimized. At the same time, the resource demand cannot exceed its availability for any resource and the task precedence relationships must be fulfilled. Meanwhile, the task duration of the RCPSP is assumed to be fixed.

To this day, it seems that the best exact methods can only solve instances including at most 60 tasks where the tasks are not highly resource-constrained. But in realistic industry, the projects usually exceed this size and solutions must be quickly decided. For this purpose, most of the researchers have focused on heuristics for providing the best configuration in consideration of accuracy and computation speed.

Besides, differing from the fixed duration time of traditional RCPSP, there are various scheduling models where the processing times of tasks are time-dependent [2]. Scheduling with deteriorating effects is introduced by Browne and Yechiali [3], where the task duration is a linear non-decreasing function of its starting time. Over past

several decades, scheduling with deterioration emerges in all walks of life, including steel manufacturing, medical, or construction industry, where any task with delay will involve the imposition of an additional time on its duration. For example, the worker usually is allowed to have a break at some specific time periods in many organizations. In this case, if a task has a later start, the extra waiting time should be attached to its processing time accordingly. In this situation, the task duration time is a step function of its starting time, which we called step deterioration.

Regretfully, to the best of our knowledge, no other literature was reported till date where focus was on the RCPSP with deterioration effects. However, the RCPSP-SD can be seen everywhere. Given a practical example, a furniture industry with multi-skilled workers can accept different orders of abundant customers and decide the production schedule with the existing resources, such as carpenters. Unfortunately, the carpenter may lower his speed gradually due to fatigue. Hence, the later an order is processed, the longer the time it needs to finish. The deteriorating effect exists in the RCPSP universally. When tasks are processed before the fatigue threshold of workers, a basic processing time is consumed. Once tasks are delayed beyond the given threshold, a longer duration is needed to compensate employee's exhaustion.

In this paper, we study the RCPSP with step deterioration (RCPSP-SD). What differentiates the RCPSP-SD studied in this paper from the RCPSP is the duration time. The former is a step function of starting time and the latter is a preset constant. Due to the NP-hard nature of RCPSP, the mathematical model of RCPSP-SD taking account of the duration deterioration is intractable undoubtedly to solve its optimal configuration. Hence, we propose a novel solution-based tabu search algorithm. The remaining of this paper is structured as follows. Firstly, we review the related research around the topic in the next section. Then, compact notations and formulation of RCPSP-SD are introduced in Section 3. In Section 4, the solution-based tabu search (SB-TS) algorithm is described. Next, the assessment of proposed algorithm and computational results are given in Section 5. After that, we investigate the role of several ingredients of SB-TS. In the last section, conclusions and research perspectives are provided.

2. Literature Review

This section reviews the meaningful research and methods about the RCPSP as well as the scheduling problem with deterioration, respectively. In the first subsection (Section 2.1), we present the related research about the classical RCPSP, which set the task duration time as a constant. In the second subsection (Section 2.2), a number of representative studies about deterioration effect are listed.

2.1. The Related Literature about Classical RCPSP. The research about the variants and extension of the classical RCPSP, accompanying with its NP-hard nature in strong sense [1], was summarized in [4, 5]. In [6], the authors proposed *Start/End* and *On/Off* MILP formulations based on the concept of *event* for the classical RCPSP and compared their formulations with three other MILP formulations issued from the literature. The results tell that no MILP formulation dominates the others in terms of exact solving. The authors of [7] updated their results of the RCPSP benchmark instances with a new version of the branch-and-bound (B&B) procedure, which is demonstrated to be competitive and effective in [8]. Besides the classical RCPSP, to solve the chance-constrained RCPSP and RCPSP with non-renewable resources, B&B algorithms and MILP formulations also were introduced [9, 10].

Apart from above exact methodologies, metaheuristics still play a significantly important role for solving large-size RCPSP, and its summary can be seen in [11]. Zamani [12] introduced a method which includes a part of implicit enumeration as well as a genetic algorithm, conducted experiments on 2040 benchmark instances, and achieved a promising result. He et al. [13] proposed a heuristic algorithm on the foundation of the filter-and-fan (F&F) procedure, a search methodology including neighborhoods more than one and a procedure to switch neighborhoods, to shed light on the classical RCPSP. Bukata et al. [14, 15] presented a parallel tabu search, where each tabu search algorithm carries out together and communicates with others simultaneously, and acquired properly well results in terms of both solving velocities and solution qualities.

All preceding methodologies were proposed to solve the RCPSP which set the duration of all tasks as a constant. However, in a large amount of project scheduling cases, this premise will not be satisfied. Next, we will have an introduction of the research about scheduling with deterioration.

2.2. The Research of Scheduling with Deterioration. Various actual situations will not consider the duration of scheduled tasks as a fixed parameter but a variable depending on their beginning time [16, 17], which is called deterioration effect. Empirical research based on industries demonstrates that deterioration effect plays an important role in scheduling systems.

Liu et al. [18] considered a single-machine scheduling problem with deteriorating jobs and convex resource allocation and concluded that the problem is polynomially solvable in $O(nlogn)$ time, where n is the number of jobs. Wang and Liang [19] presented a heuristic algorithm and a branch-and-bound algorithm for a single-machine group scheduling problem with deteriorating jobs and resource allocation. Gawiejnowicz and Kolinska [20] considered open shop scheduling problems with fixed or proportionally deteriorating operating processing times. Wu et al. [21] focused on the two-stage three-machine problem with linear deterioration. To seek for optimal schedules, they proposed simulated annealing, differential evolution, and cloud theory-based simulated annealing algorithm. Wang et al. considered a single-machine scheduling problem with simple linear deterioration where the duration of a task is a simple linear function of its beginning time and proved

that some specified problems can be solved in polynomial time [22, 23]. Yang and Lu proposed approximation algorithms for the general position-dependent scheduling problems in [24]. Step deterioration effect was firstly discussed in [25] where the tasks have duration penalties for beginning after the due dates, and the authors provided a switching algorithm. Next, Mosheiov [26] extended this problem to settings of multi-machine and multi-step deterioration and introduced several heuristics. Miao and Zhang [27] solved the related scheduling problem considering parallel-machine scheduling and step deterioration simultaneously. Here, they presented a fully polynomial time approximation scheme for the case where the number of machines is a constant and jobs have anti-agreeable parameters. Meanwhile, they also demonstrated that the single-machine scheduling is NP-hard in the strong sense if jobs have distinct release dates and distinct deteriorating dates.

Moreover, abundant solution methodologies are designed to solve the intractable scheduling problems with different deterioration effects, such as variable neighborhood search (VNS) [28, 29], B&B algorithm [30, 31], memetic algorithm [32, 33], tabu search algorithm [34, 35], differential evolution algorithm [36, 37], learning algorithm [38–40], and so forth.

From the point of view of above review of scheduling with deterioration, the literature focus on RCPSP is zero.

3. Notations and Formulation of RCPSP-SD

To formulate considered RCPSP-SD as clearly as possible, the corresponding notations and descriptions are illuminated as follows. Given a set $N = \{1, 2, \ldots, n\}$, it includes all specific tasks of whole project. Besides, task 0 is regarded as a dummy begin task while task $n + 1$ is a dummy finish task, and their duration is zero. The duration d_i of each task $i \in N$ is decided collaboratively by its starting time τ_i, deteriorating threshold h_i, and its basic processing time b_i, as shown in equation (1). If the beginning time is earlier than its due date: $\tau_i \le h_i$, the duration of task i only requires a basic processing time b_i. Otherwise, as a punishment, extra processing time e_i is attached to its duration. To fulfill the whole project, a set R of renewable resource types should be provided and the capacity C_k of resource $k \in R$ limits the sum of resource k which can be used in the project. Each task i will pin r_{ik} units of resource type k on its progress. Besides, we defined a set E to incorporate all task pairs $(i, j), i, j \in N$ where precedence relationship exists between task i and j. In other words, the pair $(i, j) \in E$ implies that task j cannot start until task i is completed.

Next, a 0-1 mixed integer programming model of the RCPSP-SD with the goal of minimizing makespan will be formulated. Before that, the binary variables x_{ij}, which receive value 1 if task i is prior to task j and zero otherwise, and the variables n_{ijk}, which represent the number of resource k delivered from task i to task j, are defined.

$$\text{Minimize } \tau_{n+1}, \tag{1}$$

subject to

$$d_i = \begin{cases} b_i, \tau_i \le h_i \\ b_i + e_i, \tau_i > h_i \end{cases} \quad \forall i \in N, \tag{2}$$

$$x_{ij} = 1, \quad \forall (i, j) \in E, \tag{3}$$

$$x_{ij} + x_{ji} \le 1 \forall i, j \in N, i \ne j, \tag{4}$$

$$n_{0ik} \le r_{ik} x_{0i} \forall i \in N, \forall k \in R, \tag{5}$$

$$n_{j(n+1)k} \le r_{jk} x_{j(n+1)} \forall j \in N, \forall k \in R, \tag{6}$$

$$n_{ijk} \le Z_{\min} x_{ij} \forall i, j \in N, i \ne j, \forall k \in R, \tag{7}$$

$$\sum_{i \in N} n_{0ik} = C_k \forall k \in R, \tag{8}$$

$$\sum_{i \in N} n_{i(n+1)k} = C_k \forall k \in R, \tag{9}$$

$$\sum_{i \in N} n_{ijk} = r_{jk} \forall i, j \in N, \forall k \in R, \tag{10}$$

$$\sum_{i \in N} n_{jik} = r_{jk} \forall i, j \in N, \forall k \in R, \tag{11}$$

$$\tau_i + d_i - M\left(1 - x_{ij}\right) \le \tau_j \forall i, j \in N, i \ne j, \tag{12}$$

$$\tau_i \ge 0, n_{ijk} \ge 0, x_{ij} \in \{0, 1\} \forall i, j \in N, \forall k \in R. \tag{13}$$

In the above model, formula (1) indicates the completion time of whole project. The set of equation (2) defines the duration of tasks. The set of equation (3) tells that the precedence constraints among tasks must be complied with. The set of inequality (4) is reflection of natural logic in the practical operations. The set of inequalities (5) and (6) denotes that all resources are passed from task 0 and gathered into task $n + 1$ finally. The set of constraint (7) is used to ensure that the resource flow between any pair of tasks (i, j) will not exceed $Z_{\min}$, where $Z_{\min} = \min\{r_{ik}, r_{jk}\}$. The set of equations (8) and (9) restricts that, in the project, the capacity of any resource is fixed. The set of equations (10) and (11) assumes that resources will be allocated as needed and any task $i \in N$ will not keep any resource when it was completed. The set of inequality (12) is the big-M formulation to guarantee no violation to the relationship between sequencing variable (x_{ij}) and the starting time variables (τ_i, τ_j). The set of constraint (13) refers to the regular domains of defined variables.

4. Solution-Based Tabu Search

Carlton and Barnes [41] presented a tabu search algorithm with a two-level tour hashing scheme to solve the traveling salesman problem with time windows, which outperformed the traditional attribute-based tabu search. Wang et al. [42] designed a solution-based tabu search algorithm and

Input: Problem instance I, the depth of tabu search α, three hash vectors H_1, H_2, H_3 and corresponding hash functions h_1, h_2, h_3, the length of tabu list L
Output: The best configuration s^b
(1) Create an initial solution s_0/* referring to Section 4.3
(2) Define the completion time function f/* referring to Section 4.2
(3) **begin**
(4) **for** $i = 0: L - 1$ **do**
(5) $H_1[i] \longleftarrow 0, H_2[i] \longleftarrow 0, H_3[i] \longleftarrow 0$;/* Initialization of three tabu lists/hash vectors
(6) **end**
(7) $s^{'} \longleftarrow s_0$/* $s^{'} \longleftarrow$ current solution for local search
(8) $s^b \longleftarrow s^{'}$
(9) Nonimp $\longleftarrow$ 0/* Nonimp counts the consecutive iterations where s^b is not improved
(10) **while** Nonimp $\leq \alpha$ **do**
(11) Execute neighborhood search procedure on current solution $s^{'}$ to seek for the candidate listl CL, the set of promising neighborhood solutions/* referring to Section 4.4*/
(12) Select a non-tabu configuration s^l with shortest makespan from CL/* the calculation of tabu status referring to Section 4.5*/
(13) $s^{'} \longleftarrow s^l$
(14) **if** $f(s\hat{b} > f(s^{'}))$ **then**
(15) $s^b \longleftarrow s^{'}$
(16) Nonimp $\longleftarrow$ 0
(17) **end**
(18) $H_1[h_1(s^{'})] \longleftarrow 1, H_2[h_2(s^{'})] \longleftarrow 1, H_3[h_3(s^{'})] \longleftarrow 1$/*update three tabu lists/hash vectors
(19) Nonimp $\longleftarrow$ Nonimp + 1
(20) **end**
(21) $t \longleftarrow$ time()/* capture the runtime of whole procedure
(22) **return** $\{s^b, t\}$
(23) **end**

Algorithm 1: Main sketch of the proposed SB-TS algorithm for the RCPSP-SD.

a memetic algorithm, which is similar to genetic algorithm, and the tabu search algorithm discovered better solutions for 179 (71.6%) out of 250 instances while the memetic algorithm found better solutions for 157 (62.8%) instances. In [42], the effectiveness of solution-based tabu search is demonstrated for solving certain kinds of binary optimization problems.

In this section, our solution-based tabu search (SB-TS) algorithm as well as its supporting procedures is detailed in the following subsections.

4.1. Method Framework. The proposed SB-TS algorithm will be elaborated by the following pseudocode as shown in Algorithm 1. In our SB-TS, we employ three hash vectors H_1, H_2, H_3 as tabu list with length L to record the tabu status of solution, and the tabu status of any solution is codeterimined by three hash functions h_1, h_2, h_3 [43].

Firstly, the SB-TS algorithm initializes a solution s_0 according to preset rules (line 1) as well as the hash vectors (lines 4-5) which are utilized to manage the tabu status of neighborhood solutions and define the evaluation function f (line 2) whose objective value is equal to the solution completion time. Next, initial solution s_0 is assigned to current solution $s^{'}$ and the best solution found so far is reported (lines 7-8). Then, the algorithm enters into a while loop which is made up the core part of the SB-TS (lines 10–20). The while loop iterates repeatedly until the predetermined stopping criteria are met. That is to say, the maximum number between two iterations without improvement is equal to the depth of tabu search α. On each run of the while loop, a candidate list CL composed of promising neighborhood solutions is constructed (Section 4.4) and a best non-tabu configuration s^l is selected to replace the current solution $s^{'}$. Obviously, the eligible s^l must respect $H_1(h_1(s^{'}))\&H_2(h_2(s^{'}))\&H_3(h_3(s^{'})) == 0$ (Section 4.5). Naturally, three tabu lists/hash vectors are updated accordingly (line 17). At the end of procedure, the runtime t of whole procedure and the best configuration s^b of problem instance will be returned when the SB-TS terminates.

4.2. Solution Expression and Calculation. With reference to the set of tasks N, the set of resource R, the strict correspondences between resources and tasks (T-C-R), the precedence relationships among tasks (T-PR-T), and the capacity limitation of each resource (CL-R), in this paper, we employ a vector $s = \{T_1, T_2, \ldots T_i, \ldots, T_n\}$ to present a configuration of the project, where $T_i, i \in N$ corresponds to task i. s, whose sequential order reflects the task priority to compete for resources, and the innate T-C-R, T-PR-T, and CL-R of the RCPSP-SD instances determine the starting time of each task collectively. The detailed computational procedure is shown in Algorithm 2. It is worthy to note that the task duration time d is a step function of starting time which has been defined in Section 3 during the calculation process. Finally, the procedure returns maximum task finishing time as the objective value of solution s.

Input: a solution s, the duration function of task d, T-C-R, T-PR-T and CL-R of problem instance I
Output: The completion time function value $f(s)$
(1) **begin**
(2) Currenttime $\longleftarrow$ 0; $AT \longleftarrow \varnothing$
(3) **for** $i = 0$: $L - 1$ **do**
(4) Decide whether task $s(i)$ can be handled at Currenttime based on the consideration of only T-PR-T, irrespective of the constraints of T-C-R and CL-R. If yes, add task $s(i)$ to AT, $AT \longleftarrow (AT, s(i))$
(5) **end**
(6) Waitlist $\longleftarrow N \backslash AT$
(7) **while** Waitlist $\neq \varnothing$ **do**
(8) Assign resources to tasks $\in AT$ in accordance with the order of s and T-C-R while guaranteeing the feasibility of CL-R at the same time
(9) Update the starting time ($\longleftarrow$ Currenttime) and finishing time ($\longleftarrow$ Currenttime + d) of corresponding tasks which have been assigned resources to
(10) Find the task with the earliest finishing time Nexttime which is later than Currenttime
(11) Release all resources occupied by tasks which are completed from Currenttime to Nexttime, and delete them from AT
(12) Currenttime $\longleftarrow$ Nexttime
(13) Add all tasks which can be handled at Currenttime to AT and delete them from Waitlist
(14) **end**
(15) **return** the latest finishing time of tasks as $f(s)$
(16) **end**

Algorithm 2: The computational procedure of the solution completion time.

4.3. Initialization of Solution. For the sake of the characteristics of benchmark instances [44], our SB-TS algorithm begins with an initial solution s_0 constructed in the descending order of successor list size (SLS). In addition, in this section, we establish a *task list* (TL) which determines the priority to compete for resources and introduce other four different *priority rules* to construct TL.

Each rule consists of its corresponding *sorting criteria* to sort tasks, and all *priority rules* share a common time breaker which imposes the task with smaller index to be situated first. Three additional defined *priority rules* are given as follows.

Priority Rule 1. Sort tasks in the ascending order of index.

Priority Rule 2. Sort tasks in the ascending order of deteriorating threshold (h).

Priority Rule 3. Sort tasks in the descending order of the extra penalty time (e).

An example for illuminating the instances and solution expression is given in Figure 1. To calculate the makespan of $s = \{3, 2, 1, 5, 4, 7, 6, 8\}$ while guaranteeing the legality of configuration, all internal restraint relationships of T-C-R, T-PR-T, and CL-R in Figure 1 cannot be violated. For example, at time 0, only tasks 1, 2, and 4 can be assigned resources to, complying with the order {2, 1, 4} implied by s.

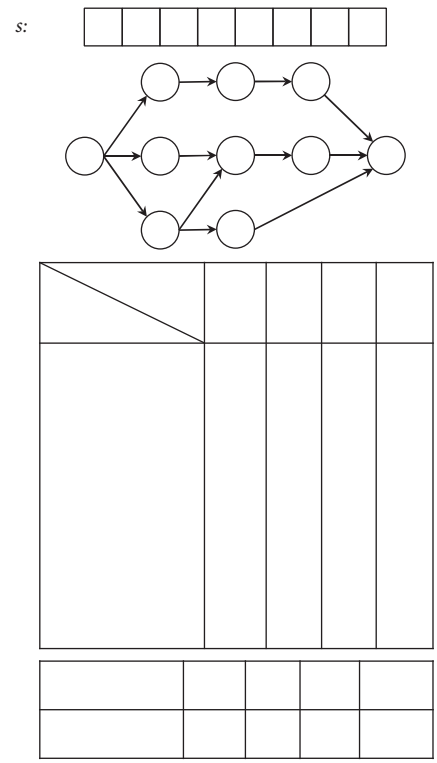

Figure 1: An example for illuminating the instances and the solution expression, where the size of N is 8 ($n = 8$) and the size of R is 4.

4.4. Local Search. In this paper, we design a local refinement procedure on the basis of tabu search which has a good performance in solving RCPSP. This section introduces the local search phase, a core part, of our SB-TS algorithm. Its function lies in finding high-quality local optima through exploring the search space. As for neighborhood used in SB-TS, it is defined by the most common swap operator $Swap()$ in combinatorial optimization [34]. Given a configuration $s = \{\ldots, i, \ldots, j \ldots\}$, the Swap$(i, j)$ operator exchanges the positions of task i and task j and delivers a neighborhood solution of s which is recorded as $s \otimes \text{Swap}(i, j)$.

Given an incumbent solution $s^{'}$, the possible neighborhood delivered by $Swap()$ consisted of all solutions which can be available by applying Swap() to $s^{'}$. Here, a candidate list CL is generated to store a set of high-quality neighbor solutions that are far less than the whole neighborhood (line 11 of Algorithm 1). After that, the SB-TS algorithm traverses the CL and selects the best non-tabu solution s^{l} to replace the incumbent solution $s^{'}$ (line 12 and 13 of Algorithm 1). As for the tabu status of any solution, all solutions which have not been visited are non-tabu and the visited solutions are labelled as tabu, in principle. The details about the tabu list management are expanded in Section 4.5. Next, above process with updated $s^{'}$ is repeated again and again till the best solution s^{b} has not improved for α consecutive iterations.

4.5. Tabu List Management. In our proposed SB-TS algorithm, tabu search is picked to explore the neighborhood. Tabu list management, which determines the tabu status of neighbors, is a key part for the tabu search and affects directly the search. In this paper, the solution-based tabu strategy which is based on three hash vectors H_1, H_2, H_3 as well as their associated hash functions h_1, h_2, h_3 is adopted. Each hash vector, which also can be called tabu list, is a L-dimensional vector and each element in H_k, $k = 1, 2, 3$, takes the value of zero or one. As for the hash functions $h_k, k = 1, 2, 3$, they are employed to map neighborhood solutions to the indices of H_k, $k = 1, 2, 3$, i.e., h_k: $s \longrightarrow \{0, 1, 2, \ldots, L-1\}$.

Following previous research [42, 45], we employ our hash functions $h_k, k = 1, 2, 3$ associated to $s = \{T_1, T_2, \ldots T_N\}$ as follows:

$$h_k(s) = \left(\sum_{i=1}^{n} \lfloor i^{\xi_k} \rfloor \times T_i \right) \mod L, \tag{14}$$

where $\xi_k, k = 1, 2, 3$ are factors used to define hash functions and L is set to 10^7 empirically.

In the SB-TS algorithm, the tabu status of a candidate configuration is computed by the following rule on the basis of above hash vectors/tabu lists and hash functions. To begin with, all elements in $H_k, k = 1, 2, 3$ are set to zero (lines 4–6 of Algorithm 1). Next, the values are updated accordingly during the search process, as depicted in line 18 of Algorithm 1.

Accompanying with above hash vectors/tabu lists and hash functions, given a current solution $s^{'}$, its tabu status can be easily calculated. $s^{'}$ is regarded as tabu if three values of hash vectors/tabu lists $H_1(h_1(s^{'}))$, $H_2(h_2(s^{'}))$, $H_3(h_3(s^{'}))$ all take value of 1. If not, $s^{'}$ has not been visited and its status is non-tabu. In a nutshell, solution $s^{'}$ is abandoned for further consideration, if and only if the equation $H_1(h_1(s^{'}))\&H_2(h_2(s^{'}))\&H_3(h_3(s^{'})) == 1$ is met.

5. Experiments and Comparisons

This section plans to evaluate the performance of the presented SB-TS algorithm through extensive contrast experiments with current leading methods. Due to the novelty of RCPSP-SD, we firstly examine the SB-TS on PSPLIB benchmark instances [44] which are considered a set of instances for classical RCPSP without deterioration effects. Then, the instances with 30 non-dummy tasks are modified to suit RCPSP-SD.

5.1. Parameter Settings and Experiment Protocol. Table 1 provides the settings of the parameters used in SB-TS algorithm. On the basis of preliminary experiments and discussion in Section 6.2, iteration depth α is set to 500, and the factors ξ_1, ξ_2, ξ_3 used in hash functions h_1, h_2, h_3 take the values of 1.8, 1.9, and 2.0, respectively.

Our SB-TS algorithm and other contrast methods were implemented in MATLAB R2017b. All experiments presented next were executed on a personal computer with an Intel Core i5 processor (1.4 GHz CPU and 2 GB RAM), running the macOS system.

5.2. Overall Results of PSPLIB Instances. The first experimental group evaluates the performance of the SB-TS algorithm on the PSPLIB instances, where the J30, J60, and J90 datasets all include 480 instances and J120 dataset has 600 problems. In our work, to assess the SB-TS fully and comprehensively, some recent methodologies in the literature are adopted, including the memetic algorithm (MA) [46], PSO-based hyper-heuristic (PSO-HH) algorithm [47], consolidated optimization algorithm (COA) [48], and the decomposition-based genetic algorithm (DBGA) [49].

The experimental results are reported in Tables 2 and 3 for the PSPLIB instances. Table 2 records the results of the SB-TS as well as some state-of-the-art algorithms, and Table 3 reports the number of instances which can obtain the optimal schedule by SB-TS. To evaluate the efficiency of SB-TS, the average relative percentage deviation from the best-known schedules (upper bound) Dev_u and the deviation from lower bound Dev_l delivered by critical path are adopted.

For PSPLIB instances, the deviation from upper bound and lower bound can be computed as

$$\begin{aligned} \mathrm{Dev}_u(\%) &= \frac{\sum_{p=1}^{\mathrm{num}} \left[f_{\mathrm{best},p} - f_{ub,p} / f_{ub,p} \times 100 \right]}{\mathrm{num}}, \\ \mathrm{Dev}_l(\%) &= \frac{\sum_{p=1}^{\mathrm{num}} \left[f_{\mathrm{best},p} - f_{lb,p} / f_{lb,p} \times 100 \right]}{\mathrm{num}}, \end{aligned} \tag{15}$$

where num denotes the number of the instances in J30, J60, J90, and J120 datasets and for any instance p, $f_{\mathrm{best},p}$ is the best one obtained by SB-TS over 20 independent replications, $f_{ub,p}$ is the known upper bound (UB), and $f_{lb,p}$ is the lower bound (LB) value. The UB and LB values are available in [50].

Table 2 compares SB-TS with the state-of-the-art methods in the literature in terms of *ARPD*. It tells that our algorithm not only finds all optimal schedules for J30 dataset but also has Dev_u and Dev_l both far smaller than other algorithms for J60, J90, and J120. Table 3 counts the total number of instances where its upper bound equals lower

Table 1: Settings of parameters in SB-TS algorithm.

Parameters	Description	Values
α	Iteration depth of SB-TS	500
ξ_1	Factor employed in the hash function h_1	1.8
ξ_2	Factor employed in the hash function h_2	1.9
ξ_3	Factor employed in the hash function h_3	2.0

Table 2: Experimental results and comparisons in terms of average relative percentage deviation (ARPD) on the PSPLIB instances.

Methods	J30		J60		J90		J120	
	Dev_u	Dev_l	Dev_u	Dev_l	Dev_u	Dev_l	Dev_u	Dev_l
SB-TS	0.00	0.00	3.03	3.69	3.22	4.30	7.79	11.10
MA [46]	0.00	0.00	8.67	10.55	7.31	9.94	23.84	31.12
DBGA [49]	0.01	0.01	8.79	10.71	7.62	9.90	22.43	31.81
COA [48]	0.00	0.00	8.70	10.58	—	—	—	31.22
POS-HH [47]	0.01	0.01	8.87	10.68	—	—	—	31.23

Table 3: The number of instances where SB-TS can achieve optimal schedule on J30, J60, J90, and J120 datasets.

Datasets	The total number of known instances	The number of instances where the schedule obtained
J30	480	480
J60	402	319
J90	375	226
J120	290	48

bound and shows the amount of instances where SB-TS achieves optimal.

Tables 2 and 3 demonstrate that the outcomes from our SB-TS are noteworthy compared to the excellent results in the literature. For J30 dataset, SB-TS finds the optimal solution for each instance and shows strongly competitive performance for solving J60, J90, and J120 datasets.

5.3. Computational Results of RCPSP-SD Instances. The previous experiments and comparisons based on PSPLIB benchmark instances in Section 5.2 demonstrate the advantages of SB-TS with respect to the relative issues of RCPSP. Next, the SB-TS is used to solve the RCPSP-SD.

Before beginning, the problem instances should be generated to evaluate the performance of SB-TS. Here, J30 and J60 datasets are picked. To fit with the characteristics of RCPSP-SD, datasets should be modified slightly. To meet with the step deterioration precisely, the detailed modifications are presented below. First, the deterioration threshold h_i for any task i is equal to b_i multiplied by a factor β, which randomly is generated from the interval (0, 10]. As for the other parameter e_i, the extra punishment is created by delay, and it is drawn from the uniform distribution (0, 20].

For the novelty of RCPSP-SD, comparative data and methods are not available in the literature. Here, we establish the traditional move-based tabu search (MB-TS) [34], with same swap neighborhood structure, as a contrast to carry out the tests on modified J30 and J60 datasets.

For the simplicity of detecting the performance of SB-TS and MB-TS, same as Section 5.2, an average relative percentage deviation (ARPD) is defined by

$$ARPD\,(\%) = \frac{\sum_{p=1}^{\text{num}} \left[z_p - z_{\text{best},p}/z_{\text{best},p} \times 100\right]}{\text{num}}, \tag{16}$$

where z_p represents the final result obtained by the current algorithm and $z_{\text{best},p}$ equals the best of two algorithms, for any problem instance p.

Table 4 sums up the results of the SB-TS as well as MB-TS for solving the modified J30 and J60 datasets. Columns 2 and 4 record the ARPD (%) of SB-TS and MB-TS, respectively. As for columns 3 and 5, $\text{Time}_{\text{avg}}(s)$ records how many seconds it takes on average while the corresponding procedure solves the problem instances of modified J30 and J60 separately. Besides, at the end of Table 4, we count the number of instances where SB-TS performs better than MB-TS (#better), the amount where the solutions derived from two methods have same objective value (#equal), and the quantity where SB-TS performs worse (#worse). At the end of Table 4, the amount of problem instances in J30 and J60 is given (#total). As depicted in Table 4, the *ARPD*s (%) derived form SB-TS equal 0.23% and 0.48% while the values of MB-TS are 0.56% and 0.53% for J30 and J60 datasets. In addition, SB-TS obtains better solutions for 51 out of 480 instances on J30 dataset and 73 out of 480 instances on J60 (51(J30) + 73(J60)). As for MB-TS, it performs better in 28 instances of J30 and 63 instances of J60 (28(J30) + 63(J60)). In addition, for J30 and J60 datasets, two methods achieve same objective values for 401 and 344 instances (401(J30) + 344(J60)), respectively. As for their runtime, MB-TS takes advantage over SB-TS especially for J60 dataset.

The experimental data in Table 4 state that SB-TS has advantages over classical MB-TS to solve the RCPSP-SD in terms of the solution quality. For J30 dataset, MB-TS has a slight advantage with respect to the runtime while for J60 dataset, it shows an obvious gap.

6. Analysis and Discussion

In this section, we study and analyze two essential ingredients of the SB-TS to shed light on its performance. One is the selected priority rule to establish the initial solution and the other is the effectiveness of hash vectors as well as hash functions.

6.1. Influence of Different Priority Rules. The presented SB-TS constructs s_0 in an ascending order of index. Besides, as shown in Section 4.3, three priority rules, *priority rule* 1, *priority rule* 2, and *priority rule* 3, are defined. Accordingly, the corresponding algorithms are nominated SB-TS_1, SB-TS_2, and SB-TS_3, respectively. Then, two groups of experiments are done. One is carried out on the J30 and J60

Table 4: Computational results of SB-TS and MB-TS on modified J30 and J60 datasets.

Datasets	SB-TS		MB-TS	
	ARPD (%)	$\text{Time}_{\text{avg}}(s)$	ARPD (%)	$\text{Time}_{\text{avg}}(s)$
Modified J30	0.23	10.03	0.56	9.57
Modified J60	0.48	122.39	0.53	93.56
#Better	51(J30) + 73(J60)	0	28(J30) + 63(J60)	960
#Equal	401(J30) + 344(J60)	0	401(J30) + 344(J60)	0
#Worse	28(J30) + 63(J60)	960	51(J30) + 73(J60)	0
#Total	480(J30) + 480(J60)	960	480(J30) + 480(J60)	960

Table 5: Computational results of SB-TS and SB-T S_1 to solve J30 and J60 datasets of RCPSP.

Methods	SB-TS		SB-T S_1	
	Dev_u	Dev_l	Dev_u	Dev_l
J30	0	0	3.03	3.69
J60	0.16	0.16	3.19	3.86
#Better	0(J30) + 120(J60)	0(J30) + 120(J60)	119(J30) + 63(J60)	119(J30) + 63(J60)
#Worse	119(J30) + 63(J60)	119(J30) + 63(J60)	0(J30) + 120(J60)	0(J30) + 120(J60)
#Equal	361(J30) + 297(J60)	361(J30) + 297(J60)	361(J30) + 297(J60)	361(J30) + 297(J60)
#Total	480(J30) + 480(J60)	480(J30) + 480(J60)	480(J30) + 480(J60)	480(J30) + 480(J60)

datasets of RCPSP. The other is based on the modified J30 and J60 of RCPSP-SD.

Tables 5 and 6 record the computational results of four algorithms based on different priority rules. Except for *priority rule* 1, *priority rule* 2 and *priority rule* 3 both sort tasks according to some characteristics defined by deterioration effect. As a result, in terms of the J30 and J60 datasets without deterioration, SB-T S_2 and SB-T S_3 are not applicable and accordingly computational results will not be recorded.

The results of Table 5 mainly compare *priority rule* 1 with the current rule of SB-T S. The data show that our SB-T S has a slight edge over SB-T S_1 to solve J30 as well as J60 datasets. Besides, in terms of the results of J60 dataset, two methods outperform the state-of-the-art algorithms in Table 2, which to some extent demonstrates the advantages of our SB-TS framework.

Table 6 records the *ARPD* (defined in Section 5.3) of four methods to solve the modified J30 and modified J60 datasets. In addition, the row #best of Table 6 indicates the number of instances where priority rule obtained the best results, including the instances where two or more algorithms achieve same optimal results. The experimental data of Table 6 demonstrate that sorting the tasks in descending order of successor list size to construct the initial solution is efficient and plays a significant role in our SB-TS algorithm.

6.2. Effectiveness of Hash Vectors and Hash Functions. As depicted in Section 5.1, the tabu list management in this paper involves two important parts, the length of hash vector L and the factor $\xi_k, k = 1, 2, 3$ employed in hash functions $h_k, k = 1, 2, 3$. To investigate the influence of L and $\xi_k, k = 1, 2, 3$ on the performance of SB-TS, two extra groups of experiments are carried out on J30 and J60 datasets of RCPSP.

The length of hash vector L is set as $L = 1 \times 10^7$ in SB-T S. In the first experimental group, to show the effectiveness of L, we establish three variants SB-T S_4, SB-T S_5, and SB-T S_6, generated by disabling H_1, H_2, and H_3 of SB-T S, respectively, while other parts of SB-TS are kept unchanged. Besides, two additional variants SB-TS_7 and SB-T S_8 are created by setting the length of hash vectors as $L = 1 \times 10^7$ and $L = 1 \times 10^8$, respectively. The final results of SB-T S_4, SB-T S_5, SB-T S_6, SB-T S_7, and SB-T S_8 to solve modified J30 and J60 datasets of RCPSP-SD are reported in Table 7.

Columns 2–4 of Table 7 tell that in terms of the *ARPD*(%), SB-TS, in the current experimental environment, performs very similar to two or three hash vectors when solving the modified J30 and J60 datasets. Similarly, through the investigation of results of SB-TS, SB-T S_7, and SB-T S_8, it is not hard to conclude that SB-T S is not sensitive to the value of L. However, theoretically, the more the hash vectors are provided, the less the possible collisions happen. At the same time, it has a significant reduction in run time. On the basis of that, our SB-T S algorithm employs three hash vectors and sets the length of hash vectors as $L = 1 \times 10^7$.

In the second experimental group, we focus on the factor $\xi_k, k = 1, 2, 3$ involved in hash functions $h_k, k = 1, 2, 3$. To expound the impact of ξ_k on the performance of the SB-TS, 6 parameter combinations (ξ_1, ξ_2, ξ_3) are selected, including (1.9, 2.0, 2.1), (1.8, 1.9, 2.0), (1.5, 1.6, 1.7), (1.4, 1.5, 1.6), (1.3, 1.4, 1.5), and (1.1, 1.2, 1.3). Table 8 records the related computational results where we compute the *ARPD*(%) for each parameter combination and each dataset, respectively. Besides, the row "#best" represents the number of instances where the specific parameter combination derives a schedule better than all other combinations in terms of the objective value and "#equal" means the quantity of instances where 6 combinations have same performances. A noteworthy thing

Table 6: Computational results of SB-TS, SB-T S_1, SB-T S_2, and SB-T S_3 to solve modified J30 and J60 datasets of RCPSP-SD.

Datasets	SB-TS *ARPD* (%)	SB-T S_1 *ARPD* (%)	SB-T S_2 *ARPD* (%)	SB-T S_3 *ARPD* (%)
Modified J30	0	0.23	0.46	0.20
Modified J60	0.79	0.99	1.76	1.53
#Best	480(J30) + 366(J60)	449(J30) + 298(J60)	458(J30) + 313(J60)	464(J30) + 342(J60)

Table 7: Computational results of SB-TS, SB-T S_4, SB-T S_5, SB-T S_6, SB-T S_7, and SB-T S_8 to solve modified J30 and J60 datasets of RCPSP-SD.

Datasets	Two hash vectors ($L = 10^6$)			Three hash vectors		
	SB-T S_4 (H_2, H_3)	SB-T S_5 (H_1, H_3)	SB-T S_6 (H_1, H_2)	SB-TS ($L = 10^6$)	SB-T S_7 ($L = 10^7$)	SB-T S_8 ($L = 10^8$)
Modified J30	0.00	0.00	0.00	0.00	0.00	0.00
Modified J60	0.00	0.00	0.00	0.00	0.00	0.00

Table 8: Computational results of SB-TS with different values of ξ_1, ξ_2, ξ_3 to solve modified J30 and J60 datasets of RCPSP-SD.

Datasets	*ARPD* (%)					
	(1.9, 2.0, 2.1)	(1.8, 1.9, 2.0)	(1.5, 1.6, 1.7)	(1.4, 1.5, 1.6)	(1.3, 1.4, 1.5)	(1.1, 1.2, 1.3)
Modified J30	0.11	0	0.02	0.18	0.22	0.28
Modified J60	0.09	0	0.11	0.15	0.29	0.16
#Best	63(J30) + 59(J60)	77(J30) + 108(J60)	71(J30) + 56(J60)	49(J30) + 51(J60)	49(J30) + 43(J60)	41(J30) + 53(J60)
#Equal	403(J30) + 372(J60)					
#Total	480(J30) + 480(J60)					

is that all these three statistics are based on the modified J30 and J60 datasets.

Different from the former experiment, the data of Table 8 demonstrate that the performance of SB-TS is sensitive to the value of ξ_1, ξ_2, ξ_3. Based on the experimental results presented in Table 8, we set the default parameter values (ξ_1, ξ_2, ξ_3) of our SB-TS to (1.8, 1.9, 2.0), for the reason of zero *ARPD*(%) for both modified J30 and J60.

7. Conclusions

This paper presents a new decoding way of solution in the field of the resource-constrained project scheduling problem (RCPSP) and proposes a solution-based tabu search (SB-TS) algorithm for RCPSP with step deterioration (RCPSP-SD). We assessed the proposed SB-TS algorithm on PSPLIB instances without consideration of deterioration effects on the first stage. Then, on newly generated datasets fitted with the characteristics of RCPSP-SD, the SB-TS algorithm also shows its excellent performance in comparison with classical MB-TS.

Besides, two essential components are investigated to shed light on their effects on the behaviour of SB-TS algorithm. One is the selected priority rule, and the other includes hash vectors as well as hash functions. The first experimental group and analysis shed light on the contribution of the SLS priority rule to the SB-TS algorithm. In addition, the results show the efficiency of SB-TS framework to some extent.

In this paper, we investigate the resource-constrained project scheduling problem with step deterioration. As an extension of RCPSP, it enriches the research range of RCPSP fields through including the deterioration effect. In this new problem, it would be meaningful to study its inner properties and design more excellent algorithms to solve the actual problem.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was partially supported by Open Research Fund of Technology and Equipment of Rail Transit Operation and Maintenance Key Laboratory of Sichuan Province (2021YW003) and Key Laboratory of Intelligent Manufacturing Quality Big Data Tracing and Analysis of Zhejiang Province (grant no. ZNZZSZ-CJLU2022-06).

References

[1] J. Blazewicz, J. K. Lenstra, and A. Kan, "Scheduling subject to resource constraints: classification and complexity," *Discrete Applied Mathematics*, vol. 5, no. 1, pp. 11–24, 1983.

[2] M. Pinedo, *Scheduling: Theory, Algorithms and Systems, Physical Review Letters*, Springer, Berlin, Germany, 2012.

[3] S. Browne and U. Yechiali, "Scheduling deteriorating jobs on a single processor," *Operations Research*, vol. 38, no. 3, pp. 495–498, 1990.

[4] A. Laurent, L. Deroussi, N. Grangeon, and S. Norre, "A new extension of the rcpsp in a multi-site context: mathematical model and metaheuristics," *Computers and Industrial Engineering*, vol. 112, no. 1, pp. 634–644, 2017.

[5] S. Hartmann and D. Briskorn, "A survey of variants and extensions of the resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 207, no. 1, pp. 1–14, 2010.

[6] O. Koné, C. Artigues, P. Lopez, and M. Mongeau, "Event-based milp models for resource-constrained project scheduling problems," *Computers and Operations Research*, vol. 38, no. 1, pp. 3–13, 2011.

[7] E. L. Demeulemeester and W. S. Herroelen, "New benchmark results for the resource-constrained project scheduling problem," *Management Science*, vol. 43, no. 11, pp. 1485–1492, 1997.

[8] A. Sprecher and Arno, "Scheduling resource-constrained projects competitively at modest memory requirements," *Management Science*, vol. 46, no. 5, pp. 710–723, 2000.

[9] M. Davari and E. Demeulemeester, "A novel branch-and-bound algorithm for the chance-constrained resource-constrained project scheduling problem," *International Journal of Production Research*, vol. 57, no. 4, pp. 1265–1282, 2019.

[10] A. S. Chaleshtarti, S. Shadrokh, and Y. Fathi, "Branch and bound algorithms for resource constrained project scheduling problem subject to nonrenewable resources with prescheduled procurement," *Mathematical Problems in Engineering*, vol. 2014, no. 30, 15 pages, 2014.

[11] R. Pellerin, N. Perrier, and F. Berthaut, "A survey of hybrid metaheuristics for the resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 280, no. 2, pp. 395–416, 2020.

[12] R. Zamani, "An evolutionary implicit enumeration procedure for solving the resource-constrained project scheduling problem," *International Transactions in Operational Research*, vol. 24, no. 6, pp. 1525–1547, 2015.

[13] J. He, X. Chen, and X. Chen, "A filter-and-fan approach with adaptive neighborhood switching for resource-constrained project scheduling," *Computers and Operations Research*, vol. 71, pp. 71–81, 2016.

[14] L. Bukata, P. Šucha, and Z. Hanzálek, "Solving the resource constrained project scheduling problem using the parallel tabu search designed for the CUDA platform," *Journal of Parallel and Distributed Computing*, vol. 77, no. 1, pp. 58–68, 2015.

[15] L. Bukata and P. Šucha, "A gpu algorithm design for resource constrained project scheduling problem," in *Proceedings of the 21st Euromicro International Conference on Parallel*, pp. 367–374, Distributed and Network-Based Processing, Belfast, UK, March 2013.

[16] T. C. E. Cheng, Q. Ding, and B. M. T. Lin, "A concise survey of scheduling with time-dependent processing times," *European Journal of Operational Research*, vol. 152, no. 1, pp. 1–13, 2004.

[17] A. Janiak, T. Krysiak, and R. Trela, "Scheduling problems with learning and ageing effects: a survey," *Decision Making in Manufacturing and Services*, vol. 5, no. 1, pp. 19–36, 2011.

[18] W. W. Liu, C. Jiang, J. B. Wang, and Y. Y. Lu, "Single-machine scheduling with simultaneous considerations of resource allocation and deteriorating jobs," *The Computer Journal*, vol. 62, no. 1, pp. 81–89, 2019.

[19] J. B. Wang and X. X. Liang, "Group scheduling with deteriorating jobs and allotted resource under limited resource availability constraint," *Engineering Optimization*, vol. 51, no. 2, pp. 231–246, 2019.

[20] S. Gawiejnowicz and M. Kolińska, "Two- and three-machine open shop scheduling using LAPT-like rules," *Computers and Industrial Engineering*, vol. 157, no. 1, pp. 107261–107269, 2021.

[21] C. C. Wu, A. Azzouz, I. H. Chung, W. C. Lin, and L. Ben Said, "A two-stage three-machine assembly scheduling problem with deterioration effect," *International Journal of Production Research*, vol. 57, no. 21, pp. 6634–6647, 2019.

[22] J. B. Wang, J. J. Wang, and P. Ji, "Scheduling jobs with chain precedence constraints and deteriorating jobs," *Journal of the Operational Research Society*, vol. 62, no. 9, pp. 1765–1770, 2011.

[23] J. B. Wang and J. J. Wang, "Single-machine scheduling problems with precedence constraints and simple linear deterioration," *Applied Mathematical Modelling*, vol. 39, no. 3-4, pp. 1172–1182, 2015.

[24] L. Yang and X. W. Lu, "Approximation algorithms for some position-dependent scheduling problems," *Discrete Applied Mathematics*, vol. 289, no. 9, pp. 22–31, 2021.

[25] P. S. Sundararaghavan and A. S. Kunnathur, "Single machine scheduling with start time dependent processing times: some solvable cases," *European Journal of Operational Research*, vol. 78, no. 3, pp. 394–403, 1994.

[26] G. Mosheiov, "Scheduling jobs with step-deterioration minimizing makespan on a single- and multi-machine," *Computers and Industrial Engineering*, vol. 28, no. 4, pp. 869–879, 1995.

[27] C. X. Miao and Y. Z. Zhang, "Scheduling with step-deteriorating jobs to minimize the makespan," *Journal of Industrial and Management Optimization*, vol. 15, no. 4, pp. 1955–1964, 2019.

[28] P. Guo, W. M. Cheng, and Y. Wang, "A general variable neighborhood search for single-machine total tardiness scheduling problem with step-deteriorating jobs," *Journal of Industrial and Management Optimization*, vol. 10, no. 4, pp. 1071–1090, 2014.

[29] W. M. Cheng, P. Guo, Z. Q. Zhang, M. Zeng, and J. Liang, "Variable neighborhood search for parallel machines scheduling problem with step deteriorating jobs," *Mathematical Problems in Engineering*, vol. 2012, no. 6, Article ID 928312, 20 pages, 2012.

[30] C. C. He, C. C. Wu, and W. C. Lee, "Branch-and-bound and weight-combination search algorithms for the total completion time problem with step-deteriorating jobs," *Journal of the Operational Research Society*, vol. 60, no. 12, pp. 1759–1766, 2009.

[31] A. A. K. Jeng and B. M. T. Lin, "Minimizing the total completion time in single-machine scheduling with step-deteriorating jobs," *Computers and Operations Research*, vol. 32, no. 3, pp. 521–536, 2005.

[32] H. F. Dai and W. M. Cheng, "A memetic algorithm for multi-skill resource-constrained project scheduling problem under linear deterioration," *Mathematical Problems in Engineering*, vol. 2019, no. 1, Article ID 9459375, 16 pages, 2019.

[33] J. Layegh, F. Jolai, and M. S. Amalnik, "A memetic algorithm for minimizing the total weighted completion time on a single machine under step-deterioration," *Advances in Engineering Software*, vol. 40, no. 10, pp. 1074–1077, 2009.

[34] H. F. Dai, W. M. Cheng, and P. Guo, "An improved tabu search for multi-skill resource-constrained project scheduling

problems under step-deterioration," *Arabian Journal for Science and Engineering*, vol. 43, no. 6, pp. 3279–3290, 2018.

[35] G. Stecco, J. F. Cordeau, and E. Moretti, "A tabu search heuristic for a sequence-dependent and time-dependent scheduling problem on a single machine," *Journal of Scheduling*, vol. 12, no. 1, pp. 3–16, 2009.

[36] 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, 2020.

[37] S. C. 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, no. 3, pp. 1430–1442, 2021.

[38] F. Zhao, R. Ma, and L. Wang, "A self-learning discrete jaya algorithm for multi-objective energy-efficient distributed no-idle flow-shop scheduling problem in heterogeneous factory system," *IEEE Transactions on Cybernetics*, vol. 52, pp. 1–12, 2021.

[39] F. Q. Zhao, X. T. Hu, L. Wang, J. Zhao, J. Tang, and Jonrinaldi, "A reinforcement learning brain storm optimization algorithm (BSO) with learning mechanism," *Knowledge-Based Systems*, vol. 235, no. 1, Article ID 107645, 2022.

[40] F. Q. Zhao, S. L. Di, L. Wang et al., "A hyperheuristic with Q-learning for the multiobjective energy-efficient distributed blocking flow shop scheduling problem," *IEEE Transactions on Cybernetics*, vol. 53, no. 5, pp. 3337–3350, 2023.

[41] W. B. Carlton and J. W. Barnes, "Solving the traveling-salesman problem with time windows using tabu search," *IIE Transactions*, vol. 28, no. 8, pp. 617–629, 1996.

[42] Y. Wang, Q. H. Wu, and F. Glover, "Effective metaheuristic algorithms for the minimum differential dispersion problem," *European Journal of Operational Research*, vol. 258, no. 3, pp. 829–843, 2017.

[43] W. B. Carlton and J. W. Barnes, "A note on hashing functions and tabu search algorithms," *European Journal of Operational Research*, vol. 95, no. 1, pp. 237–239, 1996.

[44] R. Kolisch and A. Sprecher, "PSPLIB-a project scheduling problem library," *European Journal of Operational Research*, vol. 96, no. 1, pp. 205–216, 1997.

[45] X. J. Lai, J. K. Hao, and D. Yue, "Two-stage solution-based tabu search for the multi-demand multidimensional knapsack problem," *European Journal of Operational Research*, vol. 274, no. 1, pp. 35–48, 2019.

[46] H. F. Rahman, R. K. Chakrabortty, and M. Ryan, "An Effective Memetic Algorithm for Resource Constrained Project Scheduling Problem," in *Proceedings of the IEEE Congress on Evolutionary Computation*, Wellington, New Zealand, June 2019.

[47] G. Koulinas, L. Kotsikas, and K. Anagnostopoulos, "A particle swarm optimization based hyper-heuristic algorithm for the classic resource constrained project scheduling problem," *Information Sciences*, vol. 277, no. 1, pp. 680–693, 2014.

[48] S. Elsayed, R. Sarker, T. Ray, and C. Coello, "Consolidated optimization algorithm for resource-constrained project scheduling problems," *Information Sciences*, vol. 418-419, no. 1, pp. 346–362, 2017.

[49] E. L. Demeulemeester and W. S. Herroelen, "New benchmark results for the resource-constrained project scheduling problem," *Management Science*, vol. 43, no. 11, pp. 1485–1492, 1997.

[50] M. Vanhoucke and J. Coelho, "A tool to test and validate algorithms for the resource-constrained project scheduling problem," *Computers and Industrial Engineering*, vol. 118, no. 1, pp. 251–265, 2018.