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

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-RCPS 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-RCPS 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]. Hosseini and Baradaran [9] used an evaluation method to make plan decisions with MS-RCPS. Nazafzad et al. [10] employed a biojective optimization model for the MS-RCPS 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-RCPS occurring in grid computing.

However, the MS-RCPS 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-RCPS, called the Real-RCPS. In other words, Real-RCPS is a specific case of the MS-RCPS. In the Real-RCPS, the processing time depends on the skill level of the resource. The real-life nature of Real-RCPS 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-RCPS. 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 Realocate, 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-RCPS, no one to the best of our knowledge has studied this problem. This paper is the first work that mentions Real-RCPS; thus this section introduces the existing algorithms to solve another problem that is close to the Real-RCPS, namely, MS-RCPS. In Section 5 and Section 6, these algorithms will be used as reference algorithms in our experiments.

Myszkowski et al. [6] proved that MS-RCPS 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-RCPS; 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 multi-risk 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-RCPS problem, Real-RCPS is researched and applied in many fields of the Internet of Things. Hosseini 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-Mejaia 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 DPSOCA, 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-RCPS 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 \(r_i\) to perform task \(j\)
(iii) \(S\): the set of all skills; \(S_i\): the set of skills belonging to resource \(i\)
(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; \(L_i\): the set of resources that could handle task \(i\)
(vi) \(L_i\): resource \(i\)
(vii) \(W\): the set of tasks; \(W_k\): the set of tasks that could be performed by resource \(k\)
(viii) \(W_k\): task \(k\)
(ix) \(B_k, E_k\): starting time and ending time of task \(k\)
(x) \(A'_{ui}\): 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; \(P_m\): the set of candidate schedules
(xv) \(f(P)\): makespan (execution time) of schedule \(P\)
(xvi) \(\gamma\): number of tasks; \(\zeta\): number of resources

Real-RCPSP could be defined as follows:

\[
\text{minimize } f(P),
\]

where

\[
f(P) = \max_{i \in T} [E_i] - \min_{k \in T} [B_k],
\]

subject to

\[
S_k \neq \emptyset, \quad \forall k \in L,
\]

\[
T_j \geq 0, \quad \forall j \in W,
\]

\[
E_j \geq 0, \quad \forall j \in W,
\]

\[
E_i \leq E_j - t_{ij}, \quad \forall j \in W, j \neq i, i \in C_j,
\]

\[
\forall i \in W_k \exists r \in S^k: g_r = g_{r_i}, h_r \geq h_{r_i},
\]

\[
\forall k \in L, \forall t \in m: \sum_{i=1}^{n} A'_{ik} \leq 1,
\]

\[
\forall j \in W \forall t \in m, \forall k \in L: A'_{jk} = 1,
\]

If \(h_i < h_j\) then \(t_{jk} > t_{ik} \quad \forall (r_i, r_j) \in [S_k \times S_r]\).}

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) \( S_r = S, \forall k \in L \); 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

As described in Table 2, resource 1 executes task 2 and

As described in Table 2, resource 1 executes task 2 and

As described in Table 2, resource 1 executes task 2 and

As described in Table 2, resource 1 executes task 2 and

As described in Table 2, resource 1 executes task 2 and

4.2. Measurement Model. Cuckoo Search algorithm is an optimization scheme dealing with real functions such as the Gaussian probability distribution function, whereas RealRCPSP is the optimization problem of discrete functions. Therefore, in order to apply the Cuckoo Search algorithm to the RealRCPSP, 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) \( p_i = \text{position} \left( \text{round} \left( q_i + k_i \right) \right) \) and position (i) presents the respective resource
(iv) \( k_i = v_i \times \left( \text{order} \left( p_i \right) - \text{order} \left( q_i \right) \right) \)

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 \( z_1 = 5; v_1 = 100/(5 - 1) = 25 \).

4.4. Function Reallocate. \( c_{\text{plan}} \) is the most appropriate feasible schedule until now. \( L_6 \) is the last resource to finish. Function Late() will find out the value of \( R_{\text{makespan}} \). Size() is the size of a set or an array. \( N_{\text{makespan}} \) is execution time of the new resource-task arrangement.

Table 1: Execution times of the tasks.

| Task | \( W_1 \) | \( W_2 \) | \( W_3 \) | \( W_4 \) | \( W_5 \) | \( W_6 \) | \( W_7 \) | \( W_8 \) | \( W_9 \) | \( W_{10} \) |
|------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| Time | 3       | 2       | 3       | 4       | 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 | \( L_2 \) | \( L_1 \) | \( L_3 \) | \( L_2 \) | \( L_1 \) | \( L_3 \) | \( L_3 \) | \( L_3 \) | \( L_3 \) | \( L_3 \) |

Table 3: The order of resources.

| Order | 0 | 1 | 2 | 3 | 4 |
|-------|---|---|---|---|---|
| Resource | \( L_1 \) | \( L_3 \) | \( L_4 \) | \( L_9 \) | \( L_{10} \) |
| Resource | \( L_6 \) | \( L_2 \) | \( L_8 \) | \( L_{10} \) |
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).

Algorithm 1: R-CSM algorithm.

Table 4: The assignment of schedules P and Q.

| Schedule | Task |
|----------|------|
| P        | L₁   |
| Q        | L₄   |

Table 5: Measurement value.

| Task     | 1    | 2    |
|----------|------|------|
| P        | L₁   | L₈   |
| Kₚ       | 0    | 66.66 |
| K        | 50   | 33.33 |
| Kₚ + K   | 50   | 99.99 |
| Z = Kₚ + K | L₆   | L₁₀  |

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:
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 \( \left( N_g = 50,000 \right) \)
(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.
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

### 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 | $L_1$ | $L_1$ | $L_2$ | $L_3$ | $L_3$ | $L_1$ | $L_2$ | $L_1$ | $L_3$ | $L_3$ |

### 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 | $L_1$ | $L_1$ | $L_2$ | $L_3$ | $L_3$ | $L_1$ | $L_2$ | $L_1$ | $L_1$ | $L_3$ |
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
(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%.
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 combinational 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 *Proceedings of 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, New York, NY, USA, 2000.

[6] Myszkowski, B. Pawel, E. Marek, Skowroński, and S. Krysztof, “A new benchmark dataset for multi-skil 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. Blazewicz, 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. Hosseinion 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-Arakani, and R. Nemati-Lamejani, “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 multi-verse optimization algorithm for multi-skill resource constrained project scheduling problem,” Applied Soft Computing, vol. 85, Article ID 105805, 2019.

[18] A. H. Hosseinion 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-Mejia, 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. Ballestin, 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-skill 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 (IColICT), Bandung, Indonesia, March 2013.

[28] P. B. Myszkowski, L. 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.