Hindawi
Applied Computational Intelligence and Soft Computing
Volume 2022, Article ID 6230145, 12 pages
https://doi.org/10.1155/2022/6230145

Research Article

An Effective Hybrid Algorithm Based on Particle Swarm Optimization with Migration Method for Solving the Multiskill Resource-Constrained Project Scheduling Problem

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

[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*

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

Received 31 October 2021; Revised 17 January 2022; Accepted 20 January 2022; Published 15 February 2022

Academic Editor: Upaka Rathnayake

The paper proposed a new algorithm to solve the Multiskill Resource-Constrained Project Scheduling Problem (MS-RCPSP), a combinational optimization problem proved in NP-Hard classification, so it cannot get an optimal solution in polynomial time. The NP-Hard problems can be solved using metaheuristic methods to evolve the population over many generations, thereby finding approximate solutions. However, most metaheuristics have a weakness that can be dropping into local extreme after a number of evolution generations. The new algorithm proposed in this paper will resolve that by detecting local extremes and escaping that by moving the population to new space. That is executed using the Migration technique combined with the Particle Swarm Optimization (PSO) method. The new algorithm is called M-PSO. The experiments were conducted with the iMOPSE benchmark dataset and showed that the M-PSO was more practical than the early algorithms.

1. Introduction

Scheduling problems are studied to solve many cases of science and practice, especially big projects which have many tasks. Actually, each real project often has a much larger amount of tasks than the number of renewable resources: people, tools, machines, devices, etc., which can be used to do all of them. Therefore, it is very important to find a good solution to assign resource-to-task efficiently, satisfying the project's preconstraints. The research to solve this problem has many useful implications such as scheduling of resource coordination in operating systems [1] and scheduling for production lines or applications in subjects of economy and finance [2], military [3], cloud computing [4], fog computing [5], wireless sensor networks [6], etc. Many research results published have shown that this problem is classified into NP-Hard [7–9].

Resource-Constrained Project Scheduling Problem (RCPSP) [1, 10, 11] is a project scheduling problem with limited resources, which has been proven to be an NP-Hard problem. The goal is to find a good schedule with minimal time in terms of resource constraints. Currently, this problem is studied and applied in many fields. According to the problem definition, it has two important constraints:

(i) Each renewable resource can only perform a task at a time. Until that task completes, the resource can be used to execute another task.

(ii) When a task has started, it cannot be interrupted until it completes.

In each project, tasks have precedence constraints, which means some tasks need to be finished before others.

An important problem extended from RCPSP is MS-RCPSP [7, 8, 12] (multiskill RCPSP) that has a new constraint on the skill factor of implementation resources. The new characterization of this problem states that a resource may have more than one skill type, and each skill type has a specific skill level. Each task will have exact requirements about the skill type and skill level of the performing resource. This definition makes the MS-RCPSP more appropriate to real-life projects. The problem is solved by initializing a population of many feasible schedules for the project. The population will evolve step by step through generations to find the best schedule. However, evolution methods often fall into local extremes, so the population cannot run further. This paper proposes a new algorithm using the Migration method combined with PSO traditional to crack the disadvantage when falling into the local extremes.

The innovations and main contributions of this paper are described as follows.

(i) Detecting local extremes in the population evolution.

(ii) Proposing the method to escape local extremes by moving the population to a new values space using the Migration.

The rest of the paper is separated into six sections. Section 2 introduces some early approaches to decoding the MS-RCPSP problem. In Section 3, we raise the mathematical statement of the MS-RCPSP problem. Section 4 presents a new efficient algorithm for finding the approximate solution based on the PSO strategy [2, 13]. In this section, we focus on detailing the Migration technique, which is a major component that makes up the power of the proposed algorithm. To evaluate the algorithm, this section also introduces the particle presentation. Section 5 experiments to demonstrate the efficiency of the proposed algorithm. The examination is conducted on some instances of the iMOPSE benchmark dataset. All experimental results are compared and analyzed with other algorithms to show the effectiveness of the new algorithm. Finally, Section 6 arranges the paper and draws further research directions to enhance the quality of the MS-RCPSP problem.

2. Related Works

2.1. Approximation Methods for MS-RCPSP. In recent years, researching good methods to solve the MS-RCPSP [7, 12, 14] problem has become important in deploying it into many domains. Since this is a combinatorial optimization problem belonging to the NP-Hard [11, 15, 16] classification, we cannot find an optimal solution in polynomial time, so the objective of methods is to find an approximate result based on metaheuristic techniques. Authors usually use evolutionary approaches as GA [17, 18], PSO [19–22], Greedy, Min-Max, etc. to solve and get out the approximate solutions. Some of the outstanding related works are shown in Table 1.

Myszkowski et al. [8, 9, 23] have researched this problem for many years, and they have many quality publication papers presented regarding that. The authors proposed algorithms based on evaluation methods such as Tabu Search [8], Ant colony, GA [9, 24, 25], etc. Most of them are traditional methods, so they usually have limited effective results. However, the best contribution of Myszkowski is to build and publish the standard dataset used to experiment with the new algorithms for the MS-RCPSP. It is called the iMOPSE dataset [9].

Hosseinian and Baradaran [7, 26] have many published papers about MS-RCPSP too. In 2018, Hosseinian and Baradaran published a paper [7] on the Multimode MS-RCSPSP (MMSRCPSP), a subclass of the MS-RCPSP problem. MMSRCPSP has added constraints where each task can only be executed in a few predefined modes. When the mode is selected, it cannot be changed. The authors suggested a method to make individuals in the next generation, which is built from the GA algorithm combined with the decision-making method based on Shanon-entropy data measure. Experiments were carried out on randomly generated datasets by ProGen software. After that, in 2019, the authors published a paper [26] on the Dandelion Algorithm [27]. In 2020, this group of authors continued to publish an article that solves the multiobjective MS-RCPSP problem with two objectives: total time and cost to complete a project. The authors use the Pareto-based Gray Wolf Optimizer algorithm and continue to test the results on the iMOPSE standard dataset [9].

In 2017, Javanmard et al. used two traditional evolutionary algorithms, GA and PSO, to schedule multiskill resources (actually workers and engineers) to minimize the total cost in the chemical industry [28]. The proposed algorithms are tried on the PSPLIB dataset [29] not fully suitable for scheduling problems because of no cost parameter objectives. Moreover, the authors cannot compare their traditional algorithms with newer ones at the experimental stage but only compare them.

Also studying multiobjective problems such as Hosseinian et al., Davari-Ardakani [30] proposed a multiobjective variant of the MS-RCPSP problem to minimize the time and cost of the current project. The proposed issue is called MSPSP, which is limited to projects with the following two characteristics:

(i) The timing of implementation is arbitrary; for example, the project can be done in the evenings or on weekends

(ii) Energy costs are very high, comparable to wages paid to employees.

However, the authors only gave the problem model without any new solution method to solve it. They only conducted experiments with the Max-Min method, a very traditional method.

2.2. PSO Method. Particle Swarm Optimization (PSO) [31–35] is an evolutionary algorithm. Similar to other evolutionary algorithms, PSO will perform a population-based search; initially the population is randomly initialized with a certain number of individuals. However, PSO differs

Table 1: Summary of early work.

No.	Year	Author	Method	Dataset
1	2017	Javanmard [47], [51]	GA, PSO	PS-LIB
2	2013–2019	Myszkowski và cộng sự [31], [42], [43], [44], [45]	GA, Ant, Greedy,...	iMOPSE
3	2018	Hosseinian [4], [5], [6], [7]	GA, PSO	ProGen, iMOPSE
4	2019	H. Davari-Ardakani [21]	Min-Max	iMOPSE

from other evolutionary algorithms in that each individual in the population is determined by two basic parameters, the position vector (representing the individual's experience over generations) and the velocity vector (representing the population's experience over generations). Each individual will move in the solution space at a particular velocity. After each generation, the individuals will move towards the best position of the individual in the past and the best position of the population and after each generation, the instances will move towards better search regions in the search space. In the process of finding individuals, the displacement vector and the position vector will be updated according to the best value of that individual in the past and the best position of the population according to the following formula.

An effective algorithm suggested by Kennedy and Eberhart [32] in 1995 in the evaluation optimization method is PSO (Particle Swarm Optimization). In the PSO, each particle in each generation is evaluated from two values containing position and velocity where the position is calculated as follows:

$$\begin{aligned} v_i^{k+1} &= \omega v_i^k + c_1 \text{rand}_1 \times \left(\text{pbest}_i - x_i^k\right) \\ &\quad + c_2 \text{rand}_2 \times \left(\text{gbest}_i - x_i^k\right), \\ x_i^{k+1} &= x_i^k + v_i^k, \end{aligned} \tag{1}$$

where

(i) v_i^k, v_i^{k+1} are velocity of particle i at generation k and $k+1$.

(ii) x_i^k, x_i^{k+1} are position of the particle i at generation k and $k+1$.

(iii) ω is inertia weight; c_1, c_2 are speedup coefficients.

(iv) rand_1, rand_2 are the values between 0 and 1 randomly generated.

(v) pbest_i is the best position of particle i.

(vi) gbest: the best particle position in a population.

3. Problem Definitions

The MS-RCPSP [7, 8, 12, 14] is a subproblem of RCPSP (Resource-Constrained Project Scheduling Problem) that added the skill domain to the renewable resources. The objective of this problem is to find the minimum makespan, i.e., the minimum time to complete the full project. In the running time, each task requires a particular skill, where skill level is equal to or greater than the task's requirement.

To define formulations of the MS-RCPSP, we need a notation system presented in Table 2.

Using the notations presented in Table 2, we can construct the mathematical model for the MS-RCPSP as follows.

$$f(P) \longrightarrow \min, \tag{2}$$

subject to

$$S^k \neq \varnothing, \quad \forall L_k \in L, \tag{3}$$

$$t_j \geq 0, \quad \forall W_j \in W, \tag{4}$$

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

$$E_i \leq E_j - t_j, \quad \forall W_j \in W, j \neq 1, W_i \in C_j \tag{6}$$

$$\forall W_i \in W^k, \exists S_q \in S^k, \quad g_{S_q} = g_{r^i} \text{ and } h_{S_q} \geq h_{r^i}, \tag{7}$$

$$\begin{aligned} &\forall L_k \in L, \\ &\forall q \in m: \sum_{i=1}^{n} A_{i,k}^q \leq 1, \end{aligned} \tag{8}$$

$$\begin{aligned} &\forall W_j \in W, \exists! q \in [0, m], \\ &!L_k \in L: A_{j,k}^q = 1; \text{ with } A_{j,k}^q \in \{0; 1\}, \end{aligned} \tag{9}$$

where we have the following:

(i) Constraint (3): a resource must have one or more skills.

(ii) Constraints (4) and (5): the duration of any task must be equal to or greater than zero (in fact, any real task's duration is a positive number; only two dummy tasks have the duration equal to 0, which shows the start and finish time of the project).

(iii) Constraint (6): the parent task (task i) must be finished before the child task (task j) starts. When task i ends are denoted E_i, the time when subtask j starts is $E_j - t_j$.

(iv) Constraint (7): any task $i \in W_k$, having skill $S \in S_k$ satisfied to $g_S = g_{S_i}$ shows that the skill of r is the save with skill of L_i. $h_{S_q} \geq h_{r^i}$: skill level of using the resource is equal to or greater than the task's requirement.

(v) Constraint (8): at any time, each resource can execute only one task. To check the availability of the resource, we use the expression $\sum_{i=1}^{n} A_{i,k}^q$; if it results in 0 value, the k resource cannot be assigned to do any task, else k is used.

(vi) Constraint (9): each task is executed by only one resource.

In the MS-RCPSP problem, the resource must meet the skill type and skill level to perform a task. Accordingly, a

Table 2: The notations.

C_i	The set of tasks need to be completed before task i can be executed
S	The set of all resource's skills S^i: the subset of skills owned by the resource i, $S^i \subseteq S$
S_i	The skill i
t_j	The duration of task j
L	The resources used to execute tasks of the project
L^k	The subset of the resources which can perform task k; $L^k \subseteq L$
L_i	The resource i
W	The tasks that the project needs to do
W^k	The subset of task which can be executed by the resource k, $W^k \subseteq W$
W_i	The task i
r^i	The subset of the skill required by task i. A resource has the same skill and skill level equal to or greater than the requirement that can be performed
B_k, E_k	The begin time and end time of the task k
$A_{u,v}{}^t$	The variable to identify the resource v is running task u at time t; 1: yes, 0: no
h_i	The skill level i
g_i	Type of skill i
m	Makespan of the schedule
P	The feasible solution
P_{all}	The set of all solution
$f(P)$	The function to calculate the makespan of P solution
n	Task number
z	Resource number

resource can commit the task if its skill is the same as the task's required, and the skill level must be equal to or greater than the tasks needed. The resource-to-task assignment can be presented as a matrix illustrated in Figure 1.

In Figure 1, $S_{i,j}$ denotes S_i skill type and it has j skill level. Tasks that require execution resources must meet a specific skill type and skill level. The feasible resource should have the same skill type and skill level greater than or equal to the required skill level.

Example 1. Figure 1 presents a project with four tasks and four renewable resources. The resources have skill types from S_1 to S_2, and each of them can have a skill level from 1 to 3. According to MS-RCPSP constraint, to do task W_1, the resource must meet the task's requirement that means the resource has to have skill type S_2 and skill level equal to or greater than 2. Looking into the resource list, we determine that resource L_1 has skill type S_2 and skill level is 2, so L_1 can be allocated to execute W_1.

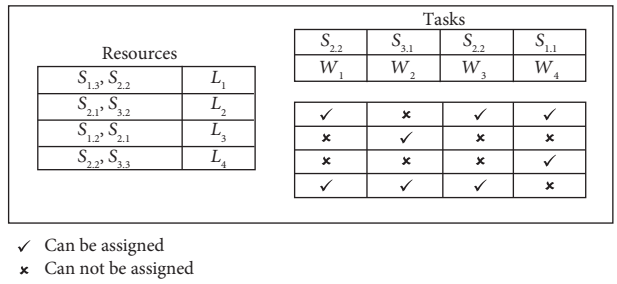

Figure 1: The relation matrix of resources-tasks assignment.

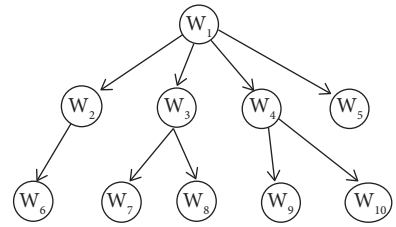

Figure 2: The priority graph of the project.

4. Proposed Algorithm

4.1. Schedule Representation. In the scheduling algorithm, a schedule is represented as a vector and the vector's item number is equal to the task number of the project. The value of each item shows the resource index to perform it. The corresponding resource has to meet the requirement constraint of the task.

Example 2. Consider a project that has 10 tasks and 2 resources.

(i) The set of tasks: $W = \{W_1, W_2, W_3, W_4, W_5, W_6, W_7, W_8, W_9, W_{10}\}$.

(ii) The set of resources: $L = \{L_1, L_2\}$.

The priority graph of the tasks is shown in Figure 2.

The duration of tasks (in hours) is shown in Table 3.

Figure 3 shows the assigning of resources to execute tasks of the project. The resource to perform the task needs to meet the priority constraints of the problem containing the skill requirements and skill level.

Table 3: The duration of task.

Task	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}
Duration	2	4	3	5	2	2	5	3	4	2

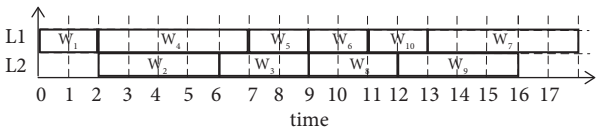

Figure 3: The Gantt diagram of the resource-task assignment.

Table 4: The task-resource assignment of schedule.

Task	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}
Resource	1	2	2	1	1	1	1	2	2	1

A vector can present the schedule shown in Figure 3 as Table 4.

Table 4 presents the task-resource assignment of the project, where the resource L_1 performs tasks: W_1, W_4, W_5, W, W_7, W_{10}; the resource L_2 is assigned to execute tasks: W_2, W_3, W_8, W_9.

4.2. Migration Method. To find a feasible solution with the minimum makespan for the MS-RCPSP problem, we study the Migration method combined with the PSO [32–34]; the new algorithm is named M-PSO.

4.2.1. Migration Method. The PSO algorithm tends to fall into the local extreme when performing evolution. The Migration method moves the population from the local extreme to new space while expanding the search space.

Definition 1. A population is said to be not successfully evaluated if the makespan of the population is still fixed between two continuous generations.

The Migration method runs over steps as follows:

Step 1: detecting the local extremes:

To detect local extremes, we use a variable n_f to count the number of times the population has failed to evolve consecutively; if this value is greater than a specified threshold ($n_{\max}$), then the population has fallen into the local extreme. Equation (10) shows how to evaluate n_f.

$$n_f = \begin{cases} n_f + 1, & \text{if new}_{\text{makespan}} = \text{old}_{\text{makespan}}, \\ 0, & \text{if new}_{\text{makespan}} \neq \text{old}_{\text{makespan}}, \end{cases} \tag{10}$$

Step 2: moving the population to a new space:

To move the population staying local extreme to new space, we continue with some next steps:

(i) Consider each particle of the population.

(ii) For each task i of the particle, find the set L^i of resources that can perform task i, sorting the resources in L^i by the index of each resource.

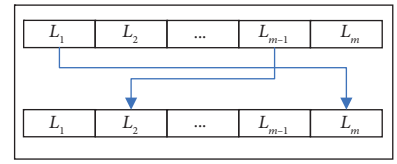

Figure 4: Changing task execution resources.

(iii) Replace the current resource used to do i task by resource staying at the opposite position in L^i sorted. Equation (11) represents this step.

$$\begin{aligned} L^i &= \{L_1, L_2, \ldots, L_m\}, \quad m \leq n, \\ \forall W_i \in W, j \in [1, m]&: L_j \leftarrow L_{m-j}. \end{aligned} \tag{11}$$

Figure 4 illustrates the Migration method by changing execution resources. Specifically, the task being performed by resource L_2 will be assigned to L_{m-1} and the task being performed by resource L_m will be allocated to L_1.

The new assigning resource-to-task has to ensure the problem's constraints. After all the particles of the population are moved, we will get a new population that is migrated to a new space, which means the population is escaping to the local extreme.

The Migration pseudocode is implemented as follows (Algorithm 1):

4.3. The M-PSO Algorithm. The M-PSO algorithm is improved from the PSO algorithm integrated with the Migration method to improve the efficiency of results. The detail of M-PSO is described in Algorithm 2 as follows:

Lines 21 to 25 show the way to find out the local extreme.

Lines 31 to 34 check the threshold and call Migration function to move the population to a new space, making the population escape the current local extremum area and expand finding space.

5. Experiment

To estimate the efficiency of our proposed method, we developed the simulator on the Matlab environment conducted on the benchmark iMOPSE dataset. The test results are compared with GA-M [9] and GRASP [9, 23], showing that M-PSO has a positive effect.

5.1. The Benchmark Dataset. In the simulated program, we use the iMOPSE [9] installers to investigate many existing algorithms as GRASP, GA-M, etc. The iMOPSE dataset has the following characteristics:

(i) Number of project tasks.

(ii) Number of resources used to implement the project.

(iii) Number of precedence constraints performed between tasks.

(iv) The number of skills of the resources in the project.

Table 4 shows 12 iMOPSE's instances to experiment with our algorithms.

```
Input: P_all–the current population
Output: P_new–the population after moving
Begin
(1)   P_new = {}
(2)   For i = 1 to size(P)
(3)     P_i ← P_all[i];
(4)     For j = 1 to n
(5)       L^i ← the subset of resource can be performed the task L_i
(6)       L^i ← Sort(L^i)
(7)       idx ← index of resource executed the task i
(8)       idx = size(L^i) – idx + 1
(9)       L_i = L^i[idx]
(10)    End for // j
(11)    P_new = P_new + {P_i}
(12)  End for // i
(13)  Return P_new;
End Function
```

Algorithm 1: Migration.

```
Input: n_max: the threshold to find local extremal, t_max: number of evolution generations.
Output: the best solution g_best
(1)   Begin
(2)     P_all ← Init data from iMOPSE dataset and create population.
(3)     t = 0
(4)     n_f = 0
(5)     while (t < t_max)
(6)       t = t + 1
(7)       for i = 1 to size (P_all) do
(8)         Calculate the objective value: f(P_i)
(9)       end for
(10)      for i = 1 to size (P_all) do
(11)        if f(P_i) < f(fitness_i) then
(12)          pbest_i = P_i
(13)          f(pbest_i) = f(P_i)
(14)        end if
(15)      end for
(16)      for i = 1 to size (P_all) do
(17)        if (f(P_i) < f(g_best)) then
(18)          g_best = P_i
(19)          f(g_best) = f(P_i)
(20)        end if
(21)        if f(P_i)! = f(P_i-1) then
(22)          n_f = 0
(23)        else
(24)          n_f + = 1
(25)        end if
(26)      end for
(27)      for i = 1 to size (P_all) do
(28)        update velocity vector
(29)        update position vector
(30)      end for
(31)      if (n_f = n_max)
(32)        P_all ← Migration (P_all)
(33)        n_f = 0
(34)      end if
(35)    end while
(36)    return g_best
(37)  end
f: objective function
```

Algorithm 2: M-PSO.

Table 5: iMOPSE benchmark dataset.

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

The simulation is run on a PC with Intel Core i5-CPU 2.2 GHz, 6 GB RAM, and using the Windows 10 OS. The dataset used in our simulator is listed in Table 5.

The input parameters and initial values are the following:

(i) The experiment environment: Matlab.

(ii) Data: 30 iMOPSE installers that are shown in Table 5.

(iii) Number of particles in the population N_p: 100.

(iv) Number of evolutionary generations N_g: 50,000.

(v) Number of executing times for a data install: 35.

All result data contains average, standard deviation, and best values.

5.2. Experiment Results. The results of the M-PSO are presented in Table 6; this table also shows the values of GA-M and GRASP algorithms published together with the iMOPSE dataset.

In Table 6, GA-M and GRASP are proposed algorithms of Myszkowski which are executed on the iMOPSE dataset.

These values are calculated after conducting experiments on the corresponding installers, including

(i) BEST: the best makespan value.

(ii) AVG: the average makespan value.

(iii) STD: the standard deviation value.

Comparing the M-PSO with GA-M, we have the following:

(i) Regarding BEST and AVG values: M-PSO is better than GA-M in all instances, in which BEST value is better than GA-M from 6.2% to 32.97%, and AVG value is better than GA-M from 4.3% to 33.56%.

(ii) Regarding STD, the total standard deviation of GA-M is 173.3 and of M-PSO is 83.6. It denotes that the M-PSO algorithm is more efficient and more stable than the GA-M algorithm.

Comparing the M-PSO with GRASP, we have the following:

(i) The results of M-PSO have better results than GRASP in most cases, specifically better than GRASP 27/30 instances (from 0.6% to 15.8%) with the BEST value and 27/30 instances data with AVG value (from 0% to 16.9%).

The M-PSO can detect local extremes and escape them using the Migration technique, achieving higher efficiency than other comparison algorithms. The experiment results also show that the more the tasks in the project are, the more the M-PSO becomes satisfactory. The superiority of M-PSO

Table 6: The experiment results on the iMOPSE dataset.

Dataset	GA-M			GRASP			M-PSO		
	Best	Avg	Std	Best	Avg	Std	Best	Avg	Std
100_5_22_15	517	524	5.3	503	504	**1.2**	**485**	**486**	1.7
100_5_46_15	584	587	4.7	552	556	3.9	**530**	**531**	**0.5**
100_5_48_9	528	535	9.7	510	510	**0.3**	**490**	**492**	1.6
100_5_64_15	527	530	2.5	501	502	**1.2**	**478**	**479**	1.3
100_5_64_9	508	521	9.9	494	494	**0.2**	**471**	**474**	2.5
100_10_26_15	292	292	**0.0**	251	258	7.1	**233**	**233**	0.4
100_10_47_9	296	296	**0.0**	263	264	0.9	**257**	**261**	3.7
100_10_48_15	279	282	2.9	256	257	1.2	**244**	**245**	**0.6**
100_10_64_9	296	305	6.6	255	257	**2.1**	**240**	**246**	5.9
100_10_65_15	286	290	5.0	256	260	3.8	**244**	**245**	**1.3**
100_20_22_15	163	169	5.8	134	137	3.7	**128**	**131**	**3.1**
100_20_46_15	197	207	6.9	170	174	3.9	**162**	**165**	**2.9**
100_20_47_9	185	186	**0.5**	133	140	6.8	**124**	**128**	3.6
100_20_65_15	240	240	0.5	**213**	**213**	**0.1**	228	230	1.8
100_20_65_9	181	187	4.5	135	135	0.4	**125**	**125**	**0.3**
200_10_128_15	577	583	4.9	491	496	5.2	**461**	**464**	**2.7**
200_10_50_15	553	577	17.5	524	528	3.8	**486**	**487**	**0.7**
200_10_50_9	585	589	5.0	506	508	**1.9**	**484**	**488**	3.7
200_10_84_9	567	583	11.4	526	527	**0.8**	**508**	**510**	2.1
200_10_85_15	549	555	4.9	496	498	**1.7**	**473**	**476**	2.5
200_20_145_15	326	328	2.0	262	271	8.5	**236**	**237**	**1.4**
200_20_54_15	363	385	20.8	304	308	3.7	**256**	**256**	**0.3**
200_20_55_9	312	318	4.2	257	258	**0.6**	**251**	**255**	3.5
200_20_97_15	424	438	9.7	347	347	**0**	**334**	**337**	**3.3**
200_20_97_9	321	326	6.2	**253**	256	3.8	255	**256**	**0.9**
200_40_133_15	215	222	6.2	163	170	6.5	**147**	**151**	**4.1**
200_40_45_15	201	210	6.3	164	**164**	**0.3**	**163**	167	4
200_40_45_9	209	213	2.9	**144**	**147**	**3.2**	152	162	9.5
200_40_90_9	211	215	3.1	148	153	**4.9**	**145**	**152**	7.1
200_40_91_15	200	205	3.4	153	159	**5.7**	**135**	**143**	8.3

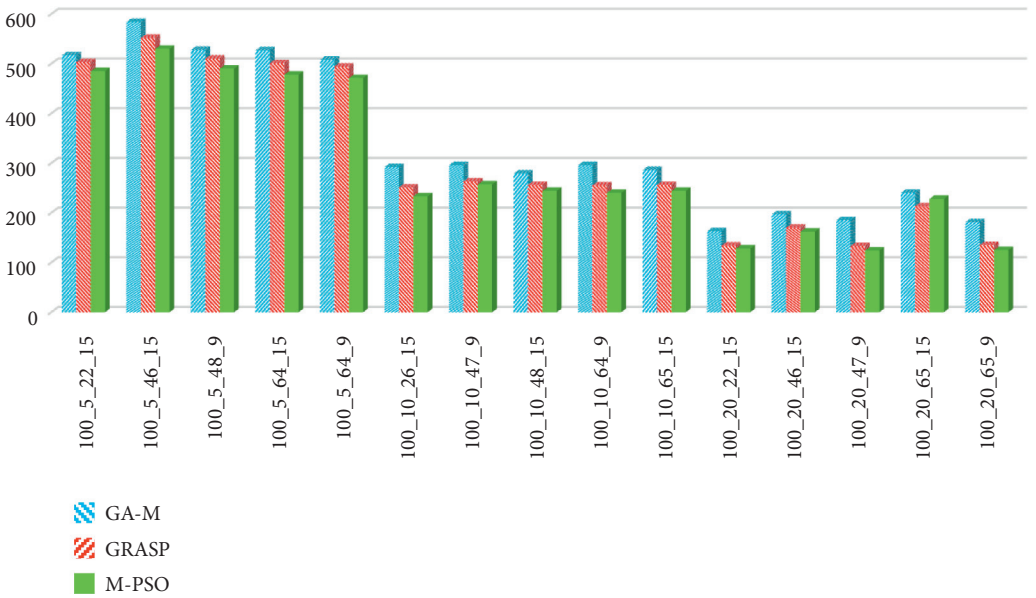

Figure 5: Comparing BEST values between GA-M, GRASP, and M-PSO of 100 tasks projects.

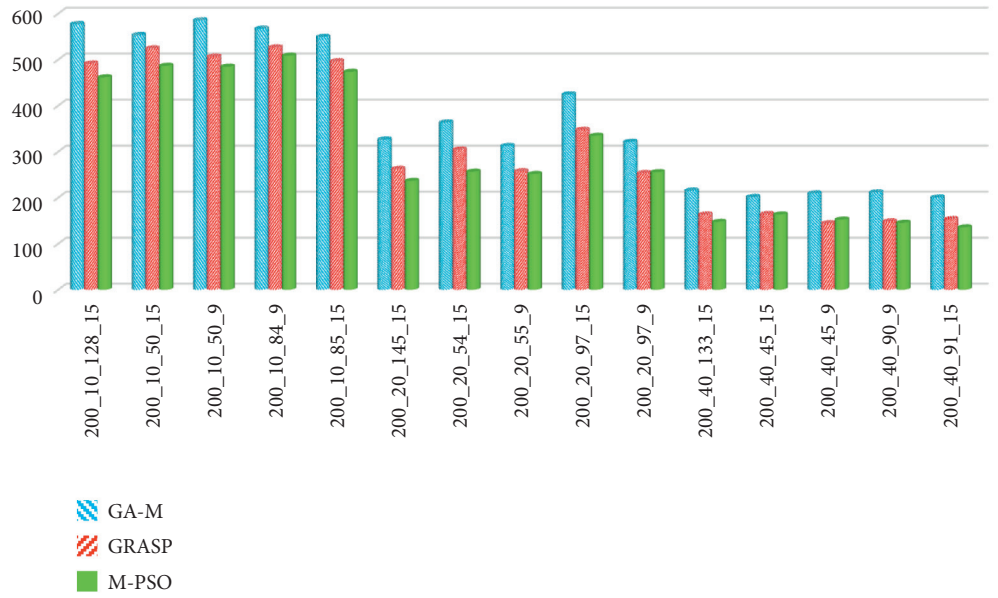

Figure 6: Comparing BEST values between GA-M, GRASP, and M-PSO of 200 tasks projects.

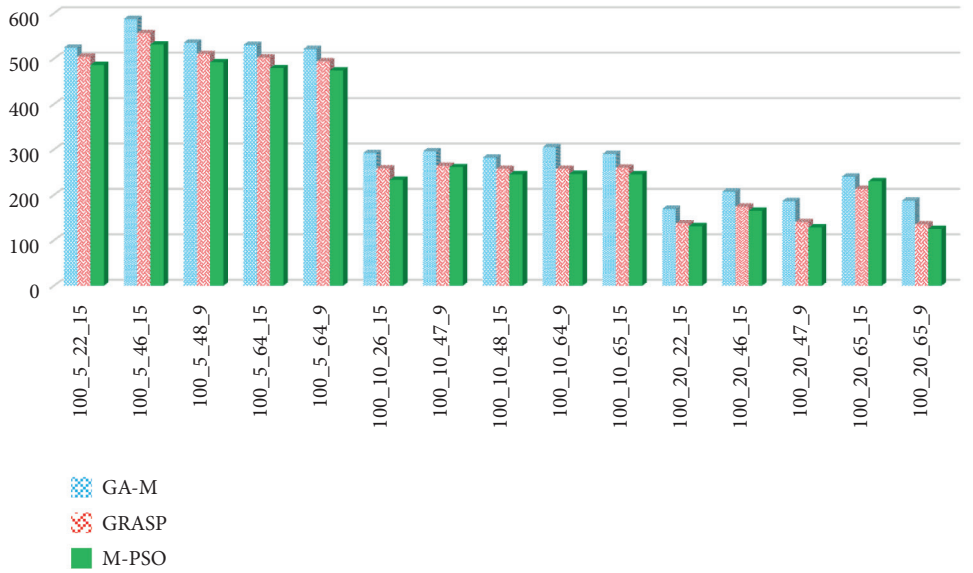

Figure 7: Comparing AVG values between GA-M, GRASP, and M-PSO of 100 tasks projects.

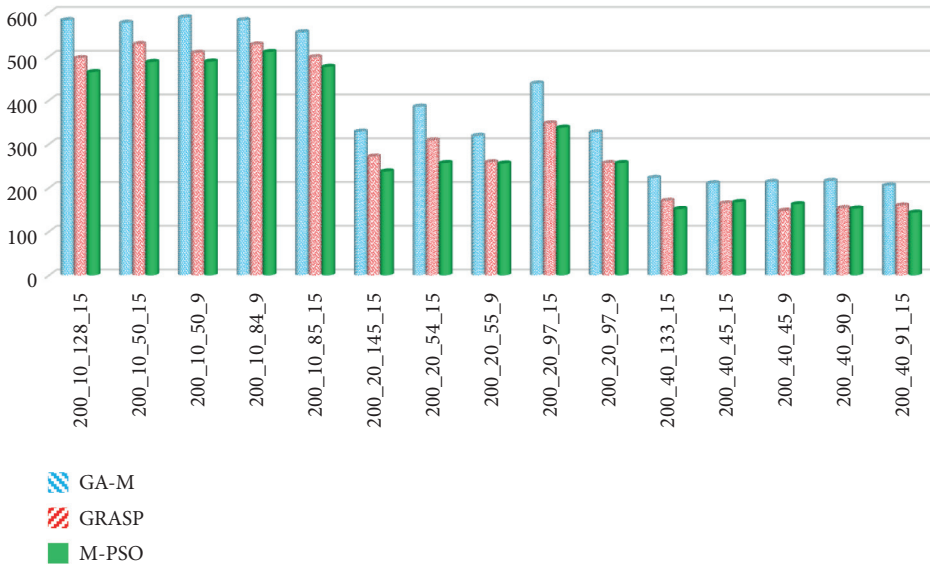

Figure 8: Comparing AVG values between GA-M, GRASP, and M-PSO of 200 tasks projects.

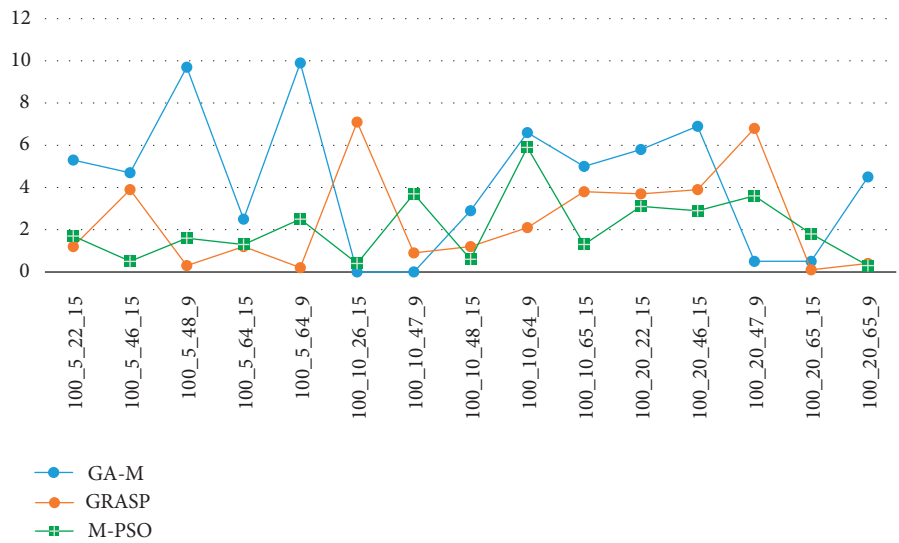

Figure 9: Comparing STD values between GA-M, GRASP, and M-PSO of 100 tasks projects.

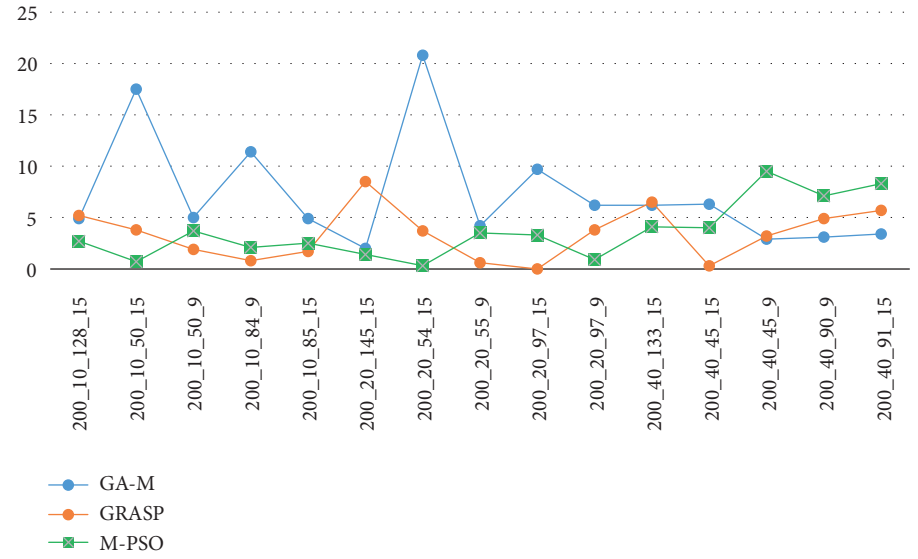

Figure 10: Comparing STD values between GA-M, GRASP, and M-PSO of 200 tasks projects.

in comparison with two competitors are shown in the figures, specifically as follows:

(i) The BEST values present in Figures 5 and 6.

(ii) The AVG values present in Figures 7 and 8.

(iii) The STD values present in Figures 9 and 10.

6. Conclusion and Future Work

In this paper, we have presented the MS-RCPSP problem, a combinatorial optimization problem with many scientific and practical applications. It has described the mathematical model of the problem and proposes a new algorithm to find a feasible solution for the MS-RCPSP problem. The proposed algorithm is M-PSO, which is improved from the PSO algorithm combined with the migration method to escape the local extremes and expand the searching space. The conduction evaluated the effectiveness of our proposed algorithm on the iMOPSE dataset (standard dataset used for MS-RCPSP problem). All experiment results were collected and compared with other algorithms such as GA-M and GRASP. Experimental results show that M-PSO's results are better than previous algorithms, specifically better than GA-M from 6.2% to 32.97% and GRASP from 0.6% to 15.8%.

In the future, the authors will continue to research and improve the algorithm based on other approximation methods, using random moves based on Gauss, Cauchy, etc. to improve the suggested effectiveness of the algorithm.

Data Availability

The paper uses the standard iMOPSE dataset to test the efficiency of the algorithm. This dataset is publicly available at https://imopse.ii.pwr.wroc.pl/ and is free of charge.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] A. Christian, S. Demassey, and E. Néron, *Resource Constrained Project Scheduling: Models, Algorithms, Extensions and Applications*, Wiley, Hoboken. NJ, USA, 2008.

[2] K. Aurangzeb, S. Aslam, M. Alhussein, R. Ali Naqvi, M. Arsalan, and S. I. Haider, "Contrast enhancement of fundus images by employing modified PSO for improving the performance of deep learning models," *IEEE Access*, vol. 9, pp. 47930–47945, 2021.

[3] H. Li and K. Womer, "Stochastic resource-constrained project scheduling and its military applications," *IEEE Transactions on Computers*, vol. 65, no. 12, pp. 3702–3712, 2016.

[4] N. Xiong, A. V. Vasilakos, J. Wu et al., "A self-tuning failure detection scheme for cloud computing service," in *Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium*, Shanghai, China, May 2012.

[5] M. Verma, N. Bhardwaj, N. Bhardwaj, and A. K. Yadav, "Real time efficient scheduling algorithm for load balancing in fog computing environment," *International Journal of Information Technology and Computer Science*, vol. 8, no. 4, pp. 1–10, 2016.

[6] M. Wu, L. Tan, and N. Xiong, "A structure fidelity approach for big data collection in wireless sensor networks," *Sensors*, vol. 15, no. 1, pp. 248–273, 2015.

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

[8] M. Skowroński, P. B. Myszkowski, P. Kwiatek, and M. Adamski, "Tabu search approach for multi–skill resource–constrained project scheduling problem," in *Proceedings of the 2013 Federated Conference on Computer Science and Information Systems*, pp. 153–158, Kraków, Poland, September 2013.

[9] P. B. Myszkowski, M. Laszczyk, I. Nikulin, and M. Skowro, "iMOPSE: a library for bicriteria optimization in multi-skill resource-constrained project scheduling problem," *Soft Computing Journal*, vol. 23, Article ID 32397, 2019.

[10] O. P. Mejia, M. C. Anselmet, C. Artigues, and P. Lopez, "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.

[11] R. Klein, *Scheduling of Resource-Constrained Projects*, Springer Science & Business Media, Berlin, Germany, 2012.

[12] H. D. Quoc, L. N. The, C. N. Doan, and T. P. Thanh, "New effective differential evolution algorithm for the project scheduling problem," in *Proceedings of the 2020 2nd International Conference on Computer Communication and the Internet (ICCCI 2020)*, Nagoya, Japan, June 2020.

[13] S. Sennan, S. Ramasubbareddy, S. Balasubramaniyam, A. Nayyar, M. Abouhawwash, and N. A. Hikal, "T2FL-PSO: type-2 fuzzy logic-based particle swarm optimization algorithm used to maximize the lifetime of internet of things," *IEEE Access*, vol. 9, pp. 63966–63979, 2021.

[14] Y.-Y. Li, J. Lin, and Z.-J. Wang, "Multi-skill resource constrained project scheduling using a multi-objective discrete Jaya algorithm," *Applied Intelligence*, pp. 1–21, 2021.

[15] M. Asadujjaman, H. F. Rahman, R. K. Chakrabortty, and M. J. Ryan, "An immune genetic algorithm for solving NPV-based resource constrained project scheduling problem," *IEEE Access*, vol. 9, pp. 26177–26195, 2021.

[16] M. H. F. Rahman, R. K. Chakrabortty, and M. J. Ryan, "Managing uncertainty and disruptions in resource constrained project scheduling problems: a real-time reactive approach," *IEEE Access*, vol. 9, pp. 45562–45586, 2021.

[17] 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, Article ID 112915, 2020.

[18] H. M. H. Saad, R. K. Chakrabortty, S. Elsayed, and M. J. Ryan, "Quantum-inspired genetic algorithm for resource-constrained project-scheduling," *IEEE Access*, vol. 9, pp. 38488–38502, 2021.

[19] J. L. Xi and H. F. Bai, "A general robot inverse kinematics solution method based on improved PSO algorithm," *IEEE Access*, vol. 9, 2021.

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

[21] R. Rani and R. Garg, "Energy efficient task scheduling using adaptive PSO for cloud computing," *International Journal of Reasoning-Based Intelligent Systems*, vol. 13, no. 2, pp. 50–58, 2021.

[22] B. Song, Z. Wang, and L. Zou, "An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree bezier curve," *Applied Soft Computing*, vol. 100, Article ID 106960, 2021.

[23] P. B. Myszkowski and M. Laszczyk, "Diversity based selection for many-objective evolutionary optimisation problems with constraints," *Information Sciences*, vol. 546, pp. 665–700, 2021.

[24] N. Kleedtke, M. Hua, and S. Pozzi, "Genetic algorithm optimization of tin-copper graded shielding for improved plutonium safeguards measurements," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 988, Article ID 164877, 2021.

[25] 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.

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

[27] X. Li, S. Han, L. Zhao, C. Gong, and X. Liu, "New dandelion algorithm optimizes extreme learning machine for biomedical classification problems," *Computational Intelligence and Neuroscience*, vol. 2017, Article ID 4523754, 13 pages, 2017.

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

[29] R. Kolisch and A. Sprecher, "PSPLIB-a project scheduling problem library: OR software-ORSEP operations research software exchange program," *European Journal of Operational Research*, vol. 96, no. 1, pp. 205–216, 1997.

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

[31] H. Liu, A. Abraham, and C. Grosan, "A novel variable neighborhood particle swarm optimization for multi-objective flexible job-shop scheduling problems," in *Proceedings of the 2nd International Conference on Digital Information Management (ICDIM '07)*, pp. 138–145, Lyon, France, October 2007.

[32] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, Perth, WA, Australia, November 1995.

[33] S. K. Sahana, "Ba-PSO: a balanced PSO to solve multi-objective grid scheduling problem," *Applied Intelligence*, pp. 1–13, 2021.

[34] C. Shang, J. Gao, H. Liu, and F. Liu, "Short-term load forecasting based on PSO-KFCM daily load curve clustering and CNN-LSTM model," *IEEE Access*, vol. 9, pp. 50344–50357, 2021.

[35] O. M. Sedeh, B. Ostadi, and F. Zagia, "A novel hybrid GA-PSO optimization technique for multi-location facility maintenance scheduling problem," *Journal of Building Engineering*, vol. 40, Article ID 102348, 2021.