HNUE JOURNAL OF SCIENCE DOI: 10.18173/2354-1059.2020-0034
Natural Sciences 2020, Volume 65, Issue 6, pp. 98-109
This paper is available online at http://stdb.hnue.edu.vn

A NEW ALGORITHM FOR MULTI-SKILL RESOURCE CONSTRAINED PROJECT SCHEDULING PROBLEM BASED ON CUCKOO SEARCH STRATEGY

Dang Quoc Huu[1], Nguyen The Loc[2], Nguyen Doan Cuong[3]
and Phan Thanh Toan[4]

[1]*Center of Information Technology, Thuong Mai University*
[2]*Faculty of Information Technology, Hanoi National University of Education*
[3]*Institute of Information Technology, Military Institute of Science and Technology*
[4]*Faculty of Technology Education, Hanoi National University of Education*

Abstract. The purpose of this paper is to consider the project scheduling problem under such limited constraint, called Multi-Skill Resource-Constrained Project Scheduling Problem or MS-RCPSP. The algorithm proposed in this paper is to find the optimal schedule, determine the start time for each task so that the execution time (also called makespan) taken is minimal. At the same time, our scheduling algorithm ensures that the given priority relationships and constraints are not violated. Our scheduling algorithm is built based on the Cuckoo Search strategy. In order to evaluate the proposed algorithm, experiments were conducted by using the iMOPSE dataset. The experimental results proved that the proposed algorithm found better solutions than the previous algorithm.

Keywords: optimization and swarm intelligence, evolutionary algorithm, resource-constrained project scheduling problem, cuckoo search algorithm, optimization algorithm.

1. Introduction

The Multi-Skill Resource-Constrained Project Scheduling Problems is a classical problem challenging combinatorial optimization problems that have increasingly attracted the attention of the scientists in the recent years, it is essentially a mapping of tasks to the resource that satisfy the order of the tasks and the makespan/cost is minimum. In MS-RCPSP many constraints related to resources and tasks have to be satisfied, there are two types of constraints: precedence between the tasks and resource constraints, given task

Received June 12, 2020. Revised June 23, 2020. Accepted June 29, 2020.
Contact Phan Thanh Toan, e-mail address: pttoan@hnue.edu.vn

can not start before its predecessors would be finished or given resource can not have more than one resource assigned in overlapping periods of time. In MS-RCPSP, each resource has a set of skills and each task requires given skill in specified familiarity level, so that not every resource can perform every task and the schedule is more difficult to be built.

The Multi-Skill Project Scheduling Problem (MS-RCPSP) has great importance for scheduling tasks in very specific fields, where many companies look for ways of optimizing their schedules. The companies require the presence of a group of technicians having a set of well-defined competencies for the execution of a task. This problem shows to be more challenging than traditional scheduling problems such as the Resource-Constrained Project Scheduling Problem. In this not only is it necessary to decide the specific resources that will be assigned to each task but also the skills with which they will contribute.

In the general case, the Multi-Skill Resource-Constrained Project Scheduling Problem (MS-RCPSP) is an NP-complete problem [1].

2. Content

2.1. Related works

Project scheduling is a big issue in many fields. Basically, it is the issue related to the mapping of each task to an appropriate resource and allowing the task to satisfy some performance constraints. The mapping of tasks to the resources and satisfy some given constrains is an NP-complete problem (According to the computational complexity theory [GAR 79]) [1]. So, past works have proposed many heuristics-based approaches to find the schedule of this problem.

Z.Chen and C.Chyu [2] proposed a scheduling algorithm for Resource-Constrained Project Scheduling Problem based on Evolutionary Algorithm, in the paper authors combine Evolutionary Algorithm and Local Search method to minimize the execution time of schedule, the solution is represented as a vector of length equal to the number of tasks. The value corresponding to each position *i* in the vector represents the resource to which task *i* was executed. P. Das [3] proposed a scheduling algorithm for RCPSP based on Simulated Annealing algorithm.

D.Huafenget et al. [4] proposed a general variable neighborhood search-based memetic algorithm for solving the multi-skill resource-constrained project scheduling problem under linear deterioration, in the paper authors integrate a solution recombination operator and a local optimization procedure, the proposed algorithm is assessed on two sets of instances and achieves highly competitive results.

Mosheiov [5] proposed a scheduling algorithm for RCPSP with processing times increase at a common rate and task weights as proportional to their normal processing time. The author has demonstrated that the optimal schedule is obtained if the optimal objective is to minimize the total weighted completion time on a single machine. Ji and Cheng [6] proposed a new method for parallel-machine scheduling and gave a fully polynomial-time approximation scheme. P. Guo, W. Cheng, and Y. Wang [7] proposed a

new neighborhood for the local search method to solve the single-machine total tardiness scheduling problem. In this paper the authors proposed a heuristic named simple weighted search procedure (SWSP) and a general variable neighborhood search algorithm (GVNS) to obtain near optimal solutions and authors used randomly generated test instances to evaluate the performance of the proposed algorithm, it is shown that the proposed algorithm can provide better solutions. The authors in articles [7-16] proposed scheduling algorithms for Resource-Constrained Project Scheduling Problems based on the Evolution Algorithm.

2.2. Problem Statement

**** Classical RCPSP***

MS-RCPSP is an extension for RCPSP [17], hence the description of RCPSP would be presented first as follows.

- A project is represented as a directed, acyclic graph G(V, E) where the nodes in the graph correspond to the task and the arcs in the graph specify precedence relationships between the tasks. Arc (u, v) appears in the graph mean task u must be completed before performing task *v* (Figure 1).

- Two dummy tasks could be added, the first one represents the start processing of the project and is a predecessor of every other task, while the final one denotes the end of the project's processing and is the successor of every other task.

- Each task can be defined by its duration, start, and finish dates.

- After started, any task cannot be stopped or delayed.

- Each task requires a constant number of units of a renewable resource.

The objective of RCPSP is to complete the project as soon as possible without violating any constraints, in other words, its goal is to find the schedule such that the makespan can be minimized while satisfying given precedence constraint between the activities and resource constraint.

**** MS-RCPSP Problem Formulations***

As a practical extension of RCPSP, MS-RCPSP [18, 19] adds the skills domain of the resources and aims to minimize the makespan. At a time only one resource can be assigned to a given task, in other words, any resource can not handle more than one task concurrently.

Each task requires a subset of skills while each resource owns another unique subset of skills, therefore not every resource could handle a given task.

MS-RCPSP can be conceptually formulated based on the following notations:

Input:

- d_j: non pre-emptive duration of task j; P_j : set of predecessors of task j
- q_j: set of skills required by task j to be performed;
- s_k: salary of resource k (hourly rate)
- Q_k : set of skills owned by resource k
- Q: set of skills, $Q_k \subseteq Q$; K: set of all resources

- J: set of all tasks; J_k: set of tasks that resource k could handle, $J_k \subseteq J$
- K_j: set of resources that could handle task k, $K_j \subseteq K$
- S_j, F_j: start time and finish time of task j
- $U_{j,k}^t$: assignment if resource k is assigned to task j in time t.
- l_q: the level of skill q; h_q: type of the skill q;
- τ : duration of a project schedule
- PS (project schedule): feasible project schedule
- PS_{all}: set of all project schedule

The feasible project schedule (PS) consists of J = 1,.., n tasks and K = 1,..., m resources.

The cost of performing task j by resource k is $c_{kj} = d_j \times S_k$ For simplicity, we have modified the cost of the task's performance from ckj to cj, because only one resource can be assigned to a given task in the duration of the project. Hence, there is no need to distinguish various costs for the same task.

Output:

Formally, MS-RCPSP could be regarded as mimization problem as follows:

$$f(PS) \rightarrow \min \tag{1}$$

Constraint:

$$s_k \geq 0 \ \forall k \in K \tag{2}$$

$$Q_k \neq \emptyset \forall k \in K \tag{3}$$

$$d_j \geq 0 \ \forall j \in J \tag{4}$$

$$F_j \geq 0 \ \forall j \in J \tag{5}$$

$$F_i \leq F_j - d_j \ \ \forall j \in J, j \neq 1, i \in P_j \tag{6}$$

$$\forall i \in J^k \ \exists q \in Q^k : \ h_q = h_{q_i} \text{ and } l_q \geq l_{q_i} \tag{7}$$

$$\forall k \in K, \forall t \in \tau : \sum_{i=1}^{n} U_{i,k}^t \leq 1 \tag{8}$$

$$\forall j \in J \exists! \, t \in \tau, ! \, k \in K : U_{j,k}^t = 1; \text{where } U_{j,k}^t \in \{0; 1\} \tag{9}$$

2.3. Proposed algorithm

Cuckoo Search (CS), the metaheuristic was developed by Yang and Deb [20] based on the obligate brood parasitic behavior of cuckoo bird species together with the Lévy flight behavior of some birds and fruit flies. Cuckoo Search is a nature-inspired algorithm, based on the brood reproductive strategy of cuckoo birds to increase their population. However, in some cases CS is more effective than other nature-inspired algorithms. In fact, DE, SA and PSO are special cases of CS algorithm, hence it is not surprising why CS algorithm outperforms them [21]. CS algorithm outperformed DE algorithm [22] in terms of convergence speed to reach the optimum solution. In addition, CS algorithm was reported as being more computationally efficient than the PSO [23].

This study proposes a new Cuckoo Search algorithm which, like conventional the Cuckoo Search algorithm [20], implements a combination of the local random walk and the global random walk.

The global random walk is carried out by applying Levy flights [5] as

$$x_i^{t+1} = x_i^t + \frac{\alpha.u}{|v|^{1/\beta}} \quad (10)$$

where u and v are the random values drawn from Gauss distribution. Their mean is zero while their standard deviation is:

$$\sigma_u = \left(\frac{\Gamma(1+\beta)sin(\pi\beta/2)}{\Gamma[(1+\beta)/2]\beta 2^{(\beta-1)/2}}\right)^{1/\beta};\ \sigma_v = 1 \quad (11)$$

Besides, the direction of the global random walks is drawn from a uniform distribution.

At each generation, pa percent of worse solutions are replaced by the new solutions generated by using Levy flight method. In the proposed algorithm CSM, pa is a dynamic chosen parameter. At the beginning the approximate value of pa is 50, that value is reduced to 25 at the ending generations when the CSM approaches the extrema.

The local random walk can be presented as:

$$x_i^{t+1} = x_i^t + \alpha\left(x_j^t - x_k^t\right) \quad (12)$$

where x^t_i and x^{t+1}_i are the current and the new solution respectively and α is the step size factor. While the x^t_j and x^t_k are randomly selected solutions among the current population in the conventional Cuckoo Search algorithm, the proposed algorithm CSM assigns x^t_k the value of the best solution in the current population. This assignment makes CSM converge faster.

2.3.1. Solution Representation

Each solution (i.e. schedule) is represented by a vector of elements, the number of elements (i.e. the size of vector) equal to the number of tasks. The value of each element demonstrates the resource which executes the corresponding task.

Example:

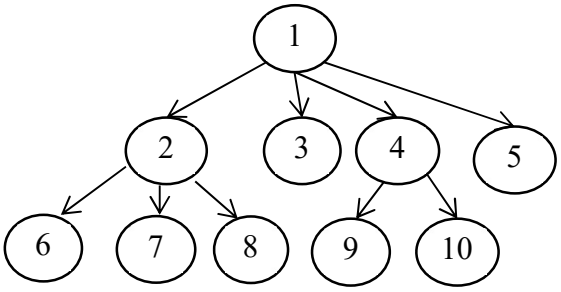

Figure 1. The relationship between tasks

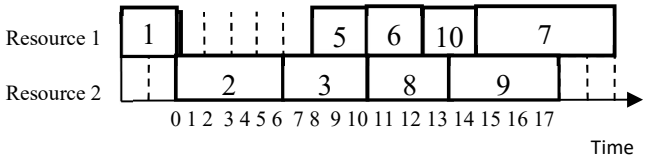

Figure 2. A feasible project schedule

Figure 1 demonstrates the relationship between tasks, based on that task 9 could not be started until task 4 finished, while task 6 and task 8 could be executed simultaneously. Figure 2 depicts a feasible project schedule,

where: - Set of task J={1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

- Set of resource K={1, 2}.

The duration of tasks can be represented as follows:

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

The schedule depicted in Fig. 2 can be represented as follows:

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

The above vector shows that the resource 1 is responsible for executing tasks 1, 4, 5, 6, 10 while resource 2 handles the rest one.

2.3.2. Measurement model

The original Cuckoo Search algorithm [20] is designed to deal with the floating-point value data, while MS-RCPSP's solution are the array of integer value elements. Thus, we build a new measurement model in order to measure the difference between two schedules (i.e. solutions) as follow:

- Unit vector $P = (p_1, p_2, ... p_n)$; $p_i = 100/(k_i\text{-}1)$ where k_i is the number of resources that could handle task T_i.

- The difference between two schedules X =(x_1, x_2, ... x_n) and Y=(y_1, y_2, ... y_n) is the differential vector D =(d_1, d_2, ... d_n);

$$D = X\text{-}Y$$

- The sum of the schedule Y and differential vector D is schedules X =(x1, x2, ... xn) where x_i = possition(round ($y_i + d_i$)); possition(i): the resource which corresponding to the position I; $d_i = p_i$. (order (x_i) - order(y_i)); order (x_i): the position of the resource (x_i) in the K_i

Example:

Assume that there are 6 resources could handle task T1: K1 = ={R1, R3, R4, R5, R9, R10}. Thus: k1 = 6; p1 = 100/(6-1) = 20.

Assume that there are 5 resources could handle task T2: K2 = {R3, R5, R7, R8, R10}. Thus: k2 = 5; p2 = 100/(5-1) = 25.

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}	

Consider 2 schedules: X = (3,5); Y = (5,10);

Schedule\Task	T_1	T_2
X	R_3	R_5
Y	R_5	R_{10}

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). We have Z = X+D'= (5,8)

Task	T_1	T_2
X	R_3	R_5
D'	35.71	5.23
p_i. order(x_i)	20	60
X+D'	55.71	65.23
Z= X+D'	R_5	R_8

2.3.3. The Functions

The operation of the proposed algorithm CSM as follows:

```
Function CSM()
Input:
Output:
  Begin
    LoadData();
    CheckDataValid();
    InitialPolulation();
    PopulationEvaluate();
   iInteration = 1;
    pa = 0.25;
    While (Stop criterion)
    Begin
          new_nest =
          CreateNewNest();
          nest_j =
          SelectRandomNest();
          if (f(new_nest)<f(nest_j)
          nest_j = new_nest;
          end
```

```
        RemoveWorstNests(pa)
        PopulationEvaluate();
      iInteration += 1;
      End
    End
```

Where:

- $f(i)$: makespan of schedule i.
- Stop criterion of CSM based on a threshold.
- p_a: a fraction of worse nests is replaced by the new one.

The objective function which returns the makespan of a given schedule could be described as follows:

```
Function f ()  // Objective function
Input:
Output: makespan of a given schedule
    Begin
      makespan = K[1,K[1].tasks.count].endtime
        for i=2 to resource_count
            if K[i,K[i].tasks.count].endtime >makespan then
                makespan = K[i,K[i].tasks.count].endtime
            end
        end
        return makespan;
    End
```

Where

- K: set of resources.
- *endtime*: the finish time of the given task.

2.4. Experiment and results

2.4.1. Experimental settings

To evaluate CSM and other algorithms, we use the benchmark iMOPSE dataset [24] which contains project instances artificially created based on real-world instances obtained from international enterprise. iMOPSE instances regarding the most common project characteristics such as number of tasks, number of resources, precedence relation, the number of the relationship between tasks, number of skills, set of skill belong to each resource.

We establish experiments over 10 iMOPSE datasets. Table 1 lists the above characteristics of them and the experimental results are given in Table II. All codes are

implemented in Matlab 2014, all experiments are run on the PC with the configuration of Intel Core i5 2.2 GHz, RAM 6 GB, Windows 7 Enterprise.

Table 1. MS-RCPSP d36 iMOPSE dataset

Dataset instance	Tasks	Resources	Precedence Relations	Skills
100_5_22_15	100	5	22	15
100_10_26_15	100	10	26	15
100_10_47_9	100	10	47	9
100_20_46_15	100	20	46	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_20_54_15	200	20	54	15
200_20_55_9	200	20	55	9
200_40_91_15	200	40	91	15

2.4.2. Results

The goal of conducted experiments is to investigate the robustness and efficiency of the proposed algorithm CSM in comparison with the previous algorithms such as GreedyDO [18], HAntCO [19] and GA [25]. The experimental results of previous algorithms taken by using GARunner tool [25] over iMOPSE dataset [24].

Table 2. Experiments results

Dataset	GreedyDO	HAntCO	GA	CSM
100_5_22_15	630	504	516	488
100_10_26_15	370	266	292	247
100_10_47_9	549	297	296	268
100_20_46_15	394	194	206	188
100_20_65_9	408	180	179	174
200_10_128_15	780	522	580	477
200_10_50_15	763	529	586	500
200_20_54_15	488	336	376	329
200_20_55_9	999	313	313	304
200_40_91_15	519	207	211	197

Figure 3 depicts the comparison between the proposed algorithm with HAntCO, the best algorithm among the predecessor one.

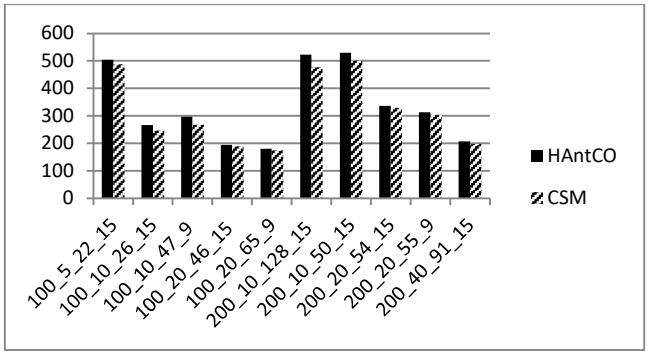

Figure 3 Experiment results of CSM and HAntCO

The experimental results showed that CSM is superior to its predecessor algorithms. Notably, the scheduling plans found by CSM are 34% - 62%, 7% - 18%, and 5% - 10% better than Greedy, GA, and HAntCO, respectively.

Based on the LévyFlight random walk, the proposed algorithm not only guarantee the fast convergence but also avoid being trapped on local extrema.

3. Conclusions

MS-RCPSP is a complex combinatorial optimization problem that has a broad scale of applications in life such as project arranging or finance organizing. In this paper, we stated the formal model of MS-RCPSP and developed a novel approach called CSM which based on Cuckoo Search algorithm. Although Cuckoo Search algorithm was very well considered only in the continuous-data problems, our results show that this algorithm could be more efficient than classical metaheuristic in the discrete-data problems such as MS-RCPSP. The experiment's results show that the proposed algorithm is superior to its predecessor. In the future, we are planning to improve this algorithm by using a blended random walk and Gaus statistical function.

REFERENCES

[1] M.R. Garey and D.S. Johnson, 1979. *Computers and Intractability. A guide to theory of NP-Completeness.* W.H. Freeman and Company, New York.

[2] Z.Chen, C.Chyu, 2010. *An Evolutionary Algorithm with Multi-Local Search for the Resource-Constrained Project Scheduling Problem.* Intelligent Information Management (2), pp. 220-226.

[3] P.P. Das, S. Acharyya, 2011. *Simulated Annealing Variants for Solving Resource Constrained Project Scheduling Problem:A Comparative Study.* Proceedings of 14th International Conference on Computer and Information Technology, pp. 469-474.

[4] D.Huafeng, Wenming Cheng, 2019. *A Memetic Algorithm for Multiskill Resource-Constrained Project Scheduling Problem under Linear Deterioration.* Mathematical Problems in Engineering, Article ID 9459375, 16 pages.

[5] G. Mosheiov, 1995. Scheduling jobs with step-deterioration; Minimizing makespan on a single- and multi-machine. *Computers & Industrial Engineering*, Vol. 28, No. 4, pp. 869-879.

[6] M. Ji and T. C. E. Cheng, 2008. Parallel-machine scheduling with simple linear deterioration to minimize total completion time. *European Journal of Operational Research,* Vol. 188, No. 2, pp. 342-347.

[7] P. Guo, W. Cheng, and Y. Wang, 2014. 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.

[8] P. B. Myszkowski, M. E. Skowroński, and K. Sikora, 2015. *A new benchmark dataset for Multi-Skill resource-constrained project scheduling problem.* Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS '15), pp. 129-138.

[9] P. B. Myszkowski, M. E. Skowroński, Ł. P. Olech, and K. Oślizło, 2015. Hybrid ant colony optimization in solving multi-skill resource-constrained project scheduling problem. *Soft Computing*, Vol. 19, No. 12, pp. 3599-3619.

[10] P. B. Myszkowski and J. J. Siemieński, 2016. GRASP applied to multi-skill resource–constrained project scheduling problem. *Computational Collective Intelligence,* pp. 402-411.

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

[12] P. B. Myszkowski, M. Laszczyk, and D. Kalinowski, 2017. *Co-evolutionary algorithm solving multi-skill resource-constrained project scheduling problem*. Proceedings of the Federated Conference on Computer Science and Information Systems, pp. 75-82.

[13] F. S. Alanzi, K. Alzame, and A. Allahverdi, 2010. Weighted multi-skill resources project scheduling. *Communications & Network*, Vol. 03, pp. 1125-1130.

[14] M. A. Santos and A. P. Tereso, 2011, On the multi-mode, multi-skill resource constrained project scheduling problem - a software application. *Advances in Intelligent and Soft Computing*, Vol. 96, pp. 239-248.

[15] H.-Y. Zheng, L. Wang, and X.-L. Zheng, 2017. Teaching-learning-based optimization algorithm for multi-skill resource constrained project scheduling problem. *Soft Computing*, Vol. 21, No. 6, pp. 1537-1548.

[16] H. Dai, W. Cheng, and P. Guo, 2018. An improved tabu search for multi-skill resource-constrained project scheduling problems under step-deterioration. *Arabian Journal for Science and Engineering*, Vol. 43, No. 6, pp. 3279-3290.

[17] R. Klein, Scheduling of Resource, 2000. *Constrained project, Springer Science Business Media NewYork.* Kluwer Academic Publisher. ISBN 978-1-4613-7093-2.

[18] P.B.Myszkowski, L.Olech, M.Laszczyk, M.Skowronski, 2018. Hybrid Differential Evolution and Greedy Algorithm (DEGR) for solving Multi-Skill Resource-Constrained Project Scheduling Problem. *Applied Soft Computing*, Vol. 62, pp.1-14.

[19] P.B.Myszkowski, M.Skowronski, L.Olech, K. Oślizło, 2015. Hybrid Ant Colony Optimization in solving Multi-Skill Resource Constrained Project Scheduling Problem. *Soft Computing Journal*, Volume 19, Issue 12, pp.3599-3619.

[20] X.S. Yang, S. Deb, 2009. *Cuckoo search via Lévy flights.* Proc. of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), India. IEEE Publications, USA, pp. 210-214.

[21] X.-S.Yang, 2014. Nature-Inspired Optimization Algorithms, first ed., Elsevier, London.

[22] M.I. Solihin, M.F. Zanil, 2016. Performance comparison of Cuckoo search and differential evolution algorithm for constrained optimization. *Intrnational Enginering Research and Innovation Symposium (IRIS),* Vol. 160(1), pp.1-7.

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

[24] P.B.Myszkowski, M.E. Skowronski, K.Sikora, 2015. A new benchmark dataset for Multi-Skill Resource-Constrained Project Scheduling Problem", Computer Science and Information Systems, ACSIS, Vol. 5, pp. 129-138.

[25] P.B.Myszkowski, M.Laszczyk, I.Nikulin, E.Skowronski, 2019. iMOPSE: a library for bicriteria optimization in Multi-Skill Resource-Constrained Project Scheduling Problem. *Soft Computing*, Vol. 23, Issue 10, pp 3397-3410.