HNUE JOURNAL OF SCIENCE
Natural Sciences 2024, Volume 69, Issue 3, pp. 57-66
This paper is available online at http://hnuejs.edu.vn/ns
DOI: 10.18173/2354-1059.2024-0035

THE SOLUTION TO THE PROJECT SCHEDULING PROBLEM BY USING AN IMPROVED GENETIC ALGORITHM

Do Ba Chin[1], Tran Truc Mai[2], Dang Quoc Huu[3,*] and Nguyen The Loc[1]
[1]*Faculty of Information Technology, Hanoi National University of Education, Hanoi city, Vietnam*
[2]*Faculty of Information Technology, VNU University of Engineering and Technology, Hanoi city, Vietnam*
[3]*Faculty of Economic Information System and E-commerce, Thuong Mai University, Hanoi city, Vietnam*
[*]Corresponding author: Dang Quoc Huu, e-mail: huudq@tmu.edu.vn

Received September 25, 2024. Revised October 24, 2024. Accepted October 31, 2024.

Abstract. Nowadays, managing and allocating resources for projects has become increasingly essential for managers. A critical factor affecting the success of a project is the work assignment plan for workers to optimize the completion time. Current solutions to project scheduling problems have not been thoroughly addressed; thus, in this study, we model the labor assignment process in project production as a scheduling problem. To solve this problem, we use an improved genetic algorithm named GA-RT (Genetic Algorithm with Random Crossover and Negative Tournament Selection) and conduct experiments on the iMOPSE standard dataset. Experimental results show that the proposed GA-RT algorithm can effectively solve the project scheduling problem, achieving better performance compared to existing algorithms.

Keywords: project scheduling, scheduling problem, genetic algorithm.

1. Introduction

In industrial production, scheduling workers to perform tasks (Figure 1) is an essential issue. Optimizing the time to complete a product helps save project production time. The plan to assign tasks to workers, ensuring that the priority of tasks and the goal of completing the task in the shortest time are satisfied, is called a schedule. In reality, any task can be performed by several workers with corresponding expertise; a more skilled worker can complete the task earlier. Additionally, some tasks can only start when the previous tasks are completed. From the above practical problem, a suitable problem model is needed to find the optimal solution.

The Real-Resource Constrained Project Scheduling Problem (Real-RCPSP) [1] is a project scheduling problem extended from the original Multi-Skill RCPSP (MS-RCPSP) problem [2]-[5] to schedule projects with limited resources and multiple skills. Real-RCPSP adds constraints on resources based on the skill level required by the task; if the resource performing the task has a higher skill level than required, the task execution can be faster [1]. MS-RCPSP has been proven to be an NP-Hard problem [2], [6], [7], and many studies [4], [8], [9] have utilized Genetic Algorithms [10] to address this problem, including contributions from several research groups.

Myszkowski et al. [8], [9] built and published the iMOPSE standard dataset [8] to replace the PSPLIB dataset [11], adding an information field on task execution costs. The iMOPSE dataset has been recognized and widely used in subsequent publications, along with the GA Runner toolkit to run and verify the results of the MS-RCPSP problem.

The Hosseinian group [4], [12], with research results published [10], used the classical GA algorithm combined with the Shannon-entropy information measure-based decision-making method to select individuals for the next generation to solve the Multi-Mode Multi-Skilled RCPSP problem (MMS-RCPSP), a variant of the MS-RCPSP problem that adds a regime constraint and cannot be changed once the execution process starts.

In studies [5], [13], this group of authors used the Dandelion Algorithm to solve the MS-RCPSP problem [13] and the Pareto-based Grey Wolf Optimizer algorithm for the multi-objective MS-RCPSP problem with two objective functions: project implementation time and cost [2]. The later studies of this group all used the iMOPSE standard dataset for experimentation and verification. Most studies focus on the MS-RCPSP problem and only address the theoretical model, which is not linked to reality. This is evident in the mathematical statement of the MS-RCPSP problem, where the task execution time is constant, regardless of which resource performs the task.

To overcome the above drawbacks, the Real-RCPSP problem has been formulated to be more suitable for real projects. In this paper, we propose a new solution for the Real-RCPSP problem to increase the efficiency of project execution schedules. The main contributions include: (i) modeling the production project scheduling problem in reality, and (ii) proposing the GA-RT (Genetic Algorithm with Random Crossover and Negative Tournament Selection) algorithm to find the optimal solution, helping project managers develop time-saving plans during the project coordination process. The rest of this paper is structured as follows. The next section presents the mathematical formulation of the Real-RCPSP problem, emphasizing the aspects that make this model more realistic than the MS-RCPSP model. Section 3 describes the proposed GA-RT algorithm in terms of:

(i) Increasing the efficiency of crossover by using the Random function;

(ii) Using the Negative Tournament Selection method to remove defective individuals and increase the efficiency of mutation operations;

(iii) Individual representation, objective function, and pseudocode of GA-RT. Section 4 presents and analyzes the experimental results, demonstrating the advantages of the proposed algorithm compared to the previously most efficient GA Runner toolkit [8], [9].

2. Content

2.1. GA-RT algorithm solving the REAL-RCPSP problem

Scheduling is a practical issue with many implications in daily life. The scheduling problem is always challenging because, to achieve desired results, the scheduler must try many different methods to properly utilize human power, resources, tools, machines, etc. In particular, the Real-RCPSP scheduling problem focuses on two factors: resources and tasks (Figure 1).

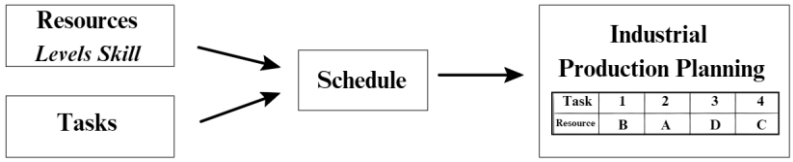

Figure 1. Simulation of the scheduling process for workers to perform jobs

2.1.1. Problem statement

In actual production, workers with higher skills and experience often complete work more quickly or produce better-quality products than lower-skilled workers. Therefore, in the Real-RCPSP problem, the objective function (project execution time) is calculated based on the worker's skill level; if the worker performing the task has a higher skill level than required, the task completion time can be faster.

**** Components of the Real-RCPSP problem***

- $X = \{A_0,...,A_{n+1}\}$: set of tasks to be performed. In which, $A_1,..., A_n$ are tasks to be performed; A_0, A_{n+1} are two hypothetical tasks, added to serve the purpose of determining the start time and end time of the project; $A = \{A_1,..., A_n\}$ set of tasks to be performed.
- F: Set of priorities, $(A_i, A_j) \in F$, task A_i is performed before task A_j.
- T_i : Set of tasks finished before starting time of task I.
- A^k : List all resource tasks that resource k can perform, $A^k \subseteq A$.
- n: Number of tasks; Nr: Number of resources.
- $g = \{g_0, g_1,..., g_{n+1}\}$ execution time of tasks.
- g_i : Execution time of A_i. Special values: $g_0 = g_{n+1} = 0$.
- g_{jk} : Execution time of task j by resource k , the execution time of the same task can be different with different execution resources.
- C_i: Start time of task i; D_i: Time to complete task i, easy to see: $D_i = C_i + g_i$.
- $X^s = \{A_i \in A | C_i \leq s < C_i + g_i\}$: the set of tasks are being performed at time s.
- N: Set of resources , $N = \{N_1, N_2,..., N_k\}$ that can all be reused.
- N^k: List all resources that can perform task k ; $N^k \subseteq N$.
- $H = \{H_1,..., H_k\}$ is a set containing information about resource capacities, H_k represents the capacity of N_k .
- m_{ik} : The number of resources m_k mobilized to perform A_i .
- K: Set of all skills; K^p: Sub set of skills of resource p, $K^p \subseteq K$.

- K_i: Skill i; l_i: Skill level of skill i; q_i: Skill type i.
- z^i: Set of skills required by task i. A resource can perform a task if its skill level is equal to or higher than the skill level required by the task.
- $G_{u,v}{}^i$: Boolean variable that determines whether resource v is performing task u at time i; t: Makespan of the schedule, $t = D_{n+1}$ is the end time of the last task.
- R: A satisfying solution; R_{all}: The set of all satisfying solutions, $R_{all} \subseteq R$.
- $f(R)$: Objective function, to calculate the makespan of R.

The Real-RCPSP problem aims to find a schedule R such that f(R) → min

Subject to the constraints:

$$C_j - C_i \geq g_i \qquad \forall (A_i, A_j) \in F \qquad (1)$$

$$\sum_{A_i \in X^s} H_{in} \leq H_n \qquad \forall N_n \in N, \forall s \geq 0 \qquad (2)$$

$$K^i \neq \emptyset \qquad \forall N_k \in N \qquad (3)$$

$$g_{jk} \geq 0 \qquad \forall A_j \in A, \forall N_k \in N \qquad (4)$$

$$D_j \geq 0 \qquad \forall A_j \in A \qquad (5)$$

$$D_i \leq D_j - g_j \qquad \forall A_j \in A, j \neq 1, W_i \in T_j \qquad (6)$$

$$\forall A_i \in A^k \; \exists K_q \in K^i : q_{k_i} = q_{z_i} \text{ and } l_{k_i} \geq l_{z_i} \qquad (7)$$

$$\forall N_k \in N, \forall s \in t : \sum_{i=1}^{n} G_{i,k}^s \leq 1 \qquad (8)$$

$$\forall A_j \in A \; \exists! s \in [0,t], ! \; N_k \in N : A_{j,k}^s = 1; with \; G_{j,k}^s \in \{0;1\} \qquad (9)$$

$$g_{jk} \leq g_{ju} \; with \; l_k \leq l_u \qquad \forall (z^i, z^u) \in \{K^i \times K^u\} \qquad (10)$$

**** Description of constraints***

Constraint (1) shows the priority order between two parent tasks *(task i)* and child tasks *(task j)*; *task j* only starts when *task i* finishes, and child tasks may not be performed immediately after parent tasks finish. Constraint (2) shows that the number of resources n used to perform task i at time s is at most equal to the resource n 's capacity. Constraint (3) *states* that each resource must have at least one skill. Constraints (4) and (5) require that the execution time of any task must be at least zero. Constraint (6) *requires* that the parent task (task i) must finish before the child task (task j) starts. D_i denoted the time when task i finishes, and when child task j starts, it is $D_i - g_j$.

Constraint (7) for every task $i \in A^k$ (set of tasks that resource k can perform), there always exists skill $K \in K^i$ (skill set of resource k) such that $q_{k_i} = q_{z_i}$: skill type of z coincides with the skill type of N_i that task i requires. Inequality $l_{k_i} \geq l_{z_i}$ means that to perform the task's requirements, the resource must have a skill level greater than or equal to the given requirement. Constraint (8) at each time point (s), each task has only one resource to execute; if $\sum_{i=1}^{n} G_{i,k}^s = 0$, then resource k is not assigned to any task; if $\sum_{i=1}^{n} G_{i,k}^s = 1$ then resource k is assigned to a single task. Constraint (9) states that each task is assigned to only one resource and can be performed by only one resource. The final constraint (10) is a new development of the Real-RCPSP problem compared to the MS-RCPSP problem. This constraint states that the higher the skill level of the resource, the shorter the task execution time.

2.1.2. Proposed algorithm

The Real-RCPSP problem is a large-scale problem with many constraints, so this paper proposes an improved genetic algorithm named GA-RT (Genetic Algorithm with Random Crossover and Negative Tournament Selection), incorporating two techniques (Figure 2) to enhance efficiency as follows:

First, in the individual crossover step, instead of using traditional crossover operators, GA-RT iterates through each task sequentially. For each task in the parent individuals, it uses a Random function:

- If Random (0;1) ≥ 0.5, take the parent's resources to crossbreed to create the child;
- If Random (0;1) < 0.5, take the resource of the other parent to create the child.

Using the Random function to combine favorable traits from parent individuals helps explore a broader solution space, maintain diversity in the population, and foster innovation, thereby improving the quality of solutions over generations and avoiding convergence to suboptimal local solutions.

Second, after the mutation step, GA-RT uses the Negative Tournament Selection method to eliminate flawed individuals by comparing the new offspring with the original parents:

- If the new child is not as good as the original parents, discard the new child;
- Correspondingly, eliminate the worst individual among the three: the new offspring and the original parents.

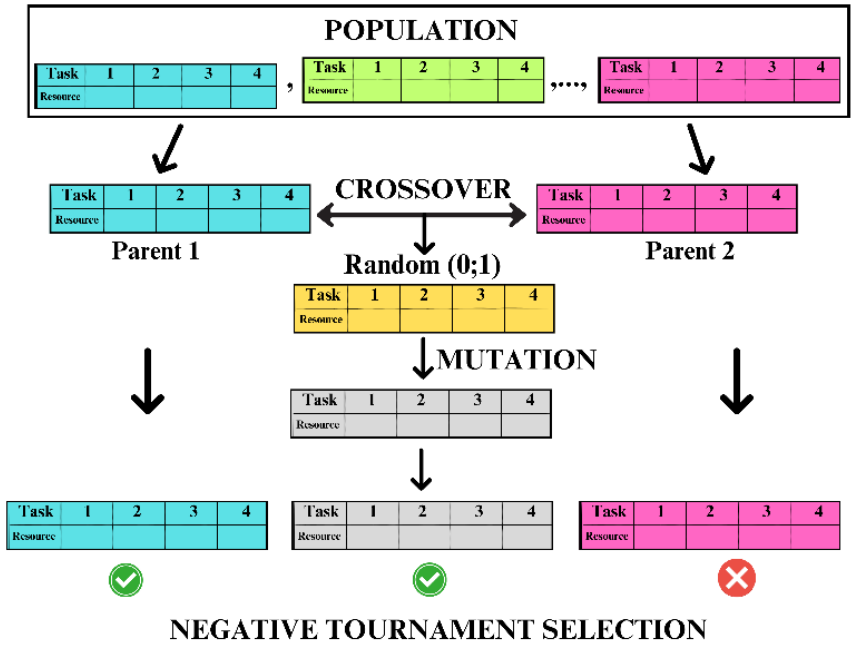

Figure 2. A description of the improvement

Applying Negative Tournament Selection at this point helps maintain genetic diversity in the population because the technique not only focuses on selecting the best individuals but also ensures that the worst individuals are eliminated, which prevents the

poor individuals from negatively affecting the next generations. Eliminating the poor individuals reduces the risk of the population falling into suboptimal local *convergence points.*

**** Steps to implement the algorithm***

- *Solution representation***:** To represent an individual, a one-dimensional vector (Table 1) must be used with: (i) each element in the vector corresponds to a project task and (ii) the value of each element is conventionally assigned by the index of the element. The individual vector meets the requirements when it satisfies the problem's constraints. Two factors must be considered for building a suitable individual: (i) the resources used must meet the standard time to perform the task (minimum time), and (ii) the capacity of the resource must meet the requirements of the given task.

- *Objective function***:** f(R), was descripted in section 2.1.1.

- *Random initialization method***:** Individuals are created completely randomly: (1) Create a list of resources that can perform each task; → (2) For each task, randomly select a resource from the list created in step 1; → (3) Repeat step 2 with the remaining tasks in the schedule; → (4) A new individual is created after each individual has its corresponding worker.

- *Individual selection***:** The mating parent pairs are randomly selected from the population; random selection helps maintain genetic diversity, creating a balance between exploiting existing good individuals and exploring new potential solutions, thereby increasing the chance of finding the global optimal solution.

- *Crossbreeding:* For ensuring genetic diversity, the GA-RT algorithm performs crossbreeding as follows: (1) Browse the tasks in turn at both the parent and child individuals; → (2) Select the resource to perform the corresponding task using the Random function. If *Random (0;1)* >= 0.5, take the resources of the parent individual to crossbreed to create a child individual; otherwise, take the resources of the parent individual; → (3) The child individual created randomly has the characteristics of both the parent and child.

- *Mutation:* The mutation probability is set to 0.05 to avoid changing the population's genetic structure. (1): Randomly select tasks to mutate at a rate of 0.05 (that is, assuming an individual has 100 tasks, select five tasks to mutate); → (2) Randomly select a resource from the available resource set for each selected task; → (3) The child individual created is of course satisfied because it is randomly mutated within the constraint.

- *Negative Tournament Selection Method:* As a variation of Tournament Selection, Negative Tournament Selection helps create diversity in the population and makes the evolution of the population smoother. Implementation steps: (1) Compare the new offspring after mutation with the original parents; → (2) Select and eliminate the worst individuals out of the three individuals; → (3) Add the remaining two individuals to the population.

- *Replacement:* Newly generated individuals replace old individuals after re-evaluating the new population using the fitness function, thereby forming a new population for the next generation.

The overall GA-RT algorithm (Figure 3) is depicted as follows:

Begin

Initialize the population by using a random method

Population assessment using the fitness function

Repeat

Randomly select two individuals for crossbreeding;

Perform re-mapping with random cross probability;

Perform random mutation;

Negative Tournament Selection Method;

Re-evaluate the new population using the fitness function;

Replace;

Until the stopping condition is satisfied

Return the best individual in the population

End

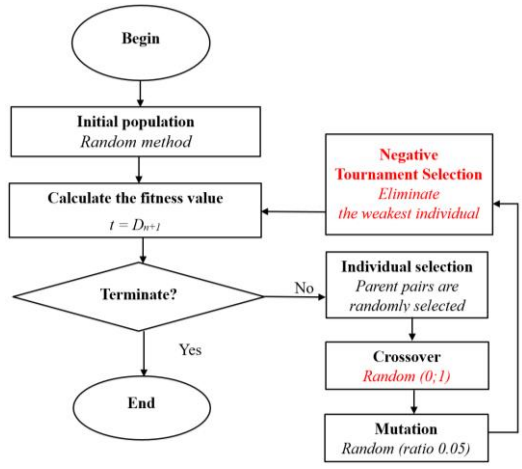

Figure 3. GA-RT algorithm

2.2. Experiment

To verify the effectiveness of the GA-RT algorithm, we conducted experiments using the published iMOPSE [4], and [5] dataset, which has been widely used by research groups. The iMOPSE dataset includes 30 files; the information fields include Task, Resource, Precedence Relation, and Skill. To match the actual production scenarios of the iMOPSE dataset, the calibrated iMOPSE dataset was processed as follows: (1) Level 1: For resources with skill levels equal to or higher than level 1, the execution time remains unchanged. (2) Level 2: For resources with skill levels 2 and 3, the task execution time is decreased by 5% compared to the standard time. (3) Level 3: For resources with skill levels from 4 to 7, the task execution time is reduced by 7% compared to the standard time.

Experimental parameters: Initial population initialization with 50 individuals; Number of evolutionary generations: 5000; Number of experimental runs on each data set: 20. The experiment was conducted in the Java environment; the experimental computer has an Intel(R) Core(TM) i3-10105F CPU configuration, 3.7 GHz speed, 16GB RAM, Windows 10 Pro (64 bit) operating system.

Table 1. Comparison results

No.	Dataset	Best		Avg		Std	
		GA-RUNNER	***GA-RT***	***GA-RUNNER***	***GA-RT***	***GA-RUNNER***	***GA-RT***
1	Dataset 01	503	*487*	507	*500*	3.1	7.9
2	Dataset 02	574	*533*	579	*546*	4	7.1
3	Dataset 03	528	*491*	533	*493*	4.5	*1.6*
4	Dataset 04	520	*483*	530	*488*	9.5	*4.5*
5	Dataset 05	504	*475*	507	*478*	2.7	*2.5*
6	Dataset 06	277	*242*	279	*254*	2	7.5
7	Dataset 07	286	*261*	294	*270*	7.2	8.9
8	Dataset 08	276	*251*	284	*257*	7.5	*3.9*
9	Dataset 09	291	*252*	301	*263*	8.7	*6.4*
10	Dataset 10	276	*251*	281	*258*	4.3	*4.0*
11	Dataset 11	157	*141*	163	*152*	4.7	6.6
12	Dataset 12	185	*159*	190	*170*	5.1	6.5
13	Dataset 13	179	*143*	181	*155*	2.3	7.7
14	Dataset 14	234	*276*	241	*276*	6.8	*0.0*
15	Dataset 15	173	*137*	181	*156*	6.9	8.8
16	Dataset 16	562	*495*	568	*508*	5	9.2
17	Dataset 17	546	*492*	550	*501*	2.9	6.1
18	Dataset 18	580	*491*	581	*504*	1.3	8.7
19	Dataset 19	556	*488*	568	*506*	11	11.5
20	Dataset 20	546	*485*	550	*491*	2.7	4.9
21	Dataset 21	317	*254*	324	*270*	7.3	8.6
22	Dataset 22	349	*280*	356	*300*	5.6	12.9
23	Dataset 23	300	*274*	306	*285*	5.1	10.6
24	Dataset 24	422	*371*	426	*371*	3.4	*3.1*
25	Dataset 25	310	*264*	317	*280*	6.8	13.7
26	Dataset 26	210	*172*	219	*188*	8.8	11.4
27	Dataset 27	190	*160*	194	*177*	3.4	8.7
28	Dataset 28	202	*171*	216	*189*	12.6	*11.9*
29	Dataset 29	207	*171*	211	*188*	3.7	12.5
30	Dataset 30	187	*170*	193	*182*	5.7	8.8

The experimental results in Table 1 show that the GA-RT algorithm consistently achieves better Best values than the GA-RUNNER in 29 out of 30 cases. The improvement ranges from approximately 3.18% to 20.87%, with GA-RT being, on average, 10% to 15% faster than GA-RUNNER in most datasets. Regarding the average

(Avg) values, GA-RT also outperforms GA-RUNNER in 29 out of 30 cases, with improvements up to 16.66%. The standard deviation (Std) values indicate that GA-RT has a minor standard deviation, demonstrating that the GA-RT results are more stable than those of GA-RUNNER in many cases. Notably, in one case, the GA-RT algorithm has a standard deviation of 0.0, indicating absolute stability.

With the Random function combined with Negative Tournament Selection, the GA-RT algorithm becomes powerful in project optimization. The GA-RT algorithm leverages the superior characteristics of parent individuals to create offspring with higher optimization ability while maintaining diversity in the population. This approach helps explore a broader solution space and prevents convergence to suboptimal solutions. Additionally, Negative Tournament Selection eliminates weak individuals, ensuring that they do not negatively impact subsequent generations. As a result, genetic diversity and solution quality are maintained and improved through each generation, enabling the population to continuously evolve and avoid getting stuck in inefficient local convergence points.

3. Conclusions

This study presents the GA-RT algorithm for addressing the current project resource scheduling problem. The GA-RT algorithm is improved from the traditional GA by incorporating a Random function in the crossover process and utilizing the Negative Tournament Selection technique to enhance the quality of solutions. Experiments on the iMOPSE dataset demonstrate that our algorithm finds better solutions and is more stable than previous algorithms. The main contribution of this paper is the proposal, development, and validation of a new algorithm to improve efficiency in managing projects with multiple resources, thereby helping managers optimize time and cost. In the future, research will be extended by applying Deep Reinforcement Learning (DRL) mechanisms to improve the convergence speed and solution quality of the algorithm, making it better suited to meet the requirements of practical projects. Another potential research direction is to enhance the algorithm and apply it to multi-objective scheduling problems.

Acknowledgment. This research was supported by the Ministry of Education and Training, Vietnam, under the Science and Technology Project No. B2024-SPH-14 supervised by the Hanoi National University of Education, Vietnam.

REFERENCES

[1] Dang QH, Nguyen TL, Nguyen DC, Xiong N, (2020). Effective Evolutionary Algorithm for Solving the Real-Resource Constrained Scheduling Problem. *Journal of Advanced Transportation*, 2020(1). DOI: 10.1155/2020/8897710.

[2] Błazewicz J, Lenstra JK & Rinnooy Kan AHG, (1983). Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5(1), 11-24.

[3] Skowroński M, Myszkowski PB, Kwiatek P & Adamski M, (2013). Tabu Search approach for Multi–Skill Resource–Constrained Project Scheduling Problem. *2013 Federated Conference on Computer Science and Information Systems*, Krakow, Poland, 153-158.

[4] Hosseinian AH & Baradaran V, (2019). 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*, 12(2), 155-178.

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

[6] Christian A, Demassey S & Néron E, (2008). *Resource Constrained Project Scheduling: Models, Algorithms, Extensions and Applications* (1st ed), Wiley-ISTE.

[7] Klein R (2012). *Scheduling of Resource-Constrained Projects, Springer Science & Business Media* (2000th ed), Kluwer Academic Publishers.

[8] Myszkowski PB, Laszczyk M, Nikulin I & Skowro M, (2019). iMOPSE: a library for bicriteria optimization in Multi-Skill Resource-Constrained Project Scheduling Problem. *Soft Computing Journal*, 23(11), 2297-3410.

[9] Myszkowski PB, Skowronski ME & Sikora K, (2015). A new benchmark dataset for Multi-Skill Resource-Constrained Project Scheduling Problem, *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*, Lodz, Poland, 129-138. DOI: 10.15439/2015F273.

[10] Katoch S, Chauhan1 SS & Kumar V, (2021). A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, 80, 8091-8126.

[11] Kolisch R & Sprecher A, (1997). PSPLIB-a project scheduling problem library: OR software-ORSEP operations research software exchange program. *European Journal of Operational Research*, 96(1), 205-216.

[12] Hosseinian AH, Baradaran V & Bashiri M, (2019). Modeling of the time-dependent multi-skilled RCPSP considering learning effect. *Journal of Modelling in Management*, 10(2), 521-558.

[13] Hosseinian AH & Baradaran V, (2019). Detecting communities of workforces for the multi-skill resource-constrained project scheduling problem: A dandelion solution approach. *Journal of Industrial and Systems Engineering*, 12(Special issue on Project Management and Control), 72-99.