Hindawi
Scientific Programming
Volume 2020, Article ID 8860384, 12 pages
https://doi.org/10.1155/2020/8860384

Research Article

Intelligent Differential Evolution Scheme for Network Resources in IoT

Huu Dang Quoc,[1] **Loc Nguyen The**,[2] **Cuong Nguyen Doan**,[3] **Toan Phan Thanh**,[2] **and Neal N. Xiong**[4]

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

Correspondence should be addressed to Huu Dang Quoc; huudq@tmu.edu.vn

Received 5 June 2020; Accepted 19 June 2020; Published 14 July 2020

Academic Editor: Tingsong Wang

Scheduling is a fundamental factor in managing the network resource of the Internet of things. For IoT systems such as production lines, to operate effectively, it is necessary to find an intelligent management scheme, i.e., schedule, for network resources. In this study, we focus on multiskill resource-constrained project scheduling problem (MS-RCPSP), a combinational optimization problem that has received extensive attention from the research community due to its advantages in network resource management. In recent years, many approaches have been utilized for solving this problem such as genetic algorithm and ant colony optimization. Although these approaches introduce various optimization techniques, premature convergence issue also occurs. Moreover, previous studies have only been verified on simulation data but not on real data of factories. This paper formulated the MS-RCPSP and then proposed a novel algorithm called DEM to solve it. The proposed algorithm was developed based on the differential evolution metaheuristic. Besides, we build the reallocate function to improve the solution quality so that the proposed algorithm converges rapidly to global extremum while also avoiding getting trapped in a local extremum. To evaluate the performance of the proposed algorithm, we conduct the experiment on iMOPSE, the simulated dataset used by previous authors in their research studies. In addition, DEM was verified on real dataset supported by a famous textile industry factory. Experimental results on both simulated data and real data show that the proposed algorithm not only finds a better schedule compared with related works but also can reduce the processing time of the production line currently used at the textile industry factory.

1. Introduction

Internet of things systems include not only sensors and automated machines like robots but also intelligent control and management algorithms [1]. Intelligent algorithms for managing network resource play a key role in controlling the operation of automated machines such as robots, enabling them to devote their full capacity to serve users.

Multiskill resource-constrained project scheduling problem (MS-RCPSP), a combinational optimization problem, has received extensive attention from the research community in recent years due to its advantages in network resources management and scheduling for Internet of things system. The main purpose of the MS-RCPSP is to find the way for assigning tasks to resources so that the execution time is minimal.

In recent years, MS-RCPSP is widely used, and this problem is found not only in critical civilian fields but also in the military setting. MS-RCPSP appears on a broad scale of military operations such as planning an onboard mission or routing an unmanned aerial vehicle (UAV) [2, 3].

The rest of the paper is organized as follows. In Section 2, we present some related works such as previous approaches

to solving the MS-RCPSP. Specifically, the summary of the differential evolution method is included in this section. Differential evolution is a well-known evolutionary algorithm that has been applied effectively to solve NP-hard problems. In this paper, the proposed algorithm is inspired by the differential evolution method. In Section 3, we present the system model and the statement of MS-RCPSP. The formulation of MS-RCPSP is introduced in this section. Section 4 describes the proposed algorithm, which is built on differential evolution strategy [4]. In this section, we focus on presenting the reallocate function, which is the key component that creates the strength of the proposed algorithm. Besides, this section introduces our design for schedule representation and a new measurement model for measuring the difference between two schedules. The proposed algorithm (DEM) is also introduced in this section. Simulation results examining the proposed algorithms along with computational and comparative results are given and analyzed in Section 5. Simulations conducted by using the iMOPSE dataset [5] are presented in this section. In addition, this section describes experiments conducted on the factory's dataset collected from TNG, a well-known national textile company. Finally, Section 6 concludes the paper and outlines some directions for future work.

2. Related Works

2.1. Approximation Algorithms for MS-RCPSP. Scheduling problem arises in many practical situations [6–11]. In the general case, the scheduling is proved to be NP-hard. The authors of [5, 12, 13] show that MS-RCPSP, the problem we mentioned in this paper, is an NP-hard scheduling problem. In addition, MS-RCPSP is multimodal or discontinuous and very challenging to solve with traditional optimization methods. In the past few years, various approaches have been proposed to solve MS-RCPSP, thereby finding out intelligent algorithms to allocate and manage network resources in the Internet of things systems [14–17]. Some popular metaheuristic algorithms are the genetic algorithm (GA), ant colony optimization (ACO), and particle swarm optimization [18–22].

Su et al. [23] used a mixed-integer model and discrete constraints to solve the problems. Maghsoudlou et al. [24] and Bibiks et al. [25] applied the cuckoo Search algorithm to plan for the multirisk project with three distinct evaluation objectives. Lin et al. [26] proposed a new solution based on GA and some other heuristic algorithms.

Myszkowski et al. [5, 13] proposed the hybrid algorithm which is the combination of the differential evolution strategy and the greedy approach for arranging the human resources in product manufacturing projects and calculating time-of-use electricity tariffs and shift differential payments. The authors aimed to achieve the shortest makespan and low cost and gave a new benchmark dataset called iMOPSE [5], an artificially created dataset based on real-world instances for the MS-RCPSP.

The MS-RCPSP is studied and applied in the planning of many practical areas of the Internet of things. Some authors also study new variants of the MS-RCPSP and apply them in different fields. Mejia [3] developed a new variant of the MS-RCPSP to coordinate research activities in the nuclear laboratory. Hosseinian [27] proposed two new algorithms, P-GWO and MOFA, to solve the MS-RCPSP with the deterioration effect and financial constraints (a case study of a gas treating company). In another study, Hosseinian et al. introduced a new mixed-integer formulation for the time-dependent MS-RCPSP considering the learning effect [28].

Nemati-Lafmejani et al. [29] developed an integrated biobjective optimization model to deal with multimode RCPSP. The objective of the proposed model is to minimize both the costs and the makespan of the project. Tahrir and Yang [30] proposed a hybrid metaheuristic algorithm that combined ant colony optimization and variable neighbourhood search approaches for job scheduling in grid computing.

Several prior research has also been conducted to solve other subproblems of RCPSP, the general problem of MS-RCPSP, to improve the efficiency of network resource management in specific fields of the Internet of things. Ballestin and Leus [31] and Javanmard [32] and their colleagues studied specific cases of RCPSPs such as multiskill stochastic and preemptive problem to calculate project implementation time, i.e., makespan, and build mathematical models for the project resource investment.

2.2. Differential Evolution Method. Differential evolution (DE) is a population-based stochastic optimization algorithm introduced by Storn and Price [4]. The advantages of DE lies in its simplicity, efficiency, and speed thanks to the local search method. DE deals with the old generation of the original population by using the mutation operator to create better solutions in the new generation. Until now, DE is clearly recognized to be the approach that has the potential to solve NP-hard [4, 33] problems.

Tanabe et al. and Ghosh and Das [34, 35] proposed a DE-based method for scheduling problems in grid computing environment whose purpose is to minimize the execution time and add more parameter adjustment to get high effects.

Like other evolutionary approaches, DE performs the evolution of a population by using three kinds of the operator: crossover (recombination), mutation, and selection. The major difference between DE and genetic algorithm, a classical evolutionary algorithm, is due to the mutation operator. DE's mutation operator uses orientation information to change solutions of the current population as follows.

Given the population that is composed of solutions, each of them consists of D components, and thus a particular solution is represented by a vector $x_i = (x_{i,2}, x_{i,2}, \ldots, x_{i,D})$ where $x_{i,j} \in R$, $i = 1, 2, \ldots, N$; $j = 1, 2, \ldots, D$.

To create the mutated solution v_i of a given solution x_i, DE picks out three random different solutions from the current population: $x_{r1} \neq x_{r2} \neq x_{r3} \neq x_i$, and then the mutation solution v_i is determined as follows:

$$v_i = x_{r1} + F(x_{r2} - x_{r3}), \tag{1}$$

where the value of the constant F, which plays the role of the mutant factor, is in the range [0, 1]. Because the mutant factor F is used to adjust the size of the directional vector, it is also called the directional hop length.

After executing the mutation step, the crossover operation is carried out by combining the parent solution x_i and the mutation solution v_i. The crossover operator is performed by selecting a random number CR (CR $\in$ [0, 1]) as the probability of crossover. The result of the crossover step is represented by the vector ($u_{1,1}$, $u_{1,2}$, ... $u_{N,D}$) where

$$u_{i,j} = \begin{cases} v_{i,j}\, if\ \text{rand}_{i,j} \le \text{CR or } i = I_{\text{rand}} \\ x_{i,j}\, if\ \text{rand}_{i,j} \ge \text{CR or } i \ne I_{\text{rand}} \end{cases} \quad (2)$$

(i) $i = 1, 2, \ldots, N$; $j = 1, 2, \ldots, D$.

(ii) $\text{rand}_{i,j}$: mutant factor.

(iii) I_{rand}: a random number varies within the range [1,D]. Thanks to I_{rand}, the mutation solution $v_i$is always different from the parent solution x_i.

Finally, the next generation is created from the x_{i} and the u_{i} as follows:

$$x_i(t+1) = \begin{cases} u_i(t), & \text{if } f(u_i(t)) < f(x_i(t)), \\ x_i(t), & \text{in other cases.} \end{cases} \quad (3)$$

Since 1997, various versions of DE have been developed by researchers. This paper proposes a variant of the DE algorithm which uses the following terminologies and conventions:

(i) A solution represents a schedule, that is, a plan for performing given tasks

(ii) The number of search space's dimensions is equal to the number of tasks

3. System Model and Definitions

3.1. RCPSP. MS-RCPSP is just a special case of RCPSP. Before the outline of the multiskill RCPSP, the description of the classical RCPSP would be introduced as follows.

(i) Assume a given project represented by a directed noncyclical graph $G(V, E)$, where each node depicts a task and the arc represents finish-to-start precedence relationship between two tasks. The arc (u, v) $\in$ E shows that the task u must be completed before task $v$begins (Figure 1).

(ii) Without any difference, two empty tasks were added to the project. The first one is placed at the beginning of the project as the predecessor of every other task, whereas the second empty task is placed at the end of the process as the successor of other tasks.

(iii) The duration (also called execution time) of a certain task is calculated from the beginning time to the ending time of that task.

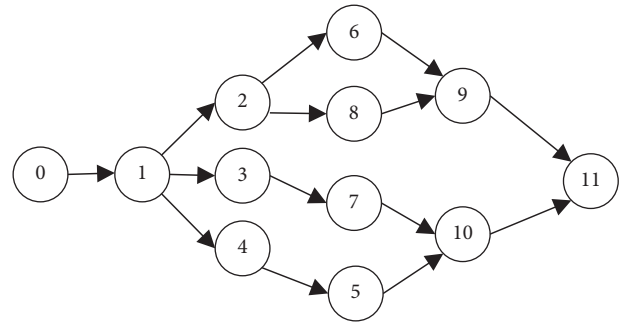

Figure 1: The relationship between tasks.

(iv) The execution progress of every task cannot be stopped until it is done.

(v) Each task needs an amount of resources for its execution.

The goal of the RCPSP is to find out a schedule of the task such that the precedence and resource constraints are satisfied and the makespan is minimized. RCPSP has been proved to be NP-hard [12]. Throughout the paper, we use the term "makespan" to refer to the execution time of the project, also called project duration. In the process of looking for the optimal schedule, both priority constraint between operations and resource constraint have to be met.

3.2. MS-RCPSP Formulation. In this paper, MS-RCPSP [5, 13, 36] was chosen because it is more practical than its general problem, RCPSP. MS-RCPSP uses the additional attribute, skill, in the problem formulation. Each task requires particular skills at the given levels to be executed, whereas each resource has some skills with given levels. Thus, the resource A is capable of performing the task B if A has skills required by B at the identical or higher level.

Before deepening the study of MS-RCPSP, we take a look at some notations as follows:

(i) d_j: the duration of task j; P_j: subset of the predecessors of task j

(ii) q_j: subset of skills required by task j

(iii) Q_k: the subset of skills owned by the resource k

(iv) Q: the set of skills, $Q_k \subseteq Q$

(v) RS: the set of resources

(vi) T: the set of tasks

(vii) T_k: the set of tasks that resource k can perform, $T_k \subseteq T$

(viii) RS_k: the set of resources that can perform task k, $RS_k \subseteq$ RS

(ix) S_j, F_j: start time and finish time of task j

(x) $U_{j,k}{}^t$: $U_{j,k}{}^t = 1$ in case of resource k is assigned to task j at given time t; $U_{j,k}{}^t = 0$ in other cases

(xi) l_q: the level of the given skill q; h_q: type of the skill q

(xii) q_j: the skill required by task j

(xiii) τ: duration of a schedule

(xiv) PS: a feasible schedule

(xv) PS_{all}: the set of all feasible schedules

(xvi) f(PS): makespan of schedule PS

Each schedule (PS) is responsible for coordinating n tasks and m resources.

Formally, MS-RCPSP could be stated as follows:

$$f(\mathrm{PS}) \longrightarrow \min, \tag{4}$$

where

$$f(PS) = \max_{F_i \in T}\{F_i\} - \min_{F_k \in T}\{F_k\}, \tag{5}$$

$$Q_k \neq \varnothing \, \forall k \quad \in \mathrm{RS}, \tag{6}$$

$$d_j \geq 0 \forall j \quad \in T, \tag{7}$$

$$F_j \geq 0 \forall j \quad \in T, \tag{8}$$

$$F_i \leq F_j \quad \forall j \in T,\ j \neq 1,\ i \in P_j, \tag{9}$$

$$\forall i \in T_k \, \exists q \in Q^k \quad : \ h_q = h_{q_i} \text{ and } l_q \geq l_{q_i}, \tag{10}$$

$$\forall k \in \mathrm{RS}, \quad \forall t \in \tau: \ \sum_{i=1}^{n} U_{i,k}^{t} \leq 1, \tag{11}$$

$$\forall j \in T \, \exists! t \in \tau, \quad !k \in \mathrm{RS}: U_{j,k}^{t} = 1, \text{ where } U_{j,k}^{t} \in \{0; 1\}. \tag{12}$$

4. Proposed Algorithm

4.1. Schedule Representation. In this study, every schedule or solution is represented by a vector that consists of elements; the number of elements (size of the vector) and the number of tasks are the same. The value of a certain element depicts the resource that can execute the corresponding task.

Example 1. Given a set of task $T = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ and a set of resource RS = {1, 2, 3}. The MS-RCPSP's goal is to find out a schedule in which the makespan can be minimized while meeting priority constraints between tasks (Figure 1).

Consider a set of task $T = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ and set of resource RS = {1, 2, 3}. Priority constraints between tasks are depicted in Figure 1, whereby task 1 is the predecessor of task 2, and thus task 2 cannot be performed until task 1 is finished: $S_2 \geq F_1$. There is no relation between task 3 and task 4; therefore, they are executed concurrently or sequentially.

Besides, in this example, we assume that

(i) $s_k = s_i \ \forall k, \ i \in \mathrm{RS}$

(ii) $Q_k = Q \ \forall k \in \mathrm{RS}$

(iii) $\mathrm{RS}_k = \mathrm{RS} \ \forall k \in \mathrm{RS}$

The duration of the tasks is given in Table 1.

Table 1: Task duration.

Task	1	2	3	4	5	6	7	8	9	10
Duration	2	4	3	5	2	2	5	3	4	2

A feasible schedule for the project is demonstrated in Figure 2, where it can be seen that the makespan is 15.

The above schedule is demonstrated in Table 2, where resource 1 performed tasks number 1, 2, 6, 8, and 9; resource 2 is assigned to execute tasks number 3 and 7; resource 3 deals with the remaining tasks.

4.2. Measurement Model. The differential evolution algorithm was primarily designed for real-valued data. However, like most of the other scheduling problems, MS-RCPSP deals with discrete values, and thus there is a need for the discrete variant of DE. In addition, this discrete DE algorithm needs a model for representing the MS-RCPSP's factors such as the difference between two schedules. In this paper, we build a new measurement model for measuring the difference between the two schedules as follows:

(i) Unit vector $P = (p_1, p_2, ..., p_n)$; $p_i = 100/(k_i - 1)$, where k_i is the number of resources that can execute task i

(ii) The difference between schedule $X = (x_1, x_2, ..., x_n)$ and schedule $Y = (y_1, y_2, ..., y_n)$ is denoted by the differential vector $D = X - Y = (d_1, d_2, ..., d_n)$

where

$$d_i = p_i \times (\mathrm{order}(x_i) - \mathrm{order}(y_i)), \tag{13}$$

order (x_i): the position of the resource (x_i) in the RS_i

(iii) The sum of the schedule Y and differential vector D is the schedule $X = (x_1, x_2, ..., x_n)$

where

$$x_i = \mathrm{position}(\mathrm{round}(y_i + d_i)), \tag{14}$$

position (i): the resource corresponding to the position i

Example 2. Assume that there are 6 resources that could handle task 1: $\mathrm{RS}_1 = \{R_1, R_3, R_4, R_5, R_9, R_{10}\}$. Thus, $k_1 = 6$; $p_1 = 100/(6-1) = 20$. Assume that there are 5 resources that could handle task 2: $\mathrm{RS}_2 = \{R_3, R_5, R_7, R_8, R_{10}\}$. Thus, $k_2 = 5$; $p_2 = 100/(5-1) = 25$. The resource order in measurement model is presented in Table 3.

Consider 2 schedules: $X = (3, 5)$; $Y = (5, 10)$. The task assignment is shown in Table 4.

$D = X - Y = (d_1, d_2)$, where $d_1 = p_1$. Abs (order (x_1) − order (y_1)) = 20. Abs (1–3) = 40.

$d_2 = p_2$. Abs (order (x_2) − order (y_2)) = 25. Abs (1–4) = 75. Thus, $D = (40, 75)$. Consider D' = (35.71, 5.23). As shown in Table 5, we have $Z = X + D' = (5, 8)$.

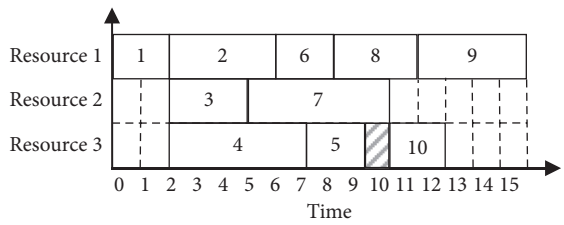

Figure 2: A feasible schedule.

Table 2: Resource-task assignment.

Task	1	2	3	4	5	6	7	8	9	10
Resource	1	1	2	3	3	1	2	1	1	3

Table 3: Resource order in measurement model.

Order	0	1	2	3	4	5
Resource	R_1	R_3	R_4	R_5	R_9	R_{10}
Resource	R_3	R_5	R_7	R_8	R_{10}	

Table 4: Schedule's task-resource assignment.

Schedule	Task	
	1	2
X	R_3	R_5
Y	R_5	R_{10}

Table 5: Measurement value.

Task	1	2
X	R_3	R_5
D'	35.71	5.23
Pi. order (xi)	20	60
$X + D'$	55.71	65.23
$Z = X + D'$	R_5	R_8

4.3. Proposed DEM Algorithm. The proposed algorithm (differential evolution multiskill, or DEM) is described in Algorithm 1:

(i) Stop criterion: the loop *while-end while* would be stopped if the difference between *makespan* of some continuous generations is smaller than a threshold

(ii) Size: number of solutions in the population

(iii) F: parameter of the mutation operator

(iv) CR: crossover probability

(v) n: number of tasks

As shown in line 25, after classical DE steps such as mutation and selection are executed, function *Reallocate*() is called to reduce the makespan of the candidate schedule *bestnest*. Procedures of this function will be dealt with in more detail in the next section (Algorithm 2):

currentBest: the best schedule among the population of the current generation.

R_b: According to the arrangement of schedule newbest (line 3), R_b is the last resource that finishes its job. In Figure 3, before the reallocate function was called, R_b is resource 1.

maxResource(): the function which returns the last resource to finish (line 3).

size(): the function which returns the number of elements of the given set or array (line 5).

newmakespan: makespan of the new schedule (line 11), which is obtained from the old schedule by reassigning task i. The task i is moved from resource R_b (line 9) to resource $R_i[j]$ (line 10).

4.4. Function Reallocate. Line 12 and 13 demonstrate that the new schedule (newbest) is better than the old schedule (currentBest) in terms of makespan, which means that the function *Reallocate* certainly returns a better schedule. Moreover, the function *Reallocate* guides DEM algorithm in the right direction by finding a new schedule (newbest) based on the best solution (currentBest); therefore, it inherits the advanced genomes of the population as well as the old schedule.

Example 3. Given set of task $T = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, set of resource RS = {1, 2, 3}. Resource 1 can execute tasks 1, 2, 3, 4, 6, 8, 9, and 10; resource 2 can perform tasks 1, 3, 7, and 9; resource 3 can handle tasks 1, 4, 5, 8, 9, and 10. The relationship between tasks is depicted in Figure 1, and the duration of the tasks is given in Table 1.

Consider a schedule (PS) as shown in Table 6. From observation, Figure 4 shows that makespan $f(\text{PS}) = 17$.

New schedule, the result of the function *Reallocate*, is described in Table 7.

Figure 4 illustrates that function *Reallocate* converts the schedule PS into the new schedule by assigning task 9 to resource 2 instead of resource 1 as before. Thanks to this allocation, the new schedule achieves better makespan (15) compared to the old schedule (17).

5. Experiment

Experiments were conducted on the simulated dataset and the real dataset to evaluate the performance of proposed algorithms in comparison with previous algorithms such as GreedyDO and GA. We first utilize the simulated iMOPSE dataset, a collection of project instances that are artificially created based on a database obtained from the international company, to evaluate the performance of the proposed algorithm.

5.1. Simulation on Simulated Dataset

5.1.1. iMOPSE Dataset. The simulated dataset iMOPSE [5] has been used to investigate existing algorithms GreedyDO and GA [13, 36]; therefore, we conduct simulations on iMOPSE dataset to fairly compare between DEM and those algorithms.

As shown in Table 8, iMOPSE's instances are classified into 15 categories according to the following attributes:

(1) Begin
(2) Load and validate iMOPSE dataset
(3) $t \longleftarrow 0$
(4) Size $\longleftarrow$ size of the population
(5) $P(t) \longleftarrow$ Initial population
(6) $f(t) \longleftarrow$ Calculate the fitness and makespan
(7) while (Stop criterion)
(8) for (int $i = 0$; $i <$ Size; i++)
(9) $x_{r1} \neq x_{r2} \neq x_{r3} \neq x_i \longleftarrow$ rand (1,Size)
(10) $F \longleftarrow$ rand (0, 1)
(11) $v_i(t) \longleftarrow x_{r1} + F \times (x_r - x_{r3})$// parameter of the mutation operator
(12) for ($j = 0$; $j < n$; j++)
(13) $u_{i,j} = \begin{cases} v_{i,j} \text{ if } \text{rand}_{i,j} \leq \text{CR or } i = I_{\text{rand}} \\ x_{i,j} \text{if } \text{rand}_{i,j} \geq \text{CR or } i \neq I_{\text{rand}} \end{cases}$
(14) end for
(15) if ($f(u_i(t)) \leq f(x_i(t))$)
(16) $x_i(t+1) = u_i(t)$
(17) else
(18) $x_i(t+1) = x_i(t)$
(19) end if
(20) end for
(21) Calculate the fitness and bestnest
(22) If (makespan > min (fitness))
(23) makespan = min (fitness)
(24) End if
(25) bestnest $\longleftarrow$ Reallocate (bestnest)
(26) $t \longleftarrow t + 1$
(27) end while
(28) return makespan
(29) End

Algorithm 1: DEM algorithm.

Input: currentBest //the best schedule among the current population
Output: //the improved schedule
(1) Begin
(2) makespan = f(best)
(3) newbest = currentBest;
(4) $R_b \longleftarrow$ maxResource (newbest) //the last resource to finish its job
(5) $Tb \longleftarrow$ set of tasks is performed by resource Rb
(6) For $i = 1$ to size(Tb) // Consider each task in T_b, the set of tasks performed by resource R_b
(7) $T_i = T_b\ [i]$;
(8) $R_i \longleftarrow$ set of resource that are skilled enough to execute the task i except R_b
(9) For $j = 1$ to size (R_i) // Consider each resource in turn
(10) $R_i[j] = R_i[j] + \{i\}$ // Reassign task i to resource $R_i[j]$
(11) $R_b = R_b - \{i\}$ // Remove task i from the task list of R_b
(12) newmakespan = f(newbest) // The makespan of the new schedule
(13) If newmakespan < makespan
(14) makespan = newmakespan
(15) Return newbest; // Successful Reallocate, return the new and better schedule
(16) End if
(17) newbest = currentBest; // Unsuccessful Reallocate, reuse the original schedule for the next iteration
(18) End for
(19) End for
(20) Return best
(21) End Function

Algorithm 2: Function reallocate.

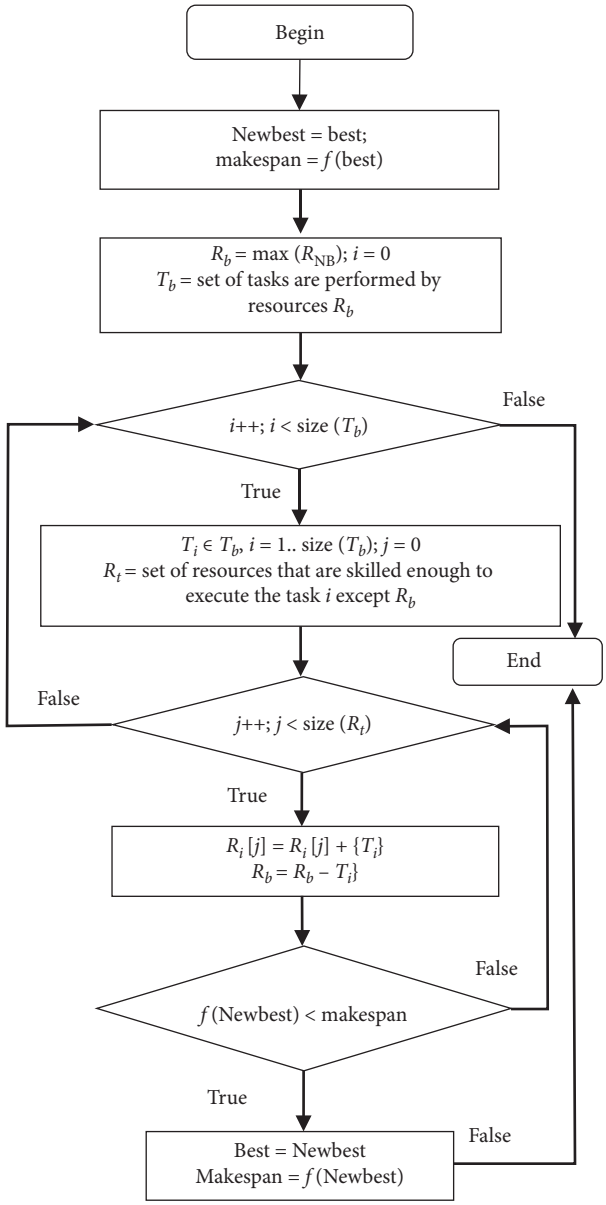

Figure 3: Function reallocate.

(i) The number of tasks and resources

(ii) The number of precedence relations between tasks

(iii) The number of skills

All simulations were run on a PC with Intel Core i5-CPU 2.2 GHz, 6 GB RAM, and operating system Windows 10. All of iMOPSE instances used by our simulations are listed in Table 8.

5.1.2. Parameters and System Settings. The simulation is conducted by using the following:

(i) Simulation environment: Matlab ver. 2014

(ii) Simulation tool: the performance of previous algorithms such as GA is fairly evaluated by using GARunner [13]; the simulation tool was provided by the author of those algorithms

Table 6: Resource-task assignment of PS.

Task	1	2	3	4	5	6	7	8	9	10
Resource	1	1	2	3	3	1	2	1	3	3

(iii) Input data: 15 iMOPSE instances that are described in Table 8

(iv) Number of solution in the population N_p: 100

(v) Number of generations N_g: 50,000

(vi) Since all of the considered algorithms are approximate, we run on each instance 35 times to get average value, the standard deviation value, and the best value

5.1.3. Simulation Results. Simulation results are shown in Tables 9 and 10, where the makespan of the best solutions found by GreedyDO, GA, and the proposed algorithm (DEM) is illustrated (in hours). Table 9 shows that the makespans acquired by GA are always better than those acquired by GreedyDO.

Table 10 illustrates the average value, the best value, and the standard deviation of the makespan found by GA and the proposed DEM algorithm.

From the observation, the result in Tables 9 and 10 shows that

(i) The solutions found by DEM are better than Greedy from 16% to 78% and GA from 0% to 21%.

(ii) Regarding the average value (Table 9): DEM is better than Greedy from 15% to 77.5% and GA from 0.7% to 20.8%.

(iii) Inspired by the differential evolution strategy, the proposed algorithm uses two parameters to guide the generation of the solution generated in the next generation: (i) the experience vector of both the population and (ii) the experience vector of the solution across different generations. DEM not only inherits those advantages of differential evolution approach but also utilizes the function reallocate, which improves its performance. Therefore, DEM has faster convergence and greater stability, which is shown by the standard deviation in Table 9 (column Std). The total standard deviation of DEM is 19.5 while the value of GA is 55.7; this result proves that the stability of DEM algorithm is better than GA.

(iv) In addition, the use of a scale to represent solutions helps DEM calculate the deviations between solutions more accurately and create better solutions in the next generations.

5.2. Simulation on Real Dataset

5.2.1. TNG Dataset. Experiments conducted on the simulated dataset are sometimes inaccurate and unsatisfactory. To overcome this drawback, we collected the dataset of TNG Investment and Trading Joint Stock Company [37], a well-

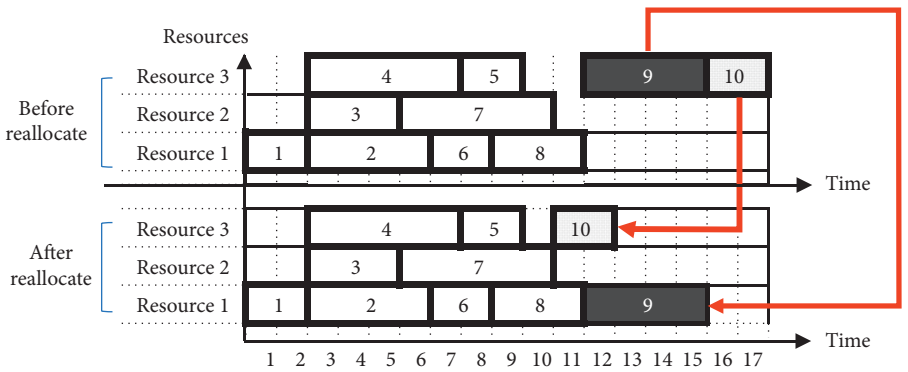

Figure 4: Gantt diagram of schedule before and after function *Reallocate*.

Table 7: Resource-task assignment of new schedule.

Task	1	2	3	4	5	6	7	8	9	10
Resource	1	1	2	3	3	1	2	1	1	3

Table 8: MS-RCPSP d36 iMOPSE dataset.

Dataset instance	Tasks	Resources	Precedence relations	Skills
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_64_9	100	10	64	9
100_10_65_15	100	10	65	15
100_20_65_15	100	20	65	15
100_20_65_9	100	20	65	9
200_10_84_9	200	10	84	9
200_10_85_15	200	10	85	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_45_9	200	40	45	9
200_40_90_9	200	40	90	9
200_40_91_15	200	40	91	15

known national textile company. TNG dataset instances have the following characteristics and parameters:

(i) The company has product sewing contracts under the business partners' contracts; each order places a sample of a product in large quantities

(ii) Each product has a certain number of stages, each of which requires a certain time to complete

(iii) Each order will be assigned to a group of workers

(iv) Each group consists of many workers

(v) Each worker's skills are rated based on his rank

The above parameters of the dataset have been converted to a suitable format as shown in Table 11.

Table 11 demonstrates that the TNG dataset parameters will be represented as follows:

Table 9: The makespan of the best solutions found by GreedyDO and GA.

Dataset instance	GreedyDO	GA
100_5_48_9	779	528
100_5_64_15	640	527
100_5_64_9	597	508
100_10_64_9	533	296
100_10_65_15	426	286
100_20_65_15	310	240
100_20_65_9	408	181
200_10_84_9	999	567
200_10_85_15	706	549
200_20_55_9	999	312
200_20_97_15	680	424
200_20_97_9	816	321
200_40_45_9	821	209
200_40_90_9	963	211
200_40_91_15	519	200

(i) The order from the business partners' is represented by a project

(ii) The stage of the product is represented by a task

(iii) A worker in the factory is represented by a resource in the problem formulation

(iv) The worker grade is represented by the skill level of the resource

(v) The order in which the tasks are executed is represented by the priority of the tasks

(vi) Order execution time is represented by the total time of project implementation, i.e., the makespan

After the conversion, the dataset is shown in Table 12

5.2.2. Parameters and System Settings. Experiment setting:

(i) Input data: 8 instances that are listed in Table 12

(ii) Number of the solution in the population N_p: 100

(iii) Number of generations N_g: 50,000

Table 10: The average value, the best value, and the standard deviation of makespan obtained from GA and DEM.

Dataset instance	GA			DEM		
	Avg	Best	Std	Avg	Best	Std
100_5_48_9	535	528	9.7	498	492	0.50
100_5_64_15	530	527	2.5	510	504	1.50
100_5_64_9	521	508	9.9	488	485	1.00
100_10_64_9	305	296	6.6	264	257	1.50
100_10_65_15	290	286	5.0	263	258	0.50
100_20_65_15	240	240	0.0	211	210	0.50
100_20_65_9	187	181	4.5	173	158	1.00
200_10_84_9	583	567	11.4	537	527	3.00
200_10_85_15	555	549	4.9	492	490	0.50
200_20_55_9	318	312	4.2	310	290	0.50
200_20_97_15	438	424	9.7	336	330	0.00
200_20_97_9	326	321	6.2	314	306	3.00
200_40_45_9	213	209	2.9	217	207	1.50
200_40_90_9	215	211	3.1	217	206	1.00
200_40_91_15	205	200	3.4	205	191	3.50

Table 11: Parameters conversion.

Real factor or parameter	The role
Order	Project
Stage of the product	Task
Time to execute the stage	Duration of the given task
Worker	Resource
The worker grade	Skill level of the resource
The order in which the tasks are executed	The priority of the tasks
Order execution time	The total time of project implementation, i.e., the makespan

Table 12: TNG dataset.

Dataset instance	Number of tasks	Number of resources	Precedence relations	Skill levels	Project time
TNG1	71	37	1026	6	409
TNG2	71	39	1026	6	325
TNG3	71	41	1026	6	296
TNG4	71	45	1026	6	392
TNG5	137	37	1894	6	1174
TNG6	137	39	1894	6	1052
TNG7	137	41	1894	6	871
TNG8	137	45	1894	6	996

(iv) Like simulation on the simulated dataset, we run on each TNG dataset instance 35 times to get the best value

5.2.3. Simulation Results. To prove the effectiveness of the proposed algorithm (DEM), we carry out experiments by using the real-world dataset collected from the TNG company. DEM will be compared with other algorithms from previous research studies, GreedyDO and GA. The performance of these algorithms is recorded and observed by using GARunner, the tool created by Myszkowski et al. [13].

Table 13 contains the makespan (measured in hours) of the best solution achieved by GreedyDO, GA, and DEM. In addition, the performing intervals of the actual projects at the TNG factory are shown in column TNG.

Table 13 proves that schedules found by DEM are better than those acquired by GA and GreedyDO. Table 13 and Figure 5 also highlight that the schedules found by evolutionary algorithms are always shorter than the execution time of the actual projects at the TNG factory.

Compared to the execution time at the TNG factory, the GreedyDO and GA algorithms reduce the makespan by 4%–42% and 7%–51%, respectively, while DEM has the best

Table 13: Comparison between the makespan of real TNG project and makespan of schedule found by GreedyDO, GA, and DEM.

Dataset instance	TNG	GreedyDO	GA	DEM
TNG1	409	236	201	165
TNG2	325	243	198	166
TNG3	296	258	212	166
TNG4	392	248	176	165
TNG5	1174	972	751	712
TNG6	1052	963	791	712
TNG7	871	834	810	728
TNG8	996	906	720	675

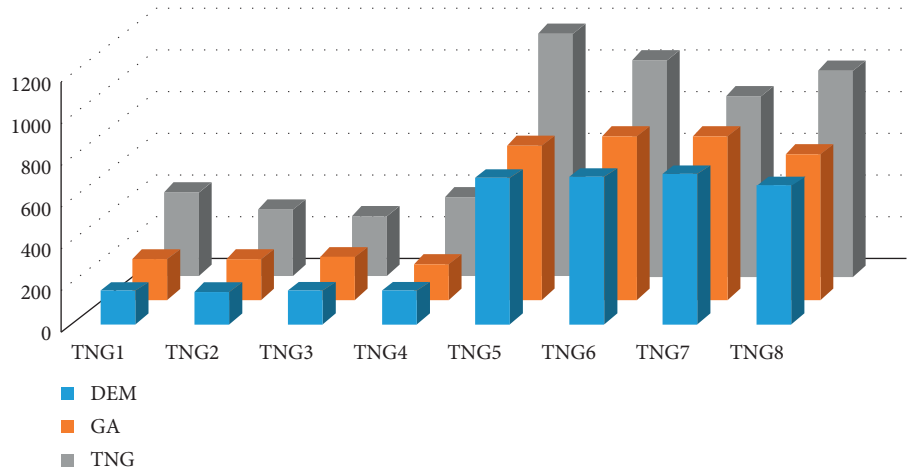

Figure 5: The comparison of makespan between DEM, GA, and TNG's solution.

results of 16%–61%. This result is understandable because so far all projects have been manually scheduled based on the scheduler's experience.

This statistical analysis of the experimental result proves that DEM achieved better solutions than the previous algorithms. Thanks to the directed nature, DEM not only ensures fast convergence but also averts getting trapped in local extrema. In experiments conducted on the TNG dataset, DEM reduces the makespan from 16% to 61% compared to the current factory schedule.

6. Conclusion and Future Work

This paper proposes to investigate the MS-RCPSP, one of the combinatorial optimization problems that have a wide range of practical applications in science, technology, and life. The MS-RCPSP is formulated and presented as a mathematical model, and then a novel algorithm called DEM which is based on differential evolution strategy is proposed.

The MS-RCPSP, which is the subject of the investigation in this paper, has significant meaning for network resource management in the Internet of things. The MS-RCPSP we study takes into account the resource's skill level, an important factor of the practical projects. Inspired by differential evolution metaheuristic, we have devised a new evolutionary algorithm that has the ability to outperform all the previous approaches. We build a function called reallocate, which is the key component that creates the strength of the proposed algorithm. This function avoids the idle time intervals in a given schedule, improving the solution quality.

To analyze the efficiency of the proposed algorithm, various simulations are conducted and compared with the previous algorithms. Besides the simulated dataset, the real dataset supported by the TNG company has also been utilized for the assessment. The simulation result shows that our developed algorithm is more effective than existing algorithms. DEM achieves not only better quality solution but also faster convergence to global extremum in comparison with the prior algorithms.

In the next steps, we are planning to improve DEM algorithm by using a blended random walk and the Gauss statistical function. We will implement local search operations to further improve the quality of the solutions. At the same time, we will integrate the Gaussian probability distribution to improve the search direction modification in the solution space. Thanks to that improvement, we expect that the proposed algorithm will not fall into the local extreme trap.

In the future, we will utilize the DE metaheuristic with the integration of the Gauss distribution and Gauss statistical function [38, 39] to improve the effectiveness of the searching for the optimum solution and avoid the local extreme trap.

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, we also tested the algorithm with TNG's garment manufacturing dataset. We have obtained permission from TNG to use these data.

Conflicts of Interest

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

Acknowledgments

We are grateful to the TNG Investment and Trading Joint Stock Company for giving us the information regarding their projects as well as providing us with their factory dataset.

References

[1] Z. Ning, G. Xu, N. Xiong et al., "TAW: cost-effective threshold authentication with weights for Internet of Things," *IEEE Access*, vol. 7, pp. 30112–30125, 2019.

[2] M. Alirezaei, S. T. A. Niaki, and S. A. A. Niaki, "A bi-objective hybrid optimization algorithm to reduce noise and data dimension in diabetes diagnosis using support vector machines," *Expert Systems with Applications*, vol. 127, pp. 47–57, 2019.

[3] O. P. Mejia, "A new RCPSP variant to schedule research activities in a nuclear laboratory," in *Proceedings of the 47th International Conference on Computers and Industrial Engineering (CIE47)*, Lisbon, Portugal, October 2017.

[4] K. Price, R. Storn, and J. Lampinen, *Differential Evolution-A Practical Approach to Global Optimization*, Springer, Berlin, Germany, 2005.

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

[6] Y. Zeng, N. Xiong, and T. H Kim, "Channel assignment and scheduling in multichannel wireless sensor networks," in *Proceedings of the 2008 33rd IEEE Conference On Local Computer Networks (LCN)*, pp. 512-513, Montreal, Canada, October 2008.

[7] H. Cheng, N. Xiong, and L. T. Yang, "Distributed access scheduling algorithms in wireless mesh networks," in *Proceedings of the 22nd International Conference On Advanced Information Networking And Applications (Aina 2008)*, pp. 509–516, Okinawa, Japan, November 2008.

[8] G. Wei, A. V. Vasilakos, and N. Xiong, "Scheduling parallel cloud computing services: an evolutional game," in *Proceedings of the 2009 First International Conference on Information Science and Engineering*, pp. 376–379, Nanjing, China, December 2009.

[9] W. Nie, N. Xiong, J. H. Park, and S. Yeo, "A fair-oriented two-level scheduling scheme for downlink traffic in wimax network," in *Proceedings of the 2010 2nd International Conference on Information Technology Convergence and Services*, pp. 1–6, Cebu, Phillippines, January 2010.

[10] W. Zheng, N. Xiong, N. Ghani, M. Peng, A. V. Vasilakos, and L. Zhou, "Adaptive scheduling for wireless video transmission in high-speed networks," in *Proceedings of the 2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 180–185, Shanghai, China, January 2011.

[11] P. Dong, J. Xie, W. Tang, N. Xiong, H. Zhong, and A. V. Vasilakos, "Performance evaluation of multipath TCP scheduling algorithms," *IEEE Access*, vol. 7, pp. 29818–29825, 2019.

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

[13] P. B. Myszkowski, M. Laszczyk, I. Nikulin, and E. Skowronski, ""iMOPSE: a library for bicriteria optimization in multi–skill resource–constrained project scheduling problem"" *Soft Computing*, vol. 23, no. 10, pp. 3397–3410, 2019.

[14] H. Cheng, N. Xiong, X. Huang, and L. T. Yang, "An efficient scheduling model for broadcasting in wireless sensor networks," in *Proceedings of the 2013 IEEE International Symposium on Parallel & Distributed Processing*, pp. 1417–1428, IEEE, Cambridge, MA, USA, May 2013.

[15] B. Lin, W. Guo, N. Xiong, G. Chen, A. V. Vasilakos, and H. Zhang, "A pretreatment workflow scheduling approach for big data applications in multicloud environments," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 581–594, 2016.

[16] B. Lin, W. Guo, G. Chen, N. Xiong, and R. Li, "Cost-driven scheduling for deadline-constrained workflow on multi-clouds," in *Proceedings of the 2015 IEEE International Parallel and Distributed Processing Symposium Workshop*, pp. 1191–1198, IEEE, Hyderabad, India, May 2015.

[17] L. Tan, Z. Zhu, F. Ge, and N. Xiong, "Utility maximization resource allocation in wireless networks: methods and algorithms," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 7, pp. 1018–1034, 2015.

[18] P. Guo, W. 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.

[19] S. Kavitha and P. Venkumar, "A vibrant crossbreed social spider optimization with genetic algorithm tactic for flexible job shop scheduling problem," *Measurement and Control*, vol. 53, no. 1-2, 2020.

[20] A. P. Agrawal, A. Choudhary, and A. Kaur, "An effective regression test case selection using hybrid whale optimization algorithm," *International Journal of Distributed Systems and Technologies*, vol. 11, no. 1, pp. 53–67, 2020.

[21] W. Guo, J. H. Park, L. T. Yang, A. V. Vasilakos, N. Xiong, and G. Chen, "Design and analysis of a MST-based topology control scheme with PSO for wireless sensor networks," in *Proceedings of the 2011 IEEE Asia-Pacific Services Computing Conference*, pp. 360–367, IEEE, Jeju, Korea, November 2011.

[22] X. Zhuang, H. Cheng, N. Xiong, and L. T. Yang, "Channel assignment in multi-radio wireless networks based on pso algorithm," in *Proceedings of the 2010 5th International Conference on Future Information Technology*, pp. 1–6, Busan, Korea, November 2010.

[23] C. T. Su, M. C. Santoro, and A. Mendes, "Constructive heuristics for project scheduling resource availability cost problem with tardiness," *Journal of Construction Engineering and Management*, vol. 144, Article ID 4018074, p. 8, 2018.

[24] H. Maghsoudlou, B. Afshar-Nadjafi, S. T. Akhavan Niaki, and S. 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.

[25] 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.

[26] J. Lin, L. Zhu, and K. Gao, "A genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem," *Expert Systems with Applications*, vol. 140, p. 112915, 2020.

[27] A. H. Hosseinian and V. Baradaran, "P-GWO and MOFA: two new algorithms for the MSRCPSP with the deterioration effect and financial constraints (case study of a gas treating company)," *Applied Intelligence*, vol. 50, pp. 2151–2176, 2020.

[28] A. H. Hosseinian, V. Baradaran, and M. Bashiri, "Modeling of the time-dependent multi-skilled RCPSP considering learning effect," *Journal of Modelling in Management*, vol. 10, 2019.

[29] R. Nemati-Lafmejani, H. Davari-Ardakani, and H. Najafzad, "Multi-mode resource constrained project scheduling and contractor selection: mathematical formulation and metaheuristic algorithms," *Applied Soft Computing*, vol. 81, Article ID 105533, 2019.

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

[31] F. Ballestín and R. Leus, "Resource-Constrained project scheduling for timely project completion with stochastic activity durations," *Production and Operations Management*, vol. 18, no. 4, pp. 459–474, 2009.

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

[33] V. Feoktistov, *Differential Evolution*, Springer, New York, NY, USA, 2006.

[34] S. Tanabe, F. Ryoji, and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *Proceedings of the 2013 IEEE congress on evolutionary computation*, IEEE, Cancun, Mexico, November 2013.

[35] T. K. Ghosh and S. Das, "A novel hybrid algorithm based on firefly algorithm and differential evolution for job scheduling in computational grid," *International Journal of Distributed Systems and Technologies*, vol. 9, no. 2, pp. 1–15, 2018.

[36] 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.

[37] http://www.tng.vn TNG Investment and Trading Joint Stock Company, 434/1 bac kan street, Thai nguyen city, Viet Nam,.

[38] L. Kang, R.-S. Chen, N. Xiong, Y. C. Chen, Y.-X. Hu, and C.-M. Chen, "Selecting hyper-parameters of Gaussian process regression based on non-inertial particle Swarm optimization in internet of things," *IEEE Access*, vol. 7, pp. 59504–59513, 2019.

[39] M. Wu, N. Xiong, and L. Tan, "Adaptive range-based target localization using diffusion gauss–newton method in industrial environments," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 11, pp. 5919–5930, 2019.