Hindawi
Journal of Advanced Transportation
Volume 2020, Article ID 8897710, 11 pages
https://doi.org/10.1155/2020/8897710

Research Article

Effective Evolutionary Algorithm for Solving the Real-Resource-Constrained Scheduling Problem

Huu Dang Quoc,[1] **Loc Nguyen The**,[2] **Cuong Nguyen Doan**,[3] **and Naixue Xiong**[4]

[1]*Thuong Mai University, Hanoi, Vietnam*
[2]*Hanoi National University of Education, Hanoi, Vietnam*
[3]*Military Institute of Science and Technology, Hanoi, Vietnam*
[4]*Northeastern State University, Tahlequah, OK, USA*

Correspondence should be addressed to Huu Dang Quoc; huudq@tmu.edu.vn and Loc Nguyen The; locnt@hnue.edu.vn

Received 20 August 2020; Revised 19 September 2020; Accepted 30 September 2020; Published 14 October 2020

Academic Editor: Hongju Cheng

This paper defines and introduces the formulation of the Real-RCPSP (Real-Resource-Constrained Project Scheduling Problem), a new variant of the MS-RCPSP (Multiskill Resource-Constrained Project Scheduling Problem). Real-RCPSP is an optimization problem that has been attracting widespread interest from the research community in recent years. Real-RCPSP has become a critical issue in many fields such as resource allocation to perform tasks in Edge Computing or arranging robots at industrial production lines at factories and IoT systems. Compared to the MS-RCPSP, the Real-RCPSP is supplemented with assumptions about the execution time of the task, so it is more realistic. The previous algorithms for solving the MS-RCPSP have only been verified on simulation data, so their results are not completely convincing. In addition, those algorithms are designed only to solve the MS-RCPSP, so they are not completely suitable for solving the new Real-RCPSP. Inspired by the Cuckoo Search approach, this literature proposes an evolutionary algorithm that uses the function Reallocate for fast convergence to the global extremum. In order to verify the proposed algorithm, the experiments were conducted on two datasets: (i) the iMOPSE simulation dataset that previous studies had used and (ii) the actual TNG dataset collected from the textile company TNG. Experimental results on the iMOPSE simulation dataset show that the proposed algorithm achieves better solution quality than the existing algorithms, while the experimental results on the TNG dataset have proved that the proposed algorithm decreases the execution time of current production lines at the TNG company.

1. Introduction

Scheduling is used to arrange the resources and tasks in many fields, where scheduling algorithms can have an important impact on the effectiveness and cost. In a logistic system, not only are cargo vehicles characterized by the factors of speed and carrying capacity, but also the skill of the driver is the most important factor in determining the quality of transportation. By taking into account all the factors, especially the driver's skill one, scheduling algorithms help manage and coordinate the transport system. An intelligent scheduling algorithm helps managers exploit the maximum potential of resources including vehicle and driver to get the project done.

In the wireless sensor networks, the node scheduling aims at selecting a set of nodes (i.e., sensors) that provide the data service. This scheduling can effectively reduce the number of nodes and messages and at the same time extend the network lifetime [1–3]. The basic goal of Edge Computing [4] is finding the optimal scheduling for extending the Cloud's resources such as servers and routers from remote data centers to the edge of the Cloud where they are closer to users, thus overcoming the bottlenecks issue by cloud computing and providing higher performance.

Solving the MS-RCPSP [5–7] problem is to find out the schedule to execute the project in the shortest possible time without breaking any constraints. In other words, the scheduling algorithm's goal is to find a schedule with the

smallest execution time while meeting any task and resource constraints. In this paper, the term "makespan" will be used to refer to the distance in time that elapses from the start of work to the end of execution time. MS-RCPSP is among the most commonly investigated optimization problems that have received a lot of attention due to their significant role in network resource scheduling and controlling.

MS-RCPSP appears in many practical situations such as logistics and cargo transportation, widely applied to military operations such as sorting missions and determining travel routes [8]. Hosseinian and Baradaran [9] used an evaluation method to make plan decisions with MS-RCPSP. Nazafzad et al. [10] employed a biobjective optimization model for the MS-RCPSP considering shift differential payments and time-of-use electricity tariffs. Their study tried to minimize the cost and the makespan of a given project. Younis and Yang [11] propose the heuristic algorithm to solve a particular case of the MS-RCPSP occurring in grid computing.

However, the MS-RCPSP has one serious defect. What often happens in practice is that a resource with a higher skill level has a shorter processing time. This paper presents a new problem, which is a more practical extension of the MS-RCPSP, called the Real-RCPSP. In other words, Real-RCPSP is a specific case of the MS-RCPSP. In the Real-RCPSP, the processing time depends on the skill level of the resource. The real-life nature of Real-RCPSP comes from production lines at the industry factory, where the higher the skill level of a worker is, the faster he can make the product.

This paper is organized as follows. The next section presents some previous algorithms for solving the Resource-Constrained Scheduling Problem. This section also briefly introduces the Cuckoo Search strategy [12], one of the most widely used metaheuristics. Section 3 formulates the Real-RCPSP. The proposed algorithm (called R-CSM) is described in the fourth section. Section 4 introduces the most important components of R-CSM, consisting of the function Reallocate, the schedule representation, and a novel schedule measurement model. To verify the performance of the proposed algorithm, in Section 5 and Section 6, we arrange the experiments on the iMOPSE dataset and TNG's dataset, respectively. In these two sections, the experimental results are analyzed to compare the performance of the proposed algorithm R-CSM with the best previous algorithms such as GreedyDO and GA. Finally, Section 7 ends the paper with the conclusion and future works.

2. Related Works

Despite the importance of the Real-RCPSP, no one to the best of our knowledge has studied this problem. This paper is the first work that mentions Real-RCPSP; thus this section introduces the existing algorithms to solve another problem that is close to the Real-RCPSP, namely, MS-RCPSP. In Section 5 and Section 6, these algorithms will be used as reference algorithms in our experiments.

Myszkowski et al. [6] proved that MS-RCPSP is an NP-hard problem, so it is difficult to deal with classical optimization methods. Until now, many different solutions have been introduced for solving the MS-RCPSP; among them, the most successful metaheuristics are the GA [13] and ACO [14].

In their research, Maghsoudlou et al. [15] and Bibiks et al. [16] applied the Cuckoo Search algorithm to build multirisk project implementation schedules based on three different evaluation objectives. Zhu et al. [17] proposed an evolutionary algorithm based on the multiverse and several other heuristic algorithms.

Myszkowski et al. [6] have built a hybrid algorithm that combined the Difference Evolution and greedy heuristic for managing human and machine resources in factory production projects. The proposed hybrid algorithm tried to minimize makespan and production costs. Besides, iMOPSE [6], a dataset that was generated based on real-world data from the project scheduling problem, was introduced.

As a specific case of the MS-RCPCP problem, Real-RCPSP is researched and applied in many fields of the Internet of Things. Hosseinian and Baradaran [18] proposed a greedy heuristic for maximizing the modularity to find high-quality communities of employees and to arrange them to the tasks based on the founded communities. Younis and Yang [11] introduced a hybrid scheduling algorithm for task arrangement in grid computing environment.

Some other researchers have also studied the new extension problems of the Constrained Project Scheduling Problem and applied them in many fields of science and finance. Polo-Mejia et al. [19] developed a scheduling algorithm to manage nuclear laboratory operations. To solve the problem of the dense sensors in wireless sensor networks, Wan et al. [20] proposed an energy-saving scheduling algorithm, which arranges some redundant sensors into the sleep mode to reduce the data transmission collision and energy dissipation. Guo et al. [21] developed a PSO-based algorithm that acquired better performance than previous approaches in power efficiency. Cheng et al. [22] have formulated another PSO-based algorithm named DPSO-CA, which is based on the discrete PSO that aims at minimizing the cochannel interference in the network.

Previous studies have also been performed to address other subissues of the Constrained Project Scheduling Problem. Barrios et al. [23] and Javanmard et al. [24] studied the Multiskill Stochastic and Preemptive Scheduling Problem to minimize the execution time and proposed the mathematical models for the project's resource investment.

Cuckoo Search (CS) algorithm is a metaheuristic introduced by Yang [25] based on the cuckoo bird behavior. Previous algorithms such as Difference Evolution (DE) [26] and Particle Swarm Optimization (PSO) [21] have been proven to be special cases of the Cuckoo Search algorithm. The efficiency of CS has also been shown to be better than those of DE and PSO in some cases [27]. For the above reasons, in this paper, we have built an algorithm inspired by CS.

3. Problem Statement

The Real-RCPSP can be described as follows.

A project represented by a graph $G(V, E)$ has to be realized. Each node of graph G represents a task, while G's

arc represents the relationship between 2 tasks (Figure 1). Specifically, the arc (i, j) means that task i has to be finished by the time when task j is started. Each task has an execution time (or duration) that is calculated by subtracting the start time from the end time. The task must be performed continuously from start to finish and must not be stopped at all.

The execution of each task requires some specific resources. Each resource can perform only one task at a time. The task's execution requires several skills, while each resource possesses its own skills; thus not every resource can perform a given task.

The objective of the Real-RCPSP is to shorten the project implementation time to the smallest value while not breaking any constraints. A schedule must be found in which execution will minimize execution time while still meeting the task and resource constraints. As mentioned above, the MS-RCPSP is proved to be NP-hard [5–7], and no polynomial-time algorithm exists, assuming that $P \neq \text{NP}$.

Real-RCPSP could be stated by using the following notations:

(i) C_i: the set of the parents of task i

(ii) r_i: the set of skills required by task i. A certain resource must have an equal or higher skill level of r_j to perform task j

(iii) S: the set of all skills; S_i: the set of skills belonging to resource i; $S_i \subseteq S$

(iv) t_{jk}: the time it takes the resource that possesses subset of skill S_k to complete task j

(v) L: the set of resources; L^k: the set of resources that could handle task k; $L^k \subseteq L$

(vi) L_i: resource i

(vii) W: the set of tasks; W^k: the set of tasks that could be performed by resource k, $W^k \subseteq W$

(viii) W_i: task i

(ix) B_k, E_k: starting time and ending time of task k

(x) $A^i_{u,v}$: a Boolean variable; when it equals 1 it means that task u will be executed by resource v at time I; it equals 0 in other cases

(xi) h_i: the level of skill i; g_i: type of skill i

(xii) r_k: a resource must possess skill r_k to perform task k

(xiii) m: the period time of the schedule

(xiv) P: a candidate schedule; P_{all}: the set of candidate schedules

(xv) $f(P)$: makespan (execution time) of schedule P

(xvi) y: number of tasks; z: number of resources

Real-RCPSP could be defined as follows:

$$\text{minimize } f(P), \tag{1}$$

where

$$f(P) = \max_{i \in T}\{E_i\} - \min_{k \in T}\{B_k\}, \tag{2}$$

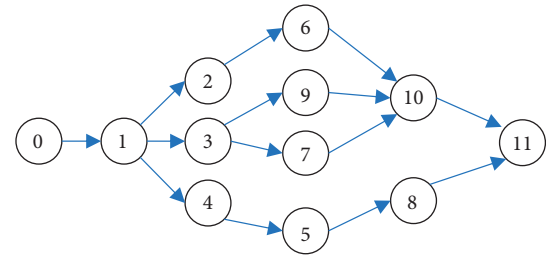

Figure 1: The relationship between tasks.

subject to

$$S_k \neq \phi, \quad \forall k \in L, \tag{3}$$

$$T_j \geq 0, \quad \forall j \in W, \tag{4}$$

$$E_j \geq 0, \quad \forall j \in W, \tag{5}$$

$$E_i \leq E_j - t_{j,} \quad \forall j \in W, j \neq 1, i \in C_j, \tag{6}$$

$$\forall i \in W_k \, \exists r \in S^k: \; g_r = g_{r_i}, h_r \geq h_{r_i}, \tag{7}$$

$$\forall k \in L, \forall t \in m: \; \sum_{i=1}^{n} A^t_{i,k} \leq 1, \tag{8}$$

$$\forall j \in W \, \exists! t \in m, !k \in L: \; A^t_{j,k} = 1, \tag{9}$$

$$\text{If } h_i < h_j \text{ then } t_{jk} > t_{ik} \, \forall \left(r_i, r_j\right) \in \{S_k \times S_v\}. \tag{10}$$

Note the following:

(i) Formulation (6) forced the parent task to must be finished before the start time of the children task

(ii) Formulation (7) means that, for every task, there is always at least one resource that has enough skill level to handle that task

(iii) Formulation (8) ensures that each resource (k) can only perform at most one task (j) at any time (t)

(iv) Constraint (9) aims to restrict each task to only be executed at most one time. Every task must be performed continuously from start to finish and must not be stopped at all.

(v) Constraint (10) means that the execution time of the higher-skill resources is smaller than the execution time of the lower-skill resources

4. Proposed Algorithm

4.1. Schedule Representation. We represent a schedule as a row that consists of several elements, and the number of elements denotes the number of tasks. Each element of the row represents the resource that will perform the respective task.

4.1.1. Example 1. Suppose that we have 10 tasks $W = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ and 3 resources $L = \{1, 2, 3\}$. Assume the following:

(i) $S_k = S, \forall k \in L$; every resource has an equal skill set

(ii) $L^k = L, \forall k \in W$; any resource can perform any task

(iii) The execution times of the tasks are presented in Table 1

We also assume that the constraint to prioritize the performance of the tasks is shown in Figure 1, specifically:

(i) Task 1 has to be performed firstly

(ii) Task 6 has to be performed after task 2

(iii) Tasks 7 and 9 have to be performed after task 3

(iv) Task 5 has to be performed after task 4

(v) Task 8 has to be performed after task 5

(vi) Task 10 has to be performed after tasks 6, 7, and 9

With the above assumptions and constraints, a possible schedule is shown in Figure 2. Table 2 shows how that schedule assigns 3 resources to perform 10 tasks in detail.

As described in Table 2, resource 1 executes task 2 and task 6; resource 2 handles tasks 1, 3, 5, and 8; resource 3 executes tasks 4, 7, 9, and 10.

4.2. Measurement Model. Cuckoo Search algorithm is an optimization scheme dealing with real functions such as the Gaussian probability distribution function, whereas Real-RCPSP is the optimization problem of discrete functions. Therefore, in order to apply the Cuckoo Search algorithm to the Real-RCPSP, it is necessary to build a model for schedules measuring. The following will present our proposed measurement model in detail:

(i) $V = (v_1, v_2, \ldots, v_n)$ is called "unit vector," where $v_i = 100/(z_i - 1)$: z_i is the number of resources that possess set of skills q_i.

(ii) Vector $K = \{k_1, k_2, \ldots, k_n\}$ is the distance between schedule $P = \{p_1, p_2, \ldots, p_n\}$ and schedule $Q = \{q_1, q_2, \ldots, q_n\}$. This leads to $K = P - Q$.

Meanwhile, if schedule $Q = \{q_1, q_2, \ldots, q_n\}$ is added with a difference $K = \{k_1, k_2, \ldots, k_n\}$, schedule $P = \{p_1, p_2, \ldots, p_n\}$ is obtained, where we have the following:

(iii) $p_i = \text{position}(\text{round}(q_i + k_i))$ and position (i) presents the respective resource

(iv) $k_i = v_i \times (\text{order}(p_i) - \text{order}(q_i))$

order (p_i): the place of p_i in the L_i

4.2.1. Example 2. Suppose that $L^1 = \{L_1, L_3, L_4, L_9, L_{10}\}$. We have $z_1 = 5$; $v_1 = 100/(5 - 1) = 25$.

Table 1: Execution times of the tasks.

Task	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}
Execution time	3	2	3	4	3	4	3	6	2	6

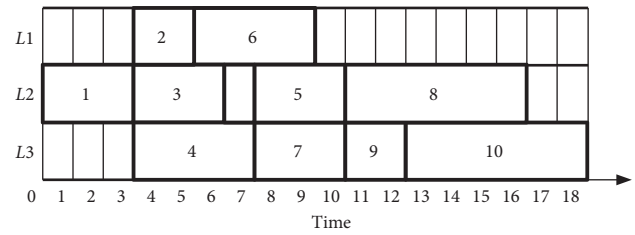

Figure 2: A possible schedule.

Table 2: The assignment of a possible schedule.

Task	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}
Resource	L_2	L_1	L_2	L_3	L_2	L_1	L_3	L_2	L_3	L_3

Table 3: The order of resources.

Order	0	1	2	3	4
Resource	L_1	L_3	L_4	L_9	L_{10}
Resource	L_6	L_7	L_8	L_{10}	

Similarly, suppose that $L^1 = \{L_6, L_7, L_8, L_{10}\}$. We have $z_2 = 4$; $v_2 = 100/(4 - 1) = 33.33$.

Table 3 depicts the order of resources.

Schedule $P = (1, 8)$ and schedule $Q = (4, 7)$ are shown in Table 4.

Consider the distance $K = P - Q = (k_1, k_2)$ where $k_1 = v_1 \times \text{abs}(\text{order}(p_1) - \text{order}(q_1)) = 25\text{abs}(0 - 2) = 25$.

$k_2 = v_2 \times \text{abs}(\text{order}(p_2) - \text{order}(q_2)) = 33.33 \times \text{abs}(2 - 1) = 33.33$.

This leads to K = $(50, 33.33)$.

Given $K_P = (0, 66.66)$, we have $Z = P + K \longrightarrow K_P + K \longrightarrow (4, 10)$ (Table 5).

4.3. Proposed Algorithm R-CSM. The proposed Algorithm 1 R-CSM is represented as follows.

f is objective function.

Note that, in line number 16, function *Reallocate()* improves the quality of *b_plan*, as analyzed in the next subsection.

4.4. Function Reallocate. c_plan is the most appropriate feasible schedule until now. L_b is the last resource to finish. Function Late() will find out the value of R_b.Size() is the size of a set or an array.*N*_makespan is execution time of the new resource-task arrangement

Table 4: The assignment of schedules P and Q.

Schedule	Task	
	1	2
P	L_1	L_8
Q	L_4	L_7

Table 5: Measurement value.

Task	1	2
P	L_1	L_8
K_P	0	66.66
K	50	33.33
$K_P + K$	50	99.99
$Z = K_P + K$	L_4	L_{10}

```
    input: maxGeneration
           iMOPSE datasets
    output: makespan of project
(1) Begin
(2)   t ⟵ 0
(3)   Size ⟵ number of individuals (i.e. possible schedules)
(4)   p(t) ⟵ the first population
(5)   f(t) ⟵ the fitness, b_plan(bestnest), makespan
(6)   pa = 0.25
(7)   While (t < max_gen)
(8)     n_plan ⟵ create new nest by Lévy Flight
(9)     r_plan ⟵ Select random nest from P(t)
(10)    If (f(n_plan) < (r_plan))
(11)      r_plan = n_plan
(12)    End if
(13)    P(t) ⟵ Remove pa worst nest and replace by new nests, new nests created by Lévy Flight
(14)    F(t) ⟵ the fitness, b_plan, makespan
(15)    b_plan ⟵ Reallocate(b_plan)//schedule b_plan is improved by the//function Reallocate(), which is described in the next
    subsection in details.
(16)    t ⟵ t + 1
(17)  End while
(18)  return makespan
(19) End
```

Algorithm 1: R-CSM algorithm.

Line 12 and line 13 show that the new schedule (n_plan) is always equal to or better than the old schedule (c_plan) in terms of the makespan.

The function Reallocate() (Figure 3) generates the new schedule from the best schedule, so it inherits and promotes the advantages of the current population (Algorithm 2).

4.4.1. Example 3. Suppose that $W = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$; $L = \{1, 2, 3\}$.

We also assume that resource 1 can handle tasks 1, 2, 3, 4, 6, 8, 9, and 10; resource 2 can execute tasks 1, 3, 7, and 9; resource 3 can perform tasks 1, 4, 5, 8, 9, and 10.

The constraint regarding the task order is illustrated in Figure 1, and the task's execution time is shown in Table 1.

Table 6 depicts the schedule P; its makespan is equal to 18 as shown in Figure 4 in detail.

The function Reallocate uses schedule P as the input and arranges task 9 to resource 1 (see Table 7) instead of resource 3 as before.

The results point out that function *Reallocate* decreases the makespan from 18 to 17, as described in Figure 4.

5. Simulation with iMOPSE Dataset

5.1. Simulation Settings. To verify the performance of the R-CSM, the simulations are conducted by using iMOPSE dataset [6], which has been used by previous studies to examine algorithms such as GreedyDO and GA [28]. iMOPSE's instances have the following fields:

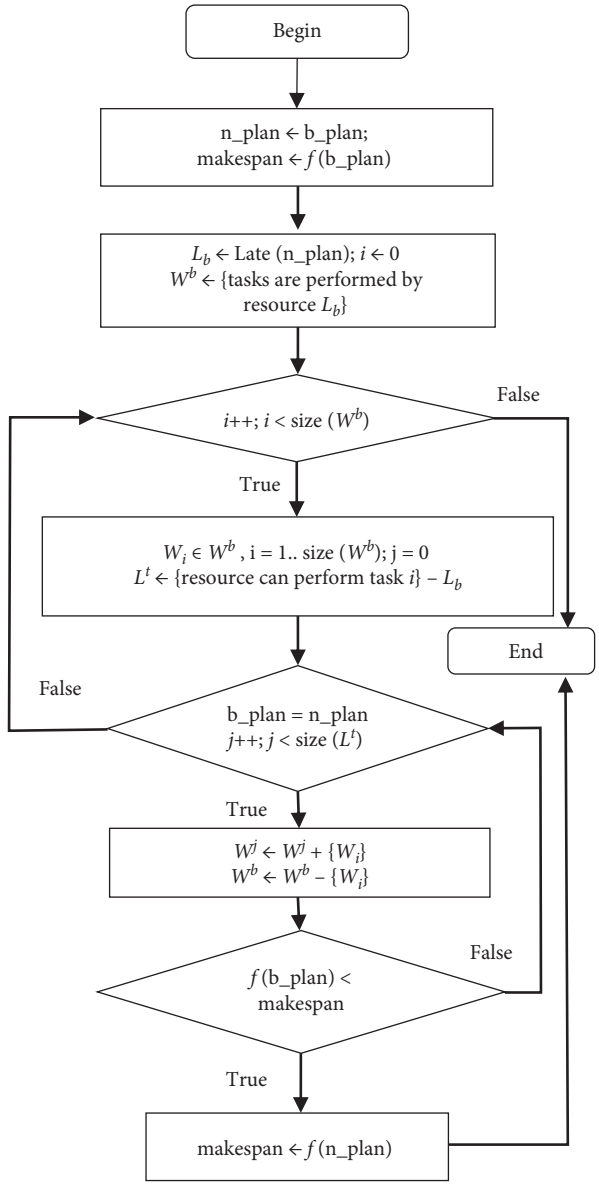

Figure 3: Function Reallocate.

(i) Number of tasks and resources

(ii) The constraint regarding the task order

(iii) Set of resource's skills

This paper arranges the simulations on iMOPSE's instances listed in Table 8. All of our simulations were run on a machine with Intel® Core i7 CPU at 2.2 GHz, 6 GB RAM, running Windows 10. Our R-CSM algorithm was programmed in Matlab. Simulation results are described in Table 9.

Each simulation was set up with parameters as follows:

(i) Dataset: 30 iMOPSE's instances that are described above

(ii) Number of individuals in population $N_p = 100$

(iii) The program execution process consists of 50,000 generations ($N_g = 50,000$)

(iv) Each instance was repeatedly executed 30 times

5.2. Simulation Results. To show the efficiency of the proposed algorithm, we compare R-CSM with two existing algorithms, which are GreedyDO and GA [28]. Myszkowski did not provide the tool for GreedyDO; thus Table 9 just lists the best value of the algorithm GreedyDO that was published in the author's literature. Meanwhile algorithm GA is reprogrammed using the GARunner, the tool provided by the authors in [6, 28]; thus Table 9 lists the average value, the best value, and the standard deviation value of the algorithm GA's makespan.

```
Input: b_plan//the best feasible schedule that the algorithm has found until now
Output://the feasible schedule which is better than the input
(1) Begin
(2)   makespan = f(b_plan)
(3)   n_plan = b_plan;//the best resource-task assignment plan so far
(4)   L_b ← Late(n_plan)//the resource finish its execution latest
(5)   W_b ←set of tasks is performed by resource L_b
(6)   For i = 1 to size(W_b)//examine the set of tasks performed by resource L_b
(7)       W_i = W_b[i];
(8)       L^i ← L − L_b L^i ← L − L_b//set of resource can perform the task i except L_b
(9)       For j = 1 to size(L^i)
(10)      W^j = W^j + {Wi}//task i will perform the resource L_j
(11)      W^b = W^b + {Wi}//task i is eliminated from L_b
(12)      n_makespan = f(n_plan)
(13)      If n_makespan < makespan
(14)      Makespan = n_makespan
(15)      Return b_best;
(16)      End if
(17)    b_plan = n_plan;
(18)    End for
(19)  End for
(20)  Return n_plan
(21) End Function
```

Algorithm 2: Function Reallocate.

Table 6: Resource-task assignment of P.

Task	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}
Resource	L_1	L_1	L_2	L_3	L_3	L_1	L_2	L_1	L_3	L_3

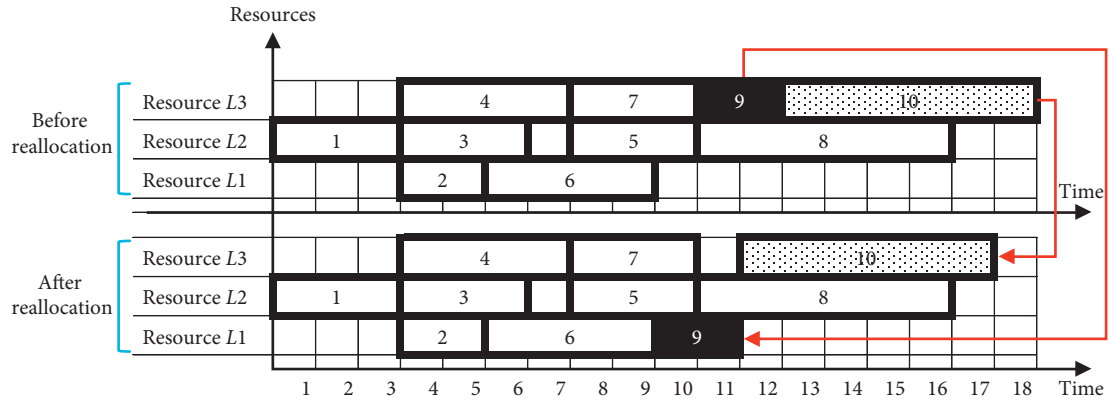

Figure 4: The schedule changes as a result of the function Reallocate.

Table 7: Resource-task assignment of new P.

Task	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}
Resource	L_1	L_1	L_2	L_3	L_3	L_1	L_2	L_1	L_1	L_3

Table 9 and Figure 5 demonstrated that the makespan of the R-CSM's schedules is smaller than the makespan of GreedyDO and GA. The comparison between algorithms is discussed as follows in detail:

(i) Compared with the original CS, the R-CSM algorithm is equipped with function Reallocate, which makes R-CSM capable of fast convergence. This ability is clearly demonstrated by the comparison of

Table 8: d36 iMOPSE dataset.

Dataset instance	Tasks	Resources	Precedence relations	Skills
100_5_22_15	100	5	22	15
100_5_46_15	100	5	46	15
100_5_48_9	100	5	48	9
100_5_64_15	100	5	64	15
100_5_64_9	100	5	64	9
100_10_26_15	100	10	26	15
100_10_47_9	100	10	47	9
100_10_48_15	100	10	48	15
100_10_64_9	100	10	64	9
100_10_65_15	100	10	65	15
100_20_22_15	100	20	22	15
100_20_46_15	100	20	46	15
100_20_47_9	100	20	47	9
100_20_65_15	100	20	65	15
100_20_65_9	100	20	65	9
200_10_128_15	200	10	128	15
200_10_50_15	200	10	50	15
200_10_50_9	200	10	50	9
200_10_84_9	200	10	84	9
200_10_85_15	200	10	85	15
200_20_145_15	200	20	145	15
200_20_54_15	200	20	54	15
200_20_55_9	200	20	55	9
200_20_97_15	200	20	97	15
200_20_97_9	200	20	97	9
200_40_133_15	200	40	133	15
200_40_45_15	200	40	45	15
200_40_45_9	200	40	45	9
200_40_90_9	200	40	90	9
200_40_91_15	200	40	91	15

Table 9: Makespan of algorithms (in hour).

Dataset instance	GreedyDO	GA			R-CSM		
		Avg	Best	Std	Avg	Best	Std
100_5_22_15	630	524	517	5	485	484	1.41
100_5_46_15	693	587	584	5	541	538	2.55
100_5_48_9	779	535	528	10	493	490	2.16
100_5_64_15	640	530	527	2	496	490	4.55
100_5_64_9	597	521	508	10	479	474	3.68
100_10_26_15	370	294	292	2	238	237	0.94
100_10_47_9	549	299	296	3	256	253	2.36
100_10_48_15	344	282	279	3	245	242	2.25
100_10_64_9	533	305	296	7	249	243	5.32
100_10_65_15	426	290	286	5	247	245	1.25
100_20_22_15	353	169	163	6	127	124	3.4
100_20_46_15	394	207	197	7	167	164	2.85
100_20_47_9	390	186	185	0	146	143	1.89
100_20_65_15	310	243	240	2	213	210	2.05
100_20_65_9	408	187	181	5	133	128	4.5
200_10_128_15	780	583	577	5	471	468	2.94
200_10_50_15	763	577	553	17	489	485	3.3
200_10_50_9	817	589	585	5	484	484	0.47
200_10_84_9	999	583	567	11	509	505	3.91
200_10_85_15	706	555	549	5	476	474	1.32
200_20_145_15	480	328	326	2	244	240	3.48
200_20_54_15	488	385	363	21	261	257	3.19
200_20_55_9	999	318	312	4	251	248	2.87
200_20_97_15	680	438	424	10	335	334	1.89
200_20_97_9	816	326	321	6	249	244	3.42
200_40_133_15	512	222	215	6	154	148	4.19
200_40_45_15	616	210	201	6	165	161	3.68
200_40_45_9	821	213	209	3	159	152	8.06
200_40_90_9	963	215	211	3	152	148	3.72
200_40_91_15	519	205	200	3	141	135	4.92

R-CSM with the previous most powerful algorithms. R-CSM's best value is smaller than GreedyDO from 21% to 85% and faster than GA from 6% to 33%. The average value of the R-CSM's makespan is better than the GA from 6% to 33%.

(ii) Thanks to the function Reallocate and the proposed measurement model, the process of finding the optimal schedule of the R-CSM is not only fast but also stable. This is demonstrated by the experimental results in Table 9. The total value of R-CSM's standard deviation is 92.52 only, whereas the total value of GA's standard deviation is equal to 180. This result shows that the R-CSM algorithm is more stable than the GA algorithm.

6. Experiment with TNG Dataset

6.1. Experimental Setting. In general, the major disadvantage of verifying on simulation datasets, such as iMOPSE, is that sometimes the results do not match what is actually happening. In order to make the experiment more convincing, we have collected and used the dataset of Investment and Trading Joint Stock Company (TNG) [29]. At TNG textile factory, the dataset construction is carried out as follows:

(i) The company TNG contracts with business partners, whereby each order corresponds to a product sample with a large quantity

(ii) A given order will be performed by a subset of employees

(iii) A product consists of several components, and each component takes an execution time

(iv) The skills of each worker are evaluated based on that worker's rank

Conducting experiments on a simulation dataset is always convenient because the parameters of the dataset are set by the experimenter; therefore these parameters are completely consistent with the problem formulation.

In contrast, the parameters of a real dataset are factory-defined, so they are not compatible with the conventions in the problem formulation. For this reason, before conducting experiments with TNG's dataset, the parameters of this dataset need to be converted to a format that matches the Real-RCPSP formulation.

This conversion is conducted as follows:

(i) Order is demonstrated by the project

(ii) Product's stage is depicted by a task

(iii) An employee is depicted by a resource

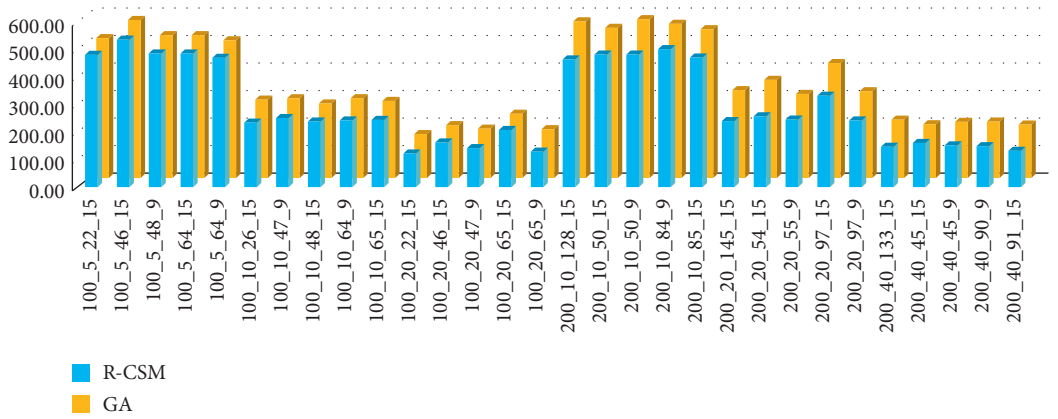

Figure 5: The comparison of performance between R-CSM and GA.

Table 10: TNG dataset.

Dataset instance	Name	Tasks	Resources	Precedence relations	Skill levels	Project time
71_37_1026_1	TNG1	71	37	1026	6	409
71_39_1026_1	TNG2	71	39	1026	6	325
71_41_1026_1	TNG3	71	41	1026	6	296
71_45_1026_1	TNG4	71	45	1026	6	392
137_37_1894_1	TNG5	137	37	1894	6	1174
137_39_1894_1	TNG6	137	39	1894	6	1052
137_41_1894_1	TNG7	137	41	1894	6	871
137_45_1894_1	TNG8	137	45	1894	6	996

Table 11: Makespan of GreedyDO, GA, and R-CSM (in hour).

Dataset instance	TNG	GreedyDO	GA	R-CSM
TNG1	409	236	201	166
TNG2	325	243	198	165
TNG3	296	258	212	168
TNG4	392	248	176	175
TNG5	1174	972	751	710
TNG6	1052	963	791	715
TNG7	871	834	810	727
TNG8	996	906	720	677

(iv) Employee's grade is depicted by the resource's skill level

(v) The manufacture sequence is denoted by the task's relationship

(vi) The execution time of the order is demonstrated by the makespan

The TNG dataset is described in Table 10.
Experiment setting is as follows:

(i) Dataset: 8 TNG's instances that are listed in Table 10

(ii) Number of individuals in population $N_p = 100$

(iii) The program execution process consists of 50,000 generations ($N_g = 50,000$)

(iv) Each instance was repeatedly executed 35 times

6.2. Experimental Results. The experiments in this section were conducted on the TNG dataset to prove that the proposed algorithm is more efficient than existing algorithms not only when they are operating on the simulated dataset such as iMOPSE but also when operating on the actual dataset.

Experimental results (listed in Table 11) demonstrated that the proposed algorithm R-CSM is not only more efficient than previous algorithms such as GreedyDO and GA but also more efficient than the actual production plan at the TNG factory, which is presented in column TNG.

As depicted in Table 11, compared to the execution time of the actual production plan at the factory TNG,the GA and GreedyDO algorithms reduce the makespan by 7%–55% and 4%–42%, respectively, while R-CSM has the best results of 17%–59%.

Table 12: Experimental results with Real-RCPSP.

Dataset instance	TNG (A)	MS-RCPSP (1)		Real-RCPSP (2)		(2) vs. (1)	
		Best	vs. (A) (%)	Best	vs. (A) (%)	Hours	%
TNG1	409	166	59	131	68	35	21.1
TNG2	325	165	49	133	59	32	19.4
TNG3	296	168	43	132	55	36	21.4
TNG4	392	175	55	127	68	48	27.4
TNG5	1174	710	40	572	51	138	19.4
TNG6	1052	715	32	626	40	89	12.4
TNG7	871	727	17	569	35	158	21.7
TNG8	996	677	32	560	44	117	17.3

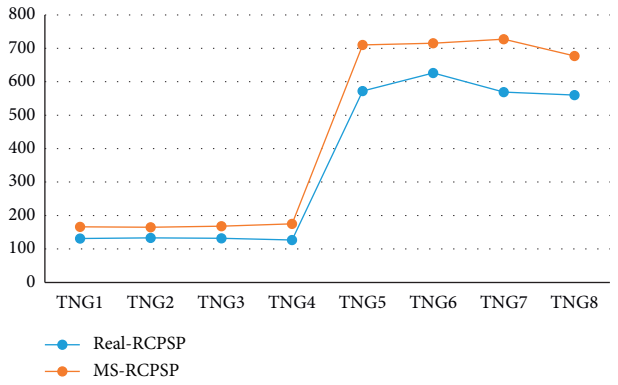

Figure 6: R-CSM to solve MS-RCPSP and Real-RCPSP.

As the best current evolutionary algorithms, GreedyDO and GA are both better than the actual production plan at the factory TNG. However, neither of these algorithms is as good as the proposed algorithm R-CSM. In experiments on the TNG datasets, R-CSM lessens the makespan from 17% to 59% compared to the current factory schedule. To sum up, the proposed algorithm R-CSM has been proven to be overall more effective compared to existing approaches such as GreedyDO, GA, and the current factory schedule.

Applying the R-CSM algorithm on the data from table Table 11 to solve the Real-RCPSP, we get the results shown in Table 12.

Experimental results in Table 12 show that, thanks to the application of the R-CSM algorithm to the actual problem with data on the textile production of TNG, the production time is reduced from 12.4% to 27%. These results also show that the greater the difference between the skill levels of workers, the more efficient the R-CSM algorithm.

Figure 6 shows the effectiveness of algorithm R-CSM when it is applied to solve the MS-RCPSP and the Real-RCPSP on the textile dataset of TNG textile company.

7. Conclusion

This article aims to announce and survey Real-RCPSP, a new combinatorial optimization problem that appears in many fields such as Edge Computing, industrial production, and IoT systems. The new problem is stated, and then a new algorithm named R-CSM is proposed. Inspired by the Cuckoo Search strategy, the proposed algorithm has been upgraded by using function Reallocate; thus it achieved high performance.

The experimental results show that the proposed algorithm R-CSM is better than the previous algorithms such as GreedyDO and GA at 21%–85% and 6%–33%, respectively. At the same time, the proposed algorithm converged to the optimal solution faster than previous approaches.

In the near future, we are going to continue researching on Real-RCPSP in order to improve the solution quality and the speed of the convergence. Multifactorial optimization seems to be one of the promising approaches to this problem.

Data Availability

The paper uses the standard iMOPSE dataset to test the efficiency of the algorithm. This dataset is publicly available at http://imopse.ii.pwr.wroc.pl/and is free of charge. In addition, the authors also tested the algorithm with TNG's garment manufacturing dataset. They have obtained permission from TNG to use these data.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors greatly acknowledge TNG Corporation (434/1 Bac Kan, Thai Nguyen, Vietnam) [29] for their cooperation and for allowing them to use their dataset regarding the production line.

References

[1] H. Cheng, Z. Su, N. Xiong, and Y. Xiao, "Energy-efficient node scheduling algorithms for wireless sensor networks using markov random field model," *Information Sciences*, vol. 329, no. 2, pp. 461–477, 2016.

[2] C. Lin, N. Xiong, J. H. Park, and T.-h. Kim, "Dynamic power management in new architecture of wireless sensor networks," *International Journal of Communication Systems*, vol. 22, no. 6, pp. 671–693, 2009.

[3] C. Lin, Y. X. He, and N. Xiong, "An energy-efficient dynamic power management in wireless sensor networks," in *Proceedingsof Fifth International Symposium on Parallel and-Distributed Computing*, IEEE, Timisoara, Romania, July 2006.

[4] Y. Zhou, D. Zhang, and N. Xiong, "Post-cloud computing paradigms: a survey and comparison," *Tsinghua Science and Technology*, vol. 22, no. 6, pp. 714–732, 2017.

[5] R. Klein, *Scheduling of Resource Constrained Project*, Springer Science Business Media, NewYork, NY, USA, 2000.

[6] Myszkowski, B. Paweł, E. Marek, Skowroński, and S. Krzysztof, "A new benchmark dataset for multi-skill resource-constrained project scheduling problem," in *Proceedings of 2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*, IEEE, Lodz, Poland, September 2015.

[7] J. B1azewicz, J. K. Lenstra, and A. H. G. Rinnooy Kan, "Scheduling subject to resource constraints: classification and complexity," *Discrete Applied Mathematics*, vol. 5, pp. 11–24, 1983.

[8] H. Li and K. Womer, "A decomposition approach for shipboard manpower scheduling," *Military Operations Research*, vol. 14, no. 3, pp. 1–24, 2009.

[9] A. H. Hosseinian and V. Baradaran, "An evolutionary algorithm based on a hybrid multi-attribute decision making method for the multi-mode multi-skilled resource-constrained project scheduling problem," *Journal of Optimization in Industrial Engineering*, vol. 12, no. 2, pp. 155–178, 2019.

[10] H. Najafzad, H. Davari-Ardakani, and R. Nemati-Lafmejani, "Multi-skill project scheduling problem under time-of-use electricity tariffs and shift differential payments," *Energy*, vol. 168, pp. 619–636, 2019.

[11] M. T. Younis and S. Yang, "Hybrid meta-heuristic algorithms for independent job scheduling in grid computing," *Applied Soft Computing*, vol. 72, pp. 498–517, 2018.

[12] X. S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proceedings of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009)*, IEEE Publications, Coimbatore, India, pp. 210–214, December 2009.

[13] W. Wu, A. R. Simpson, and H. R. Maier, "Accounting for greenhouse gas emissions in multiobjective genetic algorithm optimization of water distribution systems," *Journal of Water Resources Planning and Management*, vol. 136, no. 2, pp. 146–155, 2010.

[14] W. Deng, J. Xu, and H. Zhao, "An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem," *IEEE Access*, vol. 7, pp. 20281–20292, 2019.

[15] H. Maghsoudlou, B. Afshar-Nadjafi, and S. T. Akhavan Niaki, "Multi-skilled project scheduling with level-dependent rework risk; three multi-objective mechanisms based on cuckoo search," *Applied Soft Computing*, vol. 54, pp. 46–61, 2017.

[16] K. Bibiks, Y.-F. Hu, J.-P. Li, P. Pillai, and A. Smith, "Improved discrete cuckoo search for the resource-constrained project scheduling problem," *Applied Soft Computing*, vol. 69, pp. 493–503, 2018.

[17] L. Zhu, J. Lin, and Z.-J. Wang, "A discrete oppositional multiverse optimization algorithm for multi-skill resource constrained project scheduling problem," *Applied Soft Computing*, vol. 85, Article ID 105805, 2019.

[18] A. H. Hosseinian and V. Baradaran, "Detecting communities of workforces for the multi-skill resource-constrained project scheduling problem: a dandelion solution approach," *Journal of Industrial and Systems Engineering*, vol. 12, pp. 72–99, 2019.

[19] O. Polo-Mejía, C. Artigues, P. Lopez, and V. Basini, "Mixed-integer/linear and constraint programming approaches for activity scheduling in a nuclear research facility," *International Journal of Production Research*, pp. 1–18, 2019.

[20] R. Wan, N. Xiong, and N. T. Loc, "An energy-efficient sleep scheduling mechanism with similarity measure for wireless sensor networks," *Human-centric Computing and Information Sciences*, vol. 8, p. 18, 2018.

[21] W. Guo, N. Xiong, A. V. Vasilakos, G. Chen, and C. Yu, "Distributed k-connected fault-tolerant topology control algorithms with PSO in future autonomic sensor systems," *International Journal of Sensor Networks*, vol. 12, no. 1, pp. 53–62, 2012.

[22] H. Cheng, N. Xiong, A. V. Vasilakos, L. Tianruo Yang, G. Chen, and X. Zhuang, "Nodes organization for channel assignment with topology preservation in multi-radio wireless mesh networks," *Ad Hoc Networks*, vol. 10, no. 5, pp. 760–773, 2012.

[23] A. Barrios, F. Ballestín, and V. Valls, "A double genetic algorithm for the MRCPSP/max," *Computers & Operations Research*, vol. 38, no. 1, pp. 33–43, 2011.

[24] S. Javanmard, B. Afshar-Nadjafi, and S. T. Akhavan Niaki, "Preemptive multi-skilled resource investment project scheduling problem: mathematical modelling and solution approaches," *Computers & Chemical Engineering*, vol. 96, pp. 55–68, 2017.

[25] X. S. Yang, *Nature-Inspired Optimization Algorithms*, Elsevier, London, UK, 1st edition, 2014.

[26] M. I. Solihin and M. F. Zanil, "Performance comparison of Cuckoo search and differential evolution algorithm for constrained optimization," *International Engineering Research and Innovation Symposium (IRIS)*, vol. 160, no. 1, 7 pages, 2016.

[27] M. A. Adnan and M. A. Razzaque, "A comparative study of particle swarm optimization and cuckoo search techniques through problem-specific distance function," in *Proceedings of International Conference on Information and Communication Technology (ICoICT)*, Bandung, Indonesia, March 2013.

[28] P. B. Myszkowski, Ł. P. Olech, M. Laszczyk, and M. E. Skowroński, "Hybrid differential evolution and greedy algorithm (DEGR) for solving multi-skill resource-constrained project scheduling problem," *Applied Soft Computing*, vol. 62, pp. 1–14, 2018.

[29] "TNG Investment and trading joint stock company," http://www.tng.vn.