AdaL-PSO - a new adaptive algorithm for the Multi-Skilled resource-constrained project Scheduling problem

Phan Thanh Toan, Do Van Tuan

Samsung Display Vietnam, Yen Phong Industrial Zone, Bac Ninh, Viet Nam

*Email: ptttoan@hnue.edu.vn

Received: 14 June 2023; Accepted for publication: 4 April 2023

Abstract. MS-RCPSP is a combinatorial optimization problem with many practical applications, this problem has been proven to belong to the NP-hard class, the approach to solving this problem is to use algorithms to find an approximate solution. In this paper, we adopted a new adaptive nonlinear weight update strategy based on fitness value and new neighborhood topology for Particle Swarm Optimization algorithm, thereby helping to prevent PSO from falling into local extremes. The new algorithm is called AdaL-PSO. A numerical analysis is carried out using iMOPSE benchmark dataset and is compared with some other early algorithms. The results presented suggest the prospect of our proposed algorithm.

Keywords: Swarm intelligence, evolutionary computing, optimization methods, resource constrained Project Scheduling, particle Swarm optimization.

Classification numbers: 4.7.1, 4.7.4

1. INTRODUCTION

The resource-constrained projects cheduling problem (RCPSP) has been an important optimization problem over the past few decades with many applications in manufacturing, production planning, resource scheduling in cloud computing, and some other areas [1 - 3]. Over the past few decades, there has been increasing interest in the study of the MS-RCPSP problem, which is an extension of the RCPSP problem. MS-RCPSP is a combinatorial optimization problem that was inspired by a scheduling problem in software development industry, where staff had several different skills, such as programming, data analysis, debugging and so on. In the MS-RCPSP problem, employees have a number of skills and the skill levels vary significantly. Each activity requires a certain skill level, which increases the flexibility of the scheduling problem, but it also makes the problem more complex.

The Resource Constrained Project Scheduling Problem has been proven to belong to NP-hard class, which is one of the most investigated topics in the class of scheduling problems [4].

The paper includes six major sections: some methods for solving the MS-RCPSP problem are described in section 2, where an overview of the particle swarm optimization algorithm is also presented. Particle Swarm Optimization is a metaheuristic algorithm which is based on the
behavior of social animals, this algorithm has been applied to efficiently solve optimization problems. The mathematical formulation of the MS-RCPSP problem and system model will be presented in section 3 of this paper. The proposed algorithm is presented in section 4, which is based on the Particle Swarm Optimization strategy [4]. We also focus on presenting a new neighborhood topology called LNDS (local neighborhood descent search) and inertia weight update strategy based on sigmoid function, these are the two main components of the proposed algorithm. The proposed algorithm is conducted on the iMOPSE benchmark dataset [5], the simulation results are analyzed in detail in section 5 of this paper. We also conducted the algorithm on the factory’s data set collected from TNG, this is the data from a large textile company in Vietnam. Conclusions and directions for future research are presented in section 6 of the paper.

2. RELATED WORK

2.1. Approximation methods for MS-RCPSP problem

Scheduling problem is an issue of great concern to many companies and is applied in many fields [6 - 11]. The MS-RCPSP problem, which we mentioned in this paper, is also proven to belong to the NP-hard class [4, 12, 13]. In addition, MS-RCPSP is multimodal or discontinuous and very challenging to solve with traditional optimization strategies. In the past few years, many research works and different approaches have been proposed to solve the MS-RCPSP problem [14 - 17]. The approach based on metaheuristic algorithms is the most common, the authors in papers [18 - 22] used the Genetic Algorithm (GA), Ant Colony Optimization (ACO) and Particle Swarm Optimization.


A hybrid algorithm was created by Myszkowski et al. [5, 13], this is a combination of two methods: Differential Evolution method and the Greedy algorithm. The goal of the authors is to minimize the makespan and costs, and the authors also build a new benchmark dataset for the MS-RCPSP problem called iMopse [5], this dataset is created based on real-world instances of the problem.

Mejia et al. [3] proposed a new variant of the MS-RCPSP problem to optimize the schedule of research activities in a nuclear laboratory. In the article [25], the authors proposed two new evolutionary algorithms for the MS-RCPSP problem with the deterioration effect and financial constraints. Hosseinian [26] proposed an extended model of the MS-RCPSP problem, this model has a dual objective, the author’s purpose is to simultaneously minimize the makespan and total cost of the project.

Resource-Constrained Project Scheduling Problem and its extension, Multi-Skill Resource-Constrained Project Scheduling Problem, have become important research topics over the last few decades. Many authors have proposed different methods for this problem [27 - 30].

2.2 Mathematical formulation of PSO
PSO (Particle Swarm Optimization) is one of the most well-known population-based stochastic optimization algorithms which was proposed by Kennedy and Eberhart [32]. Velocity and position are updated as follows:

\[
\begin{align*}
    v_{i}^{t+1} &= w^{t} \times v_{i}^{t} + c_{1} \times rand_{1} \times (p_{\text{best}_{i}}^{t} - x_{i}^{t}) + c_{2} \times rand_{2} \times (p_{g\text{best}}^{t} - x_{i}^{t}) \\
    x_{i}^{t+1} &= x_{i}^{t} + v_{i}^{t}
\end{align*}
\]

(1)

(2)

2.3. Topological neighborhood for PSO

Particles have been studied in two general types of neighborhoods: 1) global best and 2) local best. There are many topological neighborhoods [33]. The formula for updating the position vector is

\[
v_{i}^{t+1} = w^{t} \times v_{i}^{t} + c_{1} \times rand_{1} \times (p_{\text{best}_{i}}^{t} - x_{i}^{t}) + c_{2} \times rand_{2} \times (p_{l\text{best}_{i}}^{t} - x_{i}^{t})
\]

(3)

where \(p_{l\text{best}_{i}}\) is the local best position of particle \(i\) with the best fitness value among its neighbors.

As shown in Figure 1, three neighborhood topologies are widely used.

![Neighborhood topologies](image)

*Figure 1. Neighborhood topologies.*

3. MATHEMATICAL MODEL AND DEFINITIONS

3.1. Classical RCPSP

MS-RCPSP is an extension of the RCPSP problem, where each resource has multiple skills and each activity requires one or more specific skills to be executed. The description of the classical RCPSP would be introduced as follows:

(i) We represent each project by a directed non-cyclical graph \(G(V, E)\), where each node on the graph represents a task and the arcs represent the ordinal relationship between the tasks. The arc \((u, v) \in E\) shows that task \(u\) must be completed before task \(v\) begins (Figure 2).

(ii) We add two empty tasks to the project. A task is placed at the beginning of the project, which is the predecessor task of all other tasks in the project and the second empty task is placed at the end of the project, which is the successor of all other tasks.

(iii) The duration of a specific task is the length of the task in execution time, from its start time to its finish time.

(iv) Tasks must not be stopped during execution, until it is done.

(v) Each task requires some resources to execute.

3.2. Mathematical model of the MS-RCPSP
AdaL-PSO - a new adaptive algorithm for the Multi-Skilled resource-constrained project …

The MS-RCPSP is an important scheduling problem that is applied to many real-world problems [5, 13, 31]. In MS-RCPSP each task requires some skills at a specified level to be executed, whereas resources have multiple skills at different levels.

![Diagram](image)

*Figure 2. The relationship between tasks in a workflow.*

Some notations of the MS-RPSP are given as follows:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre</td>
<td>The set of tasks that need to be completed before task i can be executed</td>
</tr>
<tr>
<td>SK</td>
<td>The set of all resource’s skills S': the subset of skills owned by the resource i, S' ⊆ S</td>
</tr>
<tr>
<td>S_i</td>
<td>The skill i</td>
</tr>
<tr>
<td>d_j</td>
<td>The duration of task j</td>
</tr>
<tr>
<td>t_{jk}</td>
<td>The time to complete task j by resource k</td>
</tr>
<tr>
<td>Re</td>
<td>The resources used to execute tasks of the project</td>
</tr>
<tr>
<td>Re^k_i</td>
<td>The subset of the resources which can perform task k; Re^k_i ⊆ Re</td>
</tr>
<tr>
<td>Re_i</td>
<td>The resource i</td>
</tr>
<tr>
<td>W</td>
<td>The project tasks to be done</td>
</tr>
<tr>
<td>W_k</td>
<td>The subset of the task which can be executed by resource k, W_k ⊆ W</td>
</tr>
<tr>
<td>W_i</td>
<td>The task i</td>
</tr>
<tr>
<td>p_i</td>
<td>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</td>
</tr>
<tr>
<td>B_k, E_k</td>
<td>The begin time and end time of task k</td>
</tr>
<tr>
<td>A_{u,v}</td>
<td>The variable to identify resource v running task u at time t (1: yes, 0: no)</td>
</tr>
<tr>
<td>h_i</td>
<td>The skill level i</td>
</tr>
<tr>
<td>h_{S_q}</td>
<td>Level of skill q of a resource</td>
</tr>
<tr>
<td>g_i</td>
<td>Skill type i</td>
</tr>
<tr>
<td>g_{S_q}</td>
<td>Type of skill q of a resource</td>
</tr>
<tr>
<td>Mk</td>
<td>Makespan of the schedule</td>
</tr>
<tr>
<td>P</td>
<td>The feasible solution</td>
</tr>
<tr>
<td>P_{all}</td>
<td>The set of all solutions</td>
</tr>
<tr>
<td>f(P)</td>
<td>The function to calculate the makespan of P solution</td>
</tr>
<tr>
<td>N</td>
<td>Number of tasks to execute</td>
</tr>
<tr>
<td>M</td>
<td>Number of tasks available</td>
</tr>
</tbody>
</table>
The Real-RCPSP problem could be stated as follows:

\[ f(P) \rightarrow \min \]

where:

\[ f(P) = \max_{W_j \in W} \{ E_j \} - \min_{W_k \in W} \{ B_k \} \]

subjected to the following constraints:

1. \( S_k \neq \emptyset \) \( \forall R_e \in R \) \( A_1 \) (6)
2. \( t_{jk} \geq 0 \) \( \forall W_j \in W, \forall R_e \in R \) \( A_2 \) (7)
3. \( E_j \geq 0 \) \( \forall W_j \in W \) \( A_3 \) (8)
4. \( E_i \leq E_j - t_{ij} \) \( \forall W_j \in W, j \neq I, W_i \in C_j \) \( A_4 \) (9)
5. \( \forall W_i \in W^k \exists S_q \in S^k : g_{S_q} = g_{r_i} \text{ and } h_{S_q} \geq h_{r_i} \) \( A_5 \) (10)
6. \( \forall L_k \in L, \forall q \in m : \sum_{i=1}^{n} A_{i,k}^q \leq 1 \) \( A_6 \) (11)
7. \( \forall W_j \in W \exists ! q \in [0, m] \exists L_k \in L : A_{j,k}^q = 1 \text{; where } A_{j,k}^q \in \{0; 1\} \) \( A_7 \) (12)

\[ t_{ik} \geq t_{il} \text{ where } h_{r_k} \leq h_{r_i} \forall (r^k, r^i) \in \{S^k \times S^l\} \] \( A_8 \) (13)

Example 1

In Figure 3, \( S_{i,j} \) denotes \( S_i \) skill type and it has \( j \) skill level. The resources of the project have skill types: \( S_1 \) and \( S_2 \), and each of them can have a skill level from 1 to 3. The resource constraints for the tasks are presented as follows:

Example 1

Consider a project with tasks as: \( J = \{J_1, J_2, J_3, J_4, J_5, J_6, J_7, J_8, J_9, J_{10}\} \) and project resources \( RS = \{R_{e1}, R_{e2}, R_{e3}\} \), as show in Figure 2.

4. PROPOSED ALGORITHM

4.1. Schedule Representation

The MS-RCPSP problem consists of two important components: resource allocation and task sequencing. In this paper we use the Resource-only scheme method to represent the solution of the problem. Each solution is represented as an vector \( S = \{R_{e1}, R_{e2}, \ldots R_{eN}\} \); where \( N \) is the number of task.

Example 1

Consider a project with tasks as: \( J = \{J_1, J_2, J_3, J_4, J_5, J_6, J_7, J_8, J_9, J_{10}\} \) and project resources \( RS = \{R_{e1}, R_{e2}, R_{e3}\} \), as show in Figure 2.
Our goal is to find a solution with a minimum project completion time (makespan) and satisfy two constraints: the order of execution of tasks and the skill level of the resources.

The duration of the tasks are given in Table 1

<table>
<thead>
<tr>
<th>Task</th>
<th>J_1</th>
<th>J_2</th>
<th>J_3</th>
<th>J_4</th>
<th>J_5</th>
<th>J_6</th>
<th>J_7</th>
<th>J_8</th>
<th>J_9</th>
<th>J_{10}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

Assume that the resources are assigned to the tasks as shown in the following table:

<table>
<thead>
<tr>
<th>Task</th>
<th>J_1</th>
<th>J_2</th>
<th>J_3</th>
<th>J_4</th>
<th>J_5</th>
<th>J_6</th>
<th>J_7</th>
<th>J_8</th>
<th>J_9</th>
<th>J_{10}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource</td>
<td>Re_1</td>
<td>Re_2</td>
<td>Re_3</td>
<td>Re_3</td>
<td>Re_1</td>
<td>Re_2</td>
<td>Re_1</td>
<td>Re_1</td>
<td>Re_1</td>
<td>Re_3</td>
</tr>
</tbody>
</table>

A feasible schedule for the project is demonstrated in Figure 2, where it can be seen that the makespan is 17.

Figure 4. A feasible schedule.

The above schedule is demonstrated in Table 2, where resource 1 performs tasks J_1, J_2, J_6, J_8, and J_9; resource 2 is assigned to execute tasks J_3 and J_7; resource 3 deals with the rest tasks. According to this scheme, the project completion time (makespan) is 17.

4.2. Proposed algorithm AdaL-PSO

In the PSO algorithm, two parameters C_1 and C_2 provide local/global exploration, while the inertia weight parameter determines the relative rate of convergence. Many studies have shown that the algorithm convergence speed is significantly reduced when using a fixed inertia value [34]. This is a consequence of excessive momentum in the population, so the large step size exceeds the best search regions. In this paper, we proposed an adaptive inertia weight update strategy based on the fitness value. When the fitness value tends to increase, a large inertia weight is needed in order for the design space to be searched thoroughly. As fitness value decreases, which means that the most promising area of the search space has been discovered and the speed of convergence tends to slow down, the inertia weight should be reduced, in order for the momentum of the particles to decrease, allowing them to concentrate in the best search areas.

To accomplish the above strategy, we proposed a fitness-dependent value of the inertia weight. A commonly used inertia update rule is the non-linear decreasing based on sigmoid function, calculated by the formula as below:

\[
\theta = -10 + \frac{(f_t - f_{\min}) \times 20}{f_{\max} - f_{\min}}
\]  

(14)
where: $f^t$ is the fitness value at the $t^{th}$ iteration; $f_{\text{min}}$: the minimum value of the fitness function up to time $t$; $f_{\text{max}}$: the maximum value of the fitness function up to time $t$.

The inertia weight parameter was updated by the following formula:

$$w^t = \text{sigmoid}(\theta) = \frac{1}{1+e^{-\theta}} \quad (15)$$

Sigmoid functions have the domain of all real numbers, with a return value between 0 and 1, it increases monotonically and continuously everywhere. For values less than -10, the function’s value is almost equal to zero. For values greater than 10, the function’s values are almost equal to one.

From (1), (2), (3) and (13), (14) we have the formula to update the velocity as follows:

$$v_{i}^{t+1} = \frac{1}{1+e^{\theta}} \times v_{i}^{t} + C_1 \times rand_1 \times (p_{\text{best}i}^{t} - x_{i}^{t}) + C_2 \times rand_2 \times (p_{\text{best}i}^{t} - x_{i}^{t}) \quad (16)$$

$p_{\text{best}i}^{t}$ is calculated based on the best experience of the $i^{th}$ particle and its "social" neighbors. The procedure for determining $p_{\text{best}i}^{t}$ may include a component that takes into account the current position $X_i$. In this paper, we propose a new algorithm to calculate $p_{\text{best}i}^{t}$, that is implemented by a local neighborhood descent search method (LNDS). The goal of this method is to find a new vector position $X'_i$, situated in the neighborhood of the vector $X_i$ and having a better fitness value.

LNDS starts from current position $X_i$ and makes changes in $m$ iterations ($m$ is the number of tasks). At each iteration, only one position $d, d = 1, ..., m$ is changed to reduce the fitness function. Hence, a sequence of adjacent position vectors is generated as in the following scheme:

$$X_i \rightarrow X_1^1 \rightarrow X_2^2 \rightarrow \cdots \rightarrow X_m^m \rightarrow X_i' \quad (17)$$

**Function LNDS($X_i$)**

```
function LNDS(X_i)
(1)Begin
(2)for d = 1 to m do
(3)for k = 1 to n do
(4) $R_k^{old} \leftarrow X_i^d$
(5) flag \leftarrow CheckResource(R_k, Re, Sk)
(6) if flag Then
(7) Begin
(8) $X_i^d \leftarrow R_k$
(9) val \leftarrow EvalFitness(X_i)
(10) if (val < fitness_i) then
(11) fitness_i \leftarrow val
//restore to its initial value
(12) else $X_i^d \leftarrow R_k^{old}$
```

---

**Figure 5. The sigmoid function**
AdaL-PSO - a new adaptive algorithm for the Multi-Skilled resource-constrained project …

(13) End
(14) return $X_i$, fitness
(15) End

The function CheckResource ($R_{Ek}$, $R_e$, $Sk$) returns true, if $R_{Ek}$ satisfies the skill type and skill level constraints for task $J_d$. The constraints are depicted in Figure 3. If the fitness, is not improved, the value in position #d,$X_i^d$ is restored to its initial value. When the fitness value decreases, the complemented value at position #d is kept, and the LNDS will try to diminish this value once more by complementing the next resource $R_{Ek+1}$. The algorithm continues with the next positions $X_{i+1}$, if the current position $X_i$ is a local optimum, then the LNDS will not improve the value fitness_i.

The proposed dynamic adaptive local particle swarm optimization (AdaL-PSO) algorithm can be summarized as follows:

**AdaL-PSO Algorithm**

(1) Begin
(2) Load and Valid iMopse dataset
(3) $t\leftarrow 0$
(4) Size $\leftarrow$ size of the population
(5) for $i = 1$ to Size
    $v_i \leftarrow$ random vector; $x_i \leftarrow$ random vector
(6) end for
(7) $f(t)\leftarrow$ calculate fitness, makespan
(8) fitness.Add($f(t)$)
(9) while (Stop criterion)
(10) calculate $\theta$ according to (14)
(11) for $i = 1$ to Size
(12) Calculate $P_{best_i}$
(13) $P_{best_i} \leftarrow LNDS(x_i)$
(14) update velocity $v_i \leftarrow$ according to formula (16)
(15) update vector position $x_i \leftarrow$ according to formula (2)
(16) end for
(17) Update $P_{gbest}$
(18) $gbest \leftarrow$ Makespan ($P_{gbest}$)
(19) $t \leftarrow t+1$
(20) end while
(21) return makespan
(22) End

Where:

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

(ii) *Size:* number of solutions in the population

(iii) *N:* number of tasks
As shown in line 10, the values of inertia weight are dynamically changing and are determined based on the fitness function and line 13, the local best position of the $i$th particle is calculated by the LNDS function.

At the $t$-th iteration, the position vector of particle $i$-th is updated based on the best position of this particle up to time $t$ and the local best of this particle, the $p_{tbest}$ is calculated by the local neighborhood descent search method. This method only explores different neighborhoods randomly, it also often runs faster than previous local search algorithms and gives good quality solutions.

We see, each subpopulation will converge to a different optimal value in the search space due to the presence of local attractive features. Thus, niches from the overall population don't have any knowledge of domain problem. However, subpopulation has only few particles, the search capabilities of this model are quite poor, especially in the large search spaces and with many extreme points. To overcome this disadvantage, we proposed a new method to improve the search process in an local best model, which is called local neighborhood descent search (LNDS).

The inertial weights in the proposed algorithm are updated according to the dynamic adaptive method based on the sigmoid function, using formula (15). The value of the inertial weights is updated according to the value of the objective function. This method allows timely adjustment of particle speed, gets out of a locally extrema solution and improves convergence speed.

5. EXPERIMENT AND ANALYSIS

5.1. Simulation datasets

We use two datasets: benchmark dataset iMOPSE and real dataset TNG [35] to evaluate the performance of the proposed algorithm. The above two datasets are also used for GreedyDO and GA algorithms, the experimental results of the proposed algorithm are compared with the experimental results of the two algorithms GreedyDO and GA.

Table 3. iMOPSE dataset.

<table>
<thead>
<tr>
<th>Dataset instance</th>
<th>Tasks</th>
<th>Resources</th>
<th>Precedence Relations</th>
<th>Skills</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>100</td>
<td>5</td>
<td>48</td>
<td>9</td>
</tr>
<tr>
<td>Q2</td>
<td>100</td>
<td>5</td>
<td>64</td>
<td>15</td>
</tr>
<tr>
<td>Q3</td>
<td>100</td>
<td>5</td>
<td>64</td>
<td>9</td>
</tr>
<tr>
<td>Q4</td>
<td>100</td>
<td>10</td>
<td>64</td>
<td>9</td>
</tr>
<tr>
<td>Q5</td>
<td>100</td>
<td>10</td>
<td>65</td>
<td>15</td>
</tr>
<tr>
<td>Q6</td>
<td>100</td>
<td>20</td>
<td>65</td>
<td>15</td>
</tr>
<tr>
<td>Q7</td>
<td>100</td>
<td>20</td>
<td>65</td>
<td>9</td>
</tr>
<tr>
<td>Q8</td>
<td>200</td>
<td>10</td>
<td>84</td>
<td>9</td>
</tr>
<tr>
<td>Q9</td>
<td>200</td>
<td>10</td>
<td>85</td>
<td>15</td>
</tr>
<tr>
<td>Q10</td>
<td>200</td>
<td>20</td>
<td>55</td>
<td>9</td>
</tr>
<tr>
<td>Q11</td>
<td>200</td>
<td>20</td>
<td>97</td>
<td>15</td>
</tr>
<tr>
<td>Q12</td>
<td>200</td>
<td>20</td>
<td>97</td>
<td>9</td>
</tr>
<tr>
<td>Q13</td>
<td>200</td>
<td>40</td>
<td>45</td>
<td>9</td>
</tr>
<tr>
<td>Q14</td>
<td>200</td>
<td>40</td>
<td>90</td>
<td>9</td>
</tr>
<tr>
<td>Q15</td>
<td>200</td>
<td>40</td>
<td>91</td>
<td>15</td>
</tr>
</tbody>
</table>
The benchmark iMOPSE is a dataset for investigating optimization algorithm, which consists of 15 groups and is presented in Table 3.

The TNG dataset, which is a digitized data set from the industrial sewing production of the TNG company [38], is presented in Table 4.

<table>
<thead>
<tr>
<th>Dataset instance</th>
<th>Number of tasks</th>
<th>Number of resources</th>
<th>Precedence Relations</th>
<th>Skill Levels</th>
<th>Project Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>TNG_1</td>
<td>71</td>
<td>37</td>
<td>1026</td>
<td>6</td>
<td>409</td>
</tr>
<tr>
<td>TNG_2</td>
<td>71</td>
<td>39</td>
<td>1026</td>
<td>6</td>
<td>325</td>
</tr>
<tr>
<td>TNG_3</td>
<td>71</td>
<td>41</td>
<td>1026</td>
<td>6</td>
<td>296</td>
</tr>
<tr>
<td>TNG_4</td>
<td>71</td>
<td>45</td>
<td>1026</td>
<td>6</td>
<td>392</td>
</tr>
<tr>
<td>TNG_5</td>
<td>137</td>
<td>37</td>
<td>1894</td>
<td>6</td>
<td>1174</td>
</tr>
<tr>
<td>TNG_6</td>
<td>137</td>
<td>39</td>
<td>1894</td>
<td>6</td>
<td>1052</td>
</tr>
<tr>
<td>TNG_7</td>
<td>137</td>
<td>41</td>
<td>1894</td>
<td>6</td>
<td>871</td>
</tr>
<tr>
<td>TNG_8</td>
<td>137</td>
<td>45</td>
<td>1894</td>
<td>6</td>
<td>996</td>
</tr>
</tbody>
</table>

5.2. Parameters and system settings

The simulation is conducted using: Simulation environment: Matlab ver. 2014

Simulation tool: the performance of previous algorithms such as GA is fairly evaluated using GARunner [13], the simulation tool provided by the author of those algorithms themself.

(i) Input data: 15 iMOPSE instances and TNG that are described in Table 3 and Table 4
(ii) Number of solutions in the population $N_p$: 100
(iii) Number of generations $N_g$: 40,000
(iv) Since all of the considered algorithms are metaheuristic algorithms, so they do not guarantee the best solution. To evaluate the algorithms, we run on each instance 30 times to get the average value, the standard deviation value, and the best value.

5.3. Performance analysis

5.3.1. Performance on iMOPSE dataset

Table 5. The makespan of the best solutions found by GreedyDO and GA.

<table>
<thead>
<tr>
<th>Dataset instance</th>
<th>GreedyDO</th>
<th>GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>779</td>
<td>528</td>
</tr>
<tr>
<td>Q2</td>
<td>640</td>
<td>527</td>
</tr>
<tr>
<td>Q3</td>
<td>597</td>
<td>508</td>
</tr>
<tr>
<td>Q4</td>
<td>533</td>
<td>296</td>
</tr>
<tr>
<td>Q5</td>
<td>426</td>
<td>286</td>
</tr>
<tr>
<td>Q6</td>
<td>310</td>
<td>240</td>
</tr>
<tr>
<td>Q7</td>
<td>408</td>
<td>181</td>
</tr>
<tr>
<td>Q8</td>
<td>999</td>
<td>567</td>
</tr>
<tr>
<td>Q9</td>
<td>706</td>
<td>549</td>
</tr>
<tr>
<td>Q10</td>
<td>999</td>
<td>312</td>
</tr>
<tr>
<td>Q11</td>
<td>680</td>
<td>424</td>
</tr>
</tbody>
</table>
Table 6. The average, the best, and the standard deviation of makespan obtained from GA and AdaL-PSO.

<table>
<thead>
<tr>
<th>Dataset instance</th>
<th>GA</th>
<th>AdaL-PSO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Avg</td>
<td>Best</td>
</tr>
<tr>
<td>Q1</td>
<td>535</td>
<td>528</td>
</tr>
<tr>
<td>Q2</td>
<td>530</td>
<td>527</td>
</tr>
<tr>
<td>Q3</td>
<td>521</td>
<td>508</td>
</tr>
<tr>
<td>Q4</td>
<td>305</td>
<td>296</td>
</tr>
<tr>
<td>Q5</td>
<td>290</td>
<td>286</td>
</tr>
<tr>
<td>Q6</td>
<td>240</td>
<td>240</td>
</tr>
<tr>
<td>Q7</td>
<td>187</td>
<td>181</td>
</tr>
<tr>
<td>Q8</td>
<td>583</td>
<td>567</td>
</tr>
<tr>
<td>Q9</td>
<td>555</td>
<td>549</td>
</tr>
<tr>
<td>Q10</td>
<td>318</td>
<td>312</td>
</tr>
<tr>
<td>Q11</td>
<td>438</td>
<td>424</td>
</tr>
<tr>
<td>Q12</td>
<td>326</td>
<td>321</td>
</tr>
<tr>
<td>Q13</td>
<td>213</td>
<td>209</td>
</tr>
<tr>
<td>Q14</td>
<td>215</td>
<td>211</td>
</tr>
<tr>
<td>Q15</td>
<td>205</td>
<td>200</td>
</tr>
</tbody>
</table>

The experimental results are presented in Tables 5 and 6, where the makespan value is calculated in hour unit. The data in the above tables are the makespan values of the best solution found by GreedyDO, GA, and the proposed algorithm (AdaL-PSO). Table 5 shows that the makespans found by the GA algorithm are always better than the makespans found by the GreedyDO algorithm.

The comparison results between the average value, the best value and the standard deviation of makespan found by the GA algorithm and the proposed algorithm AdaL-PSO are presented in Table 6.

The average makespan time for all the four algorithms is computed, the results in Table 5 and Table 6 show that:

(i) Based on 15 instances of iMOPSE experimental dataset, the mean of makespan found by AdaL-PO, GA and Greedy algorithms is: 679.7 (hours); 357.2 (hours); and 325.5 (hours), respectively. We see that the solution of AdaL-PSO algorithm is better than GA by 8 % and better than Greedy by 52 %.

(ii) Based on the mean column in Table 5, we see that the solution found by AdaL-PSO is better than GA from 0.7 % to 24.8 % and better than Greedy from 17 % to 77.5 %.

(iii) In the proposed AdaL-PSO algorithm, we use adaptive inertia weight updated strategy based on the fitness value, which controls the local search and convergence to the global optimum solution. Therefore, the AdaL-PSO algorithm has higher convergence speed and stability. The data in column std of Table 5 demonstrates this problem. The average of the std (standard deviation) column of the AdaL-PSO algorithm is 1.3, while the average of the
std (standard deviation) column of the GA algorithm is 5.6, which proves that the stability of the AdaL-PSO algorithm is better than the GA.

5.3.2. Performance on TNG dataset

The makespan values of the best solution for GreedyDO, GA and AdaL-PSO algorithms are presented in Table 7. The duration of actual projects in TNG company is shown in the TNG column of Table 7.

<table>
<thead>
<tr>
<th>Dataset instance</th>
<th>TNG</th>
<th>GreedyDO</th>
<th>GA</th>
<th>AdaL-PSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>TNG_1</td>
<td>409</td>
<td>236</td>
<td>201</td>
<td>163</td>
</tr>
<tr>
<td>TNG_2</td>
<td>325</td>
<td>243</td>
<td>198</td>
<td>167</td>
</tr>
<tr>
<td>TNG_3</td>
<td>296</td>
<td>258</td>
<td>212</td>
<td>165</td>
</tr>
<tr>
<td>TNG_4</td>
<td>392</td>
<td>248</td>
<td>176</td>
<td>165</td>
</tr>
<tr>
<td>TNG_5</td>
<td>1174</td>
<td>972</td>
<td>751</td>
<td>702</td>
</tr>
<tr>
<td>TNG_6</td>
<td>1052</td>
<td>963</td>
<td>791</td>
<td>710</td>
</tr>
<tr>
<td>TNG_7</td>
<td>871</td>
<td>834</td>
<td>810</td>
<td>723</td>
</tr>
<tr>
<td>TNG_8</td>
<td>996</td>
<td>906</td>
<td>720</td>
<td>671</td>
</tr>
</tbody>
</table>

From the experimental data in Table 7, we see that the schedule generated by the AdaL-PSO algorithm is always better than the schedule generated by the GA and Greedy algorithms. The data in Table 7 and Figure 5 also show that the schedule generated by the evolutionary algorithms is always smaller than the actual schedule at the TNG company.

![Figure 5. Comparison of makespans between AdaL-PSO, GA and factory TNG’s solution.](image)

Averaged over 8 instances of the TNG dataset, the makespan values generated by the GreedyDO, GA and AdaL-PSO algorithms are: 582.5 (hours); 482.375 (hours); and 433.25 (hours), respectively. Therefore, on average, the schedule generated by the AdaL-PSO algorithm is about 10% smaller than the schedule generated by GA and about 25% smaller than the schedule generated by the GreedyDO algorithm.

From this experimental result, it has been proved that the solution found by the proposed algorithm is always better than the previous algorithms. With the combination of neighbor
architecture and adaptive inertial weight update strategy, the AdaL-PSO algorithm always meets both the important properties of evolutionary algorithms that are fast convergence and averting getting trapped in local extrema.

The data in Table 7 also shows that the schedule generated by the AdaL-PSO algorithm is 15% to 60% smaller than the actual schedule at TNG company.

6. CONCLUSION AND FUTURE WORK

This paper defines and introduces the formulation of MS-RCPSP, which is one of the famous scheduling problems and is applied in many fields such as science, industry, and life. The mathematical model of the MS-RCPSP problem is also presented in section 3 of this paper, then we have proposed a new scheduling algorithm based on Particle Swarm Optimization method, called AdaL-PSO.

Resource-Constrained Project Scheduling Problems and Multi-skilled Resource-Constrained Project Scheduling Problem (RCPSP and MS-RCPSP) have been essential topics of study over the last years. MS-RCPSP is suitable in projects with multi-skilled human resources or multi-functional machines. Based on Particle Swarm Optimization, we have devised a new metaheuristic algorithm that has the ability to outperform all the previous algorithms. We built a new function, which updates the inertial weight dynamically based on fitness value. This is the key component that creates the strength of the proposed algorithm.

To evaluate the efficiency of the proposed algorithm, we conducted many experiments based on simulated and real datasets. The experimental results are compared with the previous algorithms. The simulation result shows that our developed algorithm is more effective than existing algorithms. AdaL-PSO not only achieves better quality solutions but also converges to the global extremum faster than previous algorithms.

In the future, we wish to investigate how to improve the AdaL-PSO algorithm to solve bigger instances within a reasonable makespan.

CRediT authorship contribution statement. Author 1: Toan Phan Thanh is the principle investigator in this research and he is the proponent of the new algorithm. Author 2: Do Van Tuan helped with algorithm design and formalism and the write up of the manuscript.

Declaration of competing interest. The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

REFERENCES


25. Hosseiniyan, Amir H., Baradaran V. - P-GWO and MOFA: two new algorithms for the MSRCPSP with the deterioration effect and financial constraints (case study of a gas treating company), Applied Intelligence 50 (2020) 2151-2176. DOI: 10.1007/s10489-020-01663-x


35. TNG Investment and Trading Joint Stock Company, 434/1 Bac Kan street - Thai Nguyen city, Viet Nam, Website http://www.tng.vn