Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2023.027146

ARTICLE

# Rules Mining-Based Gene Expression Programming for the Multi-Skill Resource Constrained Project Scheduling Problem

**Min Hu[1,2,3], Zhimin Chen[4], Yuan Xia[4], Liping Zhang[1,2,3,*] and Qiuhua Tang[1,2,3]**

[1]Key Laboratory of Metallurgical Equipment and Control Technology, Ministry of Education, Wuhan University of Science and Technology, Wuhan, 430081, China

[2]Hubei Key Laboratory of Mechanical Transmission and Manufacturing Engineering, Wuhan University of Science and Technology, Wuhan, 430081, China

[3]Precision Manufacturing Institute, Wuhan University of Science and Technology, Wuhan, 430081, China

[4]China Ship Development and Design Center, Wuhan, 430060, China

*Corresponding Author: Liping Zhang. Email: zhangliping@wust.edu.cn

Received: 16 October 2022 Accepted: 25 November 2022

## ABSTRACT

The multi-skill resource-constrained project scheduling problem (MS-RCPSP) is a significant management science problem that extends from the resource-constrained project scheduling problem (RCPSP) and is integrated with a real project and production environment. To solve MS-RCPSP, it is an efficient method to use dispatching rules combined with a parallel scheduling mechanism to generate a scheduling scheme. This paper proposes an improved gene expression programming (IGEP) approach to explore newly dispatching rules that can broadly solve MS-RCPSP. A new backward traversal decoding mechanism, and several neighborhood operators are applied in IGEP. The backward traversal decoding mechanism dramatically reduces the space complexity in the decoding process, and improves the algorithm's performance. Several neighborhood operators improve the exploration of the potential search space. The experiment takes the intelligent multi-objective project scheduling environment (iMOPSE) benchmark dataset as the training set and testing set of IGEP. Ten newly dispatching rules are discovered and extracted by IGEP, and eight out of ten are superior to other typical dispatching rules.

## KEYWORDS

Project scheduling; multi-skill; gene expression programming; dispatching rules

## 1 Introduction

Resource-constrained project scheduling problem (RCPSP) is an important problem in project management, which is widely used in the modern management system and industrial production management, such as production planning, inventory management, and transportation management. Since the 1960s, many researchers have been studying on RCPSP, all precedence relationships of tasks in this problem can be described by the activity-on-arrow network [1]. Meanwhile, resource constraints restrict the total resource consumption every time during the duration. This problem minimizes project completion time, cost, or resource equilibrium by sequencing tasks and assigning resources.

With the advent of the intelligent era, more and more practitioners are engaged in the Internet Technology (IT) industry. To improve their job competitiveness, most IT engineers are proficient in various skills, such as web front-end development, web back-end development, database construction, etc. Due to the different preferences and emphases, these skills and experience may also vary for each engineer, and this is a kind of common multi-skills human resource issue.

MS-RCPSP has been proven to be an NP-hard problem. Exact algorithms can often get the optimal solution relatively quickly for solving small-scale problems. But it generally takes a long time to obtain feasible solutions for large-scale problems, which is unacceptable to managers. When solving large-scale problems, meta-heuristic algorithms can often get near-optimal solutions in an acceptable time. However, the results of most meta-heuristic algorithms are highly uncertain.

Nowadays, rule-based heuristic approaches are usually used to generate scheduling schemes with priority to deal with resource-constrained problems [2]. Gene expression programming (GEP) algorithm is a search algorithm based on biological evolution mechanisms. Via fitting the linear relationship of attribute values, GEP algorithm can obtain dispatching rules combining multiple attribute values. A calculated value can be accurately obtained to establish the priority of the task.

This paper uses a backward traversal gene expression programming algorithm (IGEP) to generate newly dispatching rules. Some of the contributions from this paper are listed below:

1) IGEP improves the decoding process of the basic GEP algorithm and adds several efficient neighborhood search operators. It can dramatically improve the efficiency of exploring the high-quality dispatching rules. The newly dispatching rules have superior performance and can easily be applied to the real project and production environment.
2) The dispatching rules combined with the parallel scheduling mechanism solves MS-RCPSP to obtain a scheduling scheme that minimizes project completion time.
3) The backward traversal decoding method can significantly reduce the space complexity of the IGEP algorithm.
4) The IGEP algorithm with the merit of artificial intelligence and unsupervised learning is effective for MS-RCPSP.

The rest of this paper can be described as follows: Section 2 introduces some related literature review. Section 3 describes the motivation for this research. Section 4 introduces the multi-skill resource constrained project scheduling problem in detail. And a mathematical model is given. In Section 5, the design of the IGEP algorithm is introduced. And there is a systematic description of the encoding and decoding process, evolution strategy, and fitness function. Section 6 presents the results and comparative analysis. In this section, the data of the benchmark case set iMPOSE data set is used to show the calculation results of the dispatching rule obtained by training and compare them with several typical rules. The calculation results of dispatching rules based on the benchmark case set are compared and analyzed. Section 7 concludes the paper.

## 2 Literature Review

Multi-skilled resource-constrained project scheduling problem (MS-RCPSP) is a kind of problem that is very worthy of in-depth study, which is derived from the real-world production situation when considering human resources or machines with multiple capabilities. However, compared to the MS-RCPSP with the RCPSP, there is a lack of sufficient datasets to research the problem. For this reason, Myszkowski et al. [3] created a set of iMOPSE datasets for multi-skill resource-constrained project

scheduling problems in 2015, and which later researchers used as a benchmark dataset to study this problem. After that, Myszkowski et al. [4] tried to solve MS-RCPSP with a greedy randomized adaptive search procedure (GRASP).

There are many meta-heuristic algorithms those can to solve multi-skill resource-constrained project scheduling problems, such as genetic algorithm (GA) [5–7], ant colony optimization (ACO) [8–10], taboo search (TS) [11–13], simulated annealing (SA) [14,15], particle swarm optimization (PSO) [5,16–18], migrating birds optimization (MBO) [19,20], etc. Machine learning has an excellent performance in solving real-world problems. Wen et al. [21–23] applied convolutional neural networks to fault classification. Cheng et al. [24] used Q-learning to solve multi-objective hybrid workshop scheduling, and achieved outstanding results. Zhou et al. [25] proposed a hybrid genetic algorithm to solve the multi-execution mode and multi-resource constrained project scheduling problem, and applied it to the internal scheduling process of ocean engineering construction.

Myszkowski et al. [26] introduced the hybrid ant colony algorithm to solve MS-RCPSP. A new multi-skill multi-mode resource constrained project scheduling problem with three objectives is studied by Maghsoudlou et al. [27] in 2016. A variable neighborhood search approach for the resource-constrained had proposed by Cui et al. [28] in 2020. Li et al. [29] presented a multi-objective discrete Jaya (MODJaya) algorithm to address the MS-RCPSP in 2021. In the same year, Zhu et al. [30] proposed an efficient decomposition-based multi-objective genetic programming hyper-heuristic (MOGP-HH/D) approach for the multi-skill resource constrained project scheduling problem (MS-RCPSP) with the objectives of minimizing the makespan and the total cost simultaneously. However, due to its complexity, meta-heuristic algorithms are rarely used to solve real MS-RCPSP problem.

According to the parallel scheduling mechanism, a large number of dispatching rules have been proposed, such as the shortest processing time (SPT), the minimum slack time (MSLK), and the earliest job deadline (ODD) [31], etc. These dispatching rules can often obtain some scheduling schemes in the shortest time. Although these scheduling schemes are not optimal, most of them can bring near-optimal, and are easily used by managers. Almeida et al. [32] applied dispatching rules to solving MS-RCPSP in 2016. They proved that using activity dispatching rules to solve project scheduling problems can successfully adapt to multi-skill resource-constrained project scheduling problems.

At the end of the 20th century, Portuguese biologist Ferreira [33] first proposed the concept of gene expression programming (GEP) based on genetic algorithm (GA) and genetic programming (GP) [34]. As soon as it was proposed, it attracted widespread attention. More and more scholars and scientific researchers have invested a lot of energy in this field. That is widely used in practical areas, such as mathematics, physics, biology, chemistry, military industry, microelectronics, and economics. Many related theories and research results have emerged.

In 2013, Peng et al. [35] proposed an improved gene expression programming algorithm for solving symbolic regression problems. In 2017, Zhong et al. [36] summarized the development of gene expression programming in coding design, evolutionary mechanism design, adaptive design, co-evolutionary design, parallel design, theoretical research, and application in recent years. The research result of applying GEP in supervised machine learning shows that it is very suitable for solving the problems of classification and complex functional relationship discovery. An optimal scheduling scheme can be obtained by the priority of each task, which is a problem of complex functional relationship discovery. Therefore, using GEP to train meta-heuristic dispatching rules for solving MS-RCPSP is an efficient option.

Zhang et al. [37–41] conducted a more in-depth exploration of the dispatching rule of the production scheduling, and provided a lot of research materials. Dispatching rules are also widely used in the real MS-RCPSP problems. Zhang et al. [42] proposed an eGEP algorithm, which optimizes the scheduling process of the job shop by extracting attributes. That significantly impacts energy consumption, reducing the workshop's energy consumption. Zhang et al. [43] proposed a hybrid feature gene expression programming algorithm, which applies non-destructive reverse engineering to the chip with bypass detection data.

## 3 Motivations

### *3.1 Solve Practical Problems*

In the last two decades, the software industry has developed rapidly. There are numerous Internet companies that rise and fail every year. How to scientifically manage the personnel allocation in the process of Internet project development has become a decisive point in the current fierce competition. Compared with the uncertain factors of resources allocation management in the real economy, personnel allocation management for Internet companies has natural advantages. The time cost is only related to the technology and experience of employees, and it is not necessary to consider the project delay caused by uncontrollable factors due to the source of raw materials, and the labor hours of each task in the software development process can be accurately estimated. Therefore, accurately assigning tasks to employees with different skills in a team can greatly improve efficiency, and this problem belongs to MS-RCPSP.

Recently, a company plan to develop a project for customer. The project can divide into 5 tasks, they can be described as follows:

$T_1$: Database construction.

$T_2$: The function of background business logic.

$T_3$: Web terminal development.

$T_4$: Data upload and storage function.

$T_5$: Mobile terminal development.

There is predecessor relationship among the five tasks, such as task 4 cannot begin before task 2 is completed. These five tasks will be completed independently by 3 employees, and the employees performing them must meet the skill needs of the task. Relevant information about all tasks and employees is shown in Table 1.

**Table 1:** A project example of the MS-RCPSP

| Resource ID | Skills, proficiency level | Task ID | Duration | Skills, proficiency level | Predecessors |
|---|---|---|---|---|---|
| 1 | $S_2$, 1 and $S_3$, 2 | 1 | 2 | $S_2$, 1 | - |
| 2 | $S_1$, 1 and $S_3$, 1 | 2 | 1 | $S_1$, 2 | - |
| 3 | $S_1$, 2 and $S_2$, 2 | 3 | 3 | $S_3$ 2 | - |
| - | - | 4 | 3 | $S_1$, 1 | $T_2$ |
| - | - | 5 | 1 | $S_3$, 1 | - |

Fig. 1 shows a feasible solution for the project, which will vary in the completion time of the whole project due to the different employees be assigned to task 5. A good dispatching scheme is of great value.

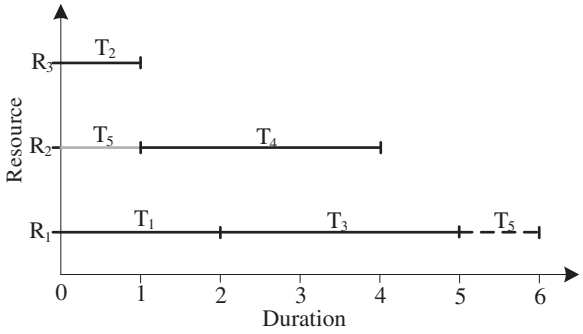


**Figure 1:** The Gantt chart of a feasible solution

### *3.2 Dispatching Rules Have Targeted*

Dispatching rules have been widely used to solve resource constraints project scheduling problems. However, it is found that few researchers have applied dispatching rules to solve MS-RCPSP. Table 2 shows the $C_{max}$ values obtained by four typical dispatching rules for six small-scale MS-RCPSP instances. Each instance has differences in the number of resources, predecessor relationships, and types of skills. The four typical dispatching rules are LPT (longest processing time), SPT (shortest processing time), LRCP (longest of the remaining critical path), and LLFT (latest finishing time).

**Table 2:** $C_{max}$ values of four typical dispatching rules

| No. | Instance | LPT | SPT | LRCP | LLFT |
|---|---|---|---|---|---|
| 1 | 10_3_5_3 | **109** | **109** | 130 | 128 |
| 2 | 10_5_8_5 | **84** | **84** | **84** | **84** |
| 3 | 10_7_10_7 | **104** | **104** | **104** | **104** |
| 4 | 15_3_5_3 | **230** | **230** | **230** | **230** |
| 5 | 15_6_10_6 | 104 | 114 | **102** | **102** |
| 6 | 15_9_12_9 | 117 | 97 | **94** | **94** |

As shown in Table 2, these four dispatching rules obtain the best solution for the instances 10_5_8_5, 10_7_10_7, and 15_3_5_3. But LPT and SPT obtain the best solution for the instance 10_3_5_3. LRCP and LLFT obtain the best solution for the instance 15_6_10_6 and 15_9_12_9. This may be caused by the different inherent properties of each dispatching rule. According to no free lunch, each dispatching rule cannot play the best performance for every instance. Therefore, it is a valuable research direction to explore the newly dispatching rule for this problem.

## 4 Multi-skill Resource Constrained Project Scheduling Problem

### 4.1 Problem Description

The Multi-skill resource-constrained project scheduling problem can be described as a project which contains a task set $T=\{0, 1, 2, \ldots, J, J+1\}$. Each task $j$ ($j=0, 1, 2, \ldots, J, J+1$) has a starting time $s_j$ and duration $d_j$. Where tasks 0 and $J+1$ are virtual tasks. Virtual tasks with duration $d_j=0$ represent the first and the last tasks in precedence constraints.

Each task $j$ should be assigned once and can start to be assigned when all its predecessor tasks are finished. These precedence relationships of all tasks are presented as a priority relationship matrix $H$. Each value $P_{ij}$ ($i, j=0, 1, 2, \ldots, J, J+1$) in the matrix $H$ is a 0–1 value. If task $i$ is the predecessor of task $j$, $P_{ij}$ is set to 1; otherwise, $P_{ij}$ is set to 0. There are $K$ resources $\{1, 2, \ldots, K\}$ with $N$ skills $\{1, 2, \ldots, N\}$. $G_{kn}$ ($G_{kn}=0, 1, \ldots, Q$) represents the proficiency level of resource $k$ at skill $n$. A large $G_{kn}$ value indicates the high proficient level of resource $k$ at skill $n$. Each task $j$ consumes a resource $k$ and must satisfy the inequality $G_{kn} \geqslant \eta_{jn}$, $\eta_{jn}$ ($\eta_{jn}=0, 1, \ldots, Q$) denotes the minimum proficiency level of skill $n$ of resource k when task $j$ is assigned to resource $k$. Moreover, each resource can be assigned at most one task at a time. This paper aims to minimize the project completion time $C_{\max}$.

### 4.2 Problem Formulation

The notations of the MS-RCPSP are shown in Table 3. In this paper, the objective is to minimize the project completion time $C_{\max}$, which can be expressed as follows:

$$\min \quad C_{\max} = \max f_j, \forall j \tag{1}$$

**Table 3:** The notations for the MS-RCPSP

| | |
|---|---|
| $i, j$ | Index for tasks where $i, j \in \{0, 1, \ldots, J, J+1\}$ |
| $k$ | Index for resources where $k \in \{1, 2, \ldots, K\}$ |
| $t$ | Index for time $t \in [0, +\infty)$ |
| $n$ | Index for skills where $n \in \{1, 2, \ldots, N\}$ |
| $Q$ | The maximum number of the skill's proficiency level |
| $M$ | The infinite value |
| $d_j$ | Duration of task $j$ |
| $s_j$ | Starting time of task $j$ |
| $f_j$ | Finishing time of task $j$ |
| $\eta_{jn}$ | The required proficiency level of skill $n$ for task $j$, $\eta_{jn} \in \{0, 1, \ldots, G\}$ |
| $G_{kn}$ | The level that resource $k$ master skill $n$, $G_{kn} \in \{0, 1, \ldots, G\}$ |
| $P_{ij}$ | The value to express whether task $i$ is the predecessor of task $j$, $P_{ij} \in \{0, 1\}$ |
| $x_{jk}$ | Binary variable to determine whether task $j$ is assigned resource $k$, $x_{jk} \in \{0, 1\}$ |
| $y_{jkt}$ | Binary variable to determine whether task $j$ is assigned resource $k$ at time $t$, $y_{jkt} \in \{0, 1\}$ |

Temporal constraints: Constraint (2) ensures that each task is continuous and cannot be interrupted. Constraints (3) and (4) ensure that the executed time ranges of each task $j$ in $s_j$ to $f_j$, $s_j$, and $f_j$ are starting time and finishing time of task $j$, respectively. Constraint (5) ensures values of the starting time and duration for each task $j$ are a nonnegative number.

$$f_j = s_j + d_j, \forall j \tag{2}$$

$$y_{jkt}t \geq s_j, \forall j, k, t \tag{3}$$

$$y_{jkt}t \leq f_j, \forall j, k, t \tag{4}$$

$$s_j \geq 0, d_j \geq 0, \forall j \tag{5}$$

Predecessor constraints: If task $i$ is the predecessor task of task $j$, the starting time $s_j$ of task $j$ must not be less than the finishing time $f_i$ of task $i$. It can be expressed as follows:

$$P_{ij}\left(s_j - f_i + \frac{1}{M}\right) > 0, \forall i, j \tag{6}$$

Resource constraints: Constraint (7) expresses the proficient level of resource $k$ at skill $n$ must be the required proficiency level of skill $n$ for task $j$. Constraint (8) ensures each task $j$ only needs one resource. Constraint (9) expresses each resource $k$ can be assigned at most one task at a time.

$$\sum_k x_{jk} G_{kn} \geq \eta_{jn}, \forall j, n, k \tag{7}$$

$$\sum_k x_{jk} = 1, \forall j, k \tag{8}$$

$$\sum_j y_{jkt} \leq 1, \forall j, k, t \tag{9}$$

Binary variable constraints: There are two binary variables $x_{jk}$ and $y_{jkt}$. Their ranges of values are shown in Eqs. (10) and (11), respectively.

$$x_{jk} = \begin{cases} 1, & \text{task } j \text{ is assigned to resource } k \\ 0, & \text{other} \end{cases} \quad \forall j, k \tag{10}$$

$$y_{jkt} = \begin{cases} 1, & \text{task } j \text{ is assigned to resource } k \text{ at time } t \\ 0, & \text{other} \end{cases} \quad \forall j, k, t \tag{11}$$

## 5 Rules-Mining Framework Using the IGEP Algorithm

GEP is an efficient evolutionary algorithm that inherits the advantages of genetic algorithm (GA) and genetic programming (GP). It has the simplicity of the 'fixed-length linear string' of GA, and the searchability of the 'dynamic tree structure' of GP. Like GA and GP, the GEP training process includes evolution operators, such as initialization, fitness evaluation, selection, mutation, and crossover (recombination). In addition, GEP has its unique transposition operator, and the rules mining perturbation operator is also designed and added to the IGEP algorithm in this paper. The fitness evaluation process of GEP is similar to that of GP, which is based on the value obtained by converting a binary tree into an ORF expression and then calculating it. The goal of GEP is to find the individual (chromosome) with the highest fitness value.

This paper applies the IGEP algorithm to explore dispatching rules with high fitness to solve MS-RCPSP. The basic idea of the IGEP algorithm is to generate an initial population. Each individual in the population represents a dispatching rule. Then iterate through importing the training set data to

evolve the initial population. Each iteration is an evolutionary selection process. Finally, a dispatching rule with high fitness can be obtained.

### *5.1 Encoding and Decoding*

#### *5.1.1 Encoding*

The heuristic algorithm based on dispatching rules has the characteristics of simplicity, practicality, and high computing efficiency. Therefore, it is often used to solve resource-constrained project scheduling problems. In the IGEP algorithm, the dispatching rule is used to select a task from the task set $\pi_k{}^*$, and the set $\pi_k{}^*$ contains all the tasks that can be assigned at the current moment.

This paper extract attributes from Myszkowski et al.'s research [3] and several typical dispatching rules [32,44] to form attribute set $V$. The elements in set $V$ can be described as follows:

1) Task duration ($pt_{(j)}$), it represents the duration of the task $j$ ($j = 1, 2, \ldots, J$), that is, $pt_{(j)} = d_j$.
2) The number of immediate predecessor tasks ($pn_{(j)}$), it represents the number of immediate predecessors of task $j$.
3) The number of immediate successor tasks ($sn_{(j)}$), it represents the number of immediate successors of task $j$.
4) The total number of predecessor tasks ($pa_{(j)}$), it represents the number of total predecessors of task $j$.
5) The total number of successor tasks ($sa_{(j)}$), it represents the number of total successors of task $j$.
6) Minimum proficiency level requirements ($sg_{(j)}$), it represents the minimum proficiency level required for task $j$.
7) The remaining critical path length ($cpl_{(j)}$), it represents the path length with the longest duration in subsequent task paths of task $j$ in the task network diagram.
8) The number of tasks on the remaining critical path ($cpn_{(j)}$), it represents the number of tasks on the path with the longest duration in the subsequent task paths of task $j$ in the task network diagram.
9) The latest finishing time ($lf_{(j)}$), it represents the latest finishing time of task $j$.
10) Slack time ($tw_{(j)}$), it represents the difference between the latest finishing time $lf_{(j)}$ and the earliest finishing time $ef_{(j)}$ of task $j$, that is, $tw_{(j)} = lf_{(j)} - (es_{(j)} + d_j)$.
11) The number of optional resources ($rn_{(j)}$), it represents the number of resources that can be assigned to task $j$, that is, $rn(j) = \sum_{k \in R} \left(G_{kn} \geq n_{jn}\right), \forall j, k, n$.

The population of the IGEP algorithm is represented as a collection of genotype-encoded individuals, that is, the chromosome with multiple linear strings. This paper defines the attribute set containing $V = \{pt, pn, sn, pa, sa, sg, cpl, cpn, lf, tw, rn\}$ the above 11 attributes. The function set $F = \{+, -, *, /, Q, \max, \min\}$ contains 7 basic function operators.

The encoding process generates an initial population $A^*$ with *popsize* chromosomes. Where *popsize* is the population size, each chromosome is composed of two parts: the head and the tail. The gene of the head is randomly selected from the function set $F$ and the attribute set $V$. But the gene of the tail only be chosen randomly from the attribute set $V$. The length of each chromosome $l$ is determined by

the head length $h$ and the maximum number of child nodes $m_{\max}$ of the function operator.

$$l = h \times (m_{\max} + 1) - 1 \tag{12}$$

Fig. 2 shows an initial population $A^*$, whose population size is $popsize = n$ and the length of head $h = 5$.

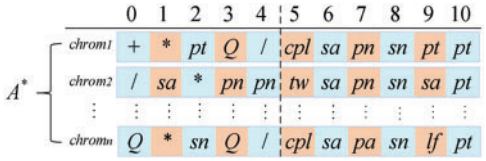


**Figure 2:** An initial population

### 5.1.2 Decoding

A feasible solution is usually represented as a genotype-encoded individual. Each chromosome has a corresponding expression tree and ORF expression. As shown in Fig. 3, the chromosome is $chrom = \{+, *, pt, Q, /, cpl, sa, pn, sn, pt, pt\}$. Its corresponding expression tree is shown as the binary tree, and its ORF expression is $f = \sqrt{cpl} \times (sa/pn) + pt$. The time complexity is $O_{(h)}$, and the space complexity is $(h \times m_{\max} + 1) \times m_{\max} = S_{(h)}$.

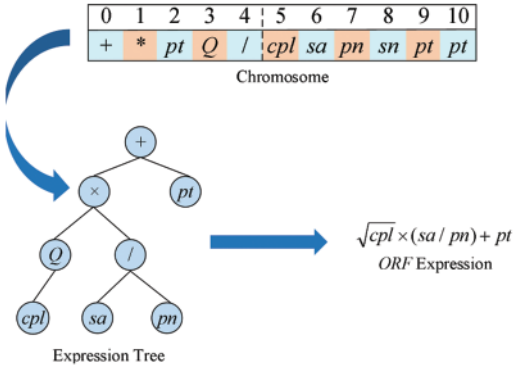


**Figure 3:** Individual 3 manifestations of basic GEP

In order to reduce the complexity, a backward traversal decoding mechanism is proposed in IGEP. The backward traversal decoding method needs to read the gene of chromosomes one by one. But different from the basic GEP decoding traversal tree building, the backward traversal decoding places the elements in a two-level list before reading the gene. All elements of each sub-list in the list correspond one-to-one with all elements of each depth of the expression tree. The calculation process of backward traversal decoding adopts the calculation-deletion method, that is, the child nodes are deleted when a root node is calculated.

The total time complexity $O_{(h)}{}^{*}$ of the backward traversal decoding mechanism equals the total time complexity $O_{(h)}$ of the basic GEP decoding process. However, because the elements in the list gradually decrease, the space complexity $S_{(h)}{}^{*}$ is much smaller than $S_{(h)}$.

Taking a chromosome *chrom* as an example, its length of *chrom* is $l$, and its length of the head is $h$. $i$ ($i = 0, 1, 2, \ldots, l-1$) represents the $i$-th gene of *chrom*. $E^c$ and $E^m$ are two lists containing contain the same amount of sub-list that stored loci-related information of the effective length $L_{\max}$ of *chrom*. The index value of each sub-list corresponds to the tree depth. $P^c_{(j,k)} = chrom_{(i)}$ and $P^m_{(j,k)} = m_i$ express $chrom_{(i)}$ and $m_i$ are the $k$-th element in the $j$-th sub-list of list $E^c$ and $E^m$ respectively, which can be expressed as position information of the gene $chrom_{(i)}$, that is, the position in the expression tree is the $k$-th element with a tree depth of $j$, and the number of child nodes is $m_i$.

The decoding process of *chrom* can be shown in the pseudo-code of **Procedure 1**, which adopted the breadth-first approach.

**Procedure 1** IGEP algorithm decoding process

```
\\The process of location matching
i ← 0, j ← 1, L_max ← 1, K_(1) ← 1
P^c_(1,1) ← chrom_(0), P^m_(1,1) ← m_0
if m_0 ≠ 0 then
    L_max ← 1 + m_0
    K_(j+1) ← m_i
    while K_(j+1) ≠ 0 do
        L_max ← L_max + K_(j+1)
        j ← j + 1
        K_(j+1) ← 0, k ← 0
        i ← i + 1, k ← k + 1
        while i < L_max do
            P^c_(j,k) ← chrom_(i), P^m_(j,k) ← m_i
            K_(j+1) ← K_(j+1) + m_i
            i ← i + 1, k ← k + 1
        end while
    end while
end if
\\The process of calculating fitness value
while j > 0 do
    count1 = 0, count2 = 0
    if count1 ≤ K_(j) then
        if P^c_(j,k) in T then
            P^c_(j,count1) ← f_(P^c_(j,count1))(P^c_(j+1,0), ..., P^c_(j+1,count2))
            remove P^c_(j+1,0), ..., P^c_(j+1,count2)
            count2 ← P^m_(j,count1)
        end if
        count ← count1 + 1
    else
        j ← j − 1
    end if
end while
End Procedure
```

The decoding procedure of the IGEP algorithm is shown in **Procedure 1**. $K_{(j+1)}$ represents the number of elements with a depth of $j+1$ in the expression tree. *count*1 is the count of the depth of the current element in the calculation process. *count*2 is a count used to match the expression tree position of each function's child node. When the function value of a root node is calculated each time, the child node corresponding to the root node is also deleted. The next depth's first element corresponds to each root node's first root node in the current depth. $f_{(P^C{}_{(j,count1)})}$ is the function expression corresponding to the root node. ($P^c_{(j+1,\,0)}, \ldots, P^c_{(j+1,\,count2)}$) is the parameter value set of all branch nodes corresponding

to $f_{(P^C_{(j,count1)})}$. The calculation process of function value is a process of gradually evaluating from the maximum depth to the minimum depth of the root node, then replacing the original function symbol of the node with the function value. It is used as the input parameter value of the function expression corresponding to a root node of the previous depth. Finally, the value of the initial root node $P^c_{(1,1)}$ is equal to the value calculated by the ORF expression of *chrom* and is used as the priority value of the task.

The decoding process has the prototype of the building expression tree. It is only necessary to substitute each element $P^c_{(j,k)}$ in the traversed list $E^c$ into the $k$-th node of the tree depth $j$. Connecting nodes of adjacent depths according to the value of $P^m_{(j,k)}$, an expression tree can be drawn.

### *5.2 Evolutionary Approach*

The main idea of the IGEP algorithm is as follows. First, the initial population is randomly generated. Then, the population evolved. Finally, an optimal chromosome is obtained, that is, an optimal dispatching rule. In the iterative process, the IGEP algorithm adopts several evolution operators: selection and replication, perform mutation operation, IS-transposition, RIS-transposition, one-point crossover, and two-point crossover [36]. In addition, the rule mining perturbation mechanism is added to avoid falling into the local optimum.

*S*1: selection and replication. The IGEP algorithm uses the roulette selection method. Based on the fitness values of chromosomes in population $A$, a new population $A_{new}$ with the same scale as population $A$ is generated. Chromosome with high fitness values in $A$ is more likely to appear in $A_{new}$. Chromosomes with high fitness values usually appear multiple times in $A_{new}$ to replace chromosomes with low fitness values in population $A$.

*S*2: rule mining disturbance operation. According to the preset rule mining perturbation preset value *d_value,* the rule mining perturbation mechanism is triggered if the historical optimal individual $chrom^*_{best}$ does not change within *d_value* iterations. The perturbed chromosome is selected by a certain perturbation probability. The rule mining perturbation mechanism can effectively break the local optimum and enrich the gene types in population $A$.

*S*3: perform mutation operation. The mutation operation is a very common evolutionary operator in the intelligent evolutionary algorithm. The mutation operation refers to selecting a chromosome from population $A$ by a certain mutation probability. A gene $chrom_{(a)}$ is randomly selected and replaced by a random element in function set $F$ or attribute set $V$. If $chrom_{(a)}$ is a tail gene, the only element in the attribute set $V$ can be selected for replacement.

*S*4 and *S*5: IS-transposition and RIS-transposition. Both an insertion sequence transposition (IS-transposition) and a root insertion sequence transposition (RIS-transposition) select *chrom* by a certain probability. Then two gene positions $a$ and $b$ are randomly selected, and a head gene position $c$ is randomly selected. The gene fragment $\{chrom_{(a)}, chrom_{(a+1)}, \ldots, chrom_{(b)}\}$ are copied and inserted into the front of gene position $c$. Meanwhile, all the head genes after position $c$ move backward in turn. If the gene's position exceeds the head, the gene should be discarded. The difference between IS-transposition and RIS-transposition lies in the position of $c$. The head gene position $c$ of the IS-transposition in any position in the head except the first position.

*S*6 and *S*7: one-point crossover and two-point crossover. The crossover operator refers to selecting two parents (*chrom*1 and *chrom*2) from population $A$. In the one-point crossover operation, a position $a$ is chosen randomly. Then two gene fragments $\{chrom1_{(0)}, chrom1_{(1)}, \ldots, chrom1_{(a)}\}$ of *chrom*1 and $\{chrom2_{(0)}, chrom2_{(0)}, \ldots, chrom2_{(a)}\}$ of *chrom*2 are swapped. For the two-point crossover operation,

two positions $a$ and $b$ are randomly selected. Then two gene fragments $\{chrom1_{(a)}, chrom1_{(a+1)}, \ldots, chrom1_{(b)}\}$ of $chrom1$ and $\{chrom2_{(a)}, chrom2_{(a+1)}, \ldots, chrom2_{(b)}\}$ of $chrom2$ are swapped. Two new chromosomes $chrom1_{new}$ and $chrom2_{new}$ will be obtained after a one-point crossover or two-point crossover. The parent chromosomes $chrom1$ and $chrom2$ will be replaced with $chrom1_{new}$ and $chrom2_{new}$.

### 5.3 Evaluation Function

The objective is to obtain the minimum project completion time $\min C_{\max}$ of MS-RCPSP. During the fitness evaluation process, the project completion time of all instances is calculated as the fitness value $f_i$ of chromosome $i$ in population $A$. The fitness value $f_i$ uses the relative deviation calculation method and is calculated as follows:

$$f_i = \sum_{j=1}^{C_i} \left( M - \left| \frac{C_{(i,j)} - T_{(j)}}{T_{(j)}} \times 100 \right| \right) \tag{13}$$

In Eq. (13), $M$ refers to the upper limit of the fitness value. $C_{(i,j)}$ refers to the $C_{\max}$ value of the instance $j$ in the $i$-th chromosome. Because the lower bound of $C_{\max}$ cannot be obtained, an unsupervised learning method is used. In Eq. (13), $T_{(j)}$ is a variable that represents the historical optimal project completion time of instance $j$. It is used to evaluate the fitness value of each individual in population $A$.

### 5.4 Feasible Solution Generation

Combined dispatching rules with the parallel scheduling mechanism, a fixed scheduling scheme will be obtained for each instance. Therefore, the specific scheduling process in IGEP does not need to compile the string encoding. The priority value of each task is determined by the dispatching rule and its attributes. Then, a scheduling scheme and the $C_{\max}$ value are obtained.

**Procedure 2** is the pseudo-code of the scheduling process for the multi-skill resource-constrained project scheduling problem. The scheduling process adopts a parallel scheduling mechanism and divides the entire scheduling process into $g$ stages. The starting time of the first stage is set to $t_{s(1)} = 0$. The starting time of each stage is equal to the finishing time of the previous stage, that is, $t_{s(g)} = t_{f(g-1)}$.

Firstly, according to the predecessor constraints and resource constraints, all tasks that can be allocated are selected from the unallocated task set $\pi$, , and stored in the optional task set $\pi^*$ of the current stage. If task $j$ from $\pi^*$ can be executed by the idle resource $k$, task $j$ is stored in the set $\pi_k{}^*(k = 1, 2, \ldots, K)$ when $G_{kn} \geq n_{jn}$. Then, if task $j$ is in $\pi_k{}^*$ with the highest priority value, the idle resources $k$ should be assigned to task $j$. It should be noted that each task can be assigned to exactly one resource and can only be assigned once. Finally, the finishing time $t_{f(g)}$ of each stage is determined by the completion time of the earliest completed task $j$ in each stage, where $t_{f(g)} = s_j + d_j$. Multiple tasks may be completed simultaneously. The completion time $C_{\max}$ is the finishing time of the last stage. A feasible scheduling scheme $Tn$ can be obtained.

Taking the example $10 * 5 * 7 * 5$ as an example, the project resource task relationship matrix is shown in Fig. 4. From Fig. 4, it can be seen which resources can meet the skill requirements of the task and which tasks can be assigned to the resources. For example, these resources selected by task 1 are $\{R_1, R_2, R_3, R_5\}$. Resource 3 can only execute task 3. The Gantt chart can be obtained as shown in Fig. 5.

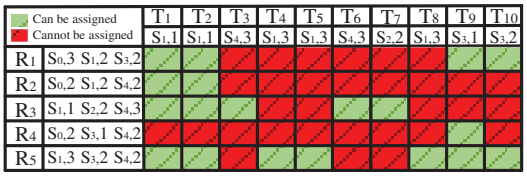


**Figure 4:** The skill matrix of the instance 10 * 5 * 7 * 5

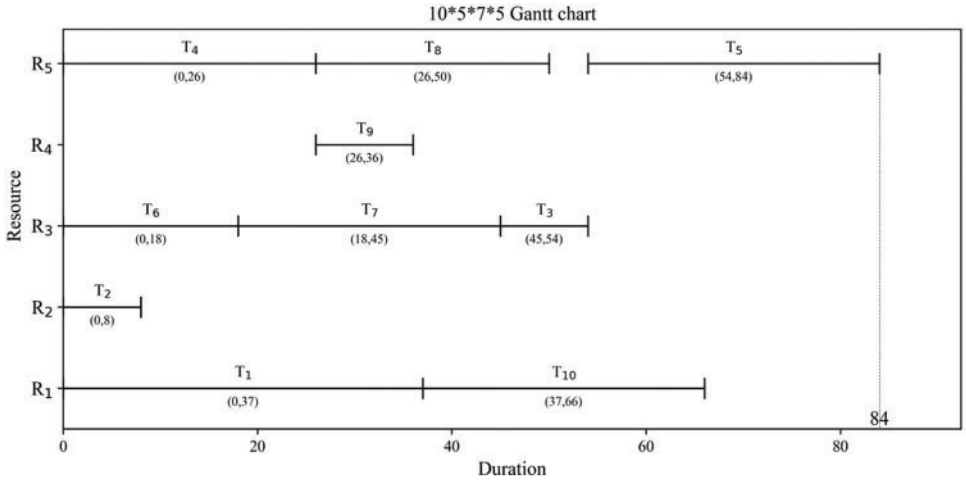


**Figure 5:** The Gantt chart of instance 10 * 5 * 7 * 5

Fig. 5 shows the starting time, the finishing time, the resource selection of each task, and so on. The project completion time is 84.

**Procedure 2** Multi-skill resource constrained project scheduling process

**Input:** $R$ the set of the resources,$T$ the set of the tasks,$S$ the set of the skills, $D$ the set of duration for each task

**Output:** $Tn$ scheme,$C_{max}$ makespan

$Tn \leftarrow \Phi$,$P \leftarrow \Phi$, $F \leftarrow \Phi$,$\pi \leftarrow T$, $R^* \leftarrow R$
$g \leftarrow 1$,$t_{s(1)} \leftarrow 0$
**while** $len(F) < len(T)$ **do**
  **for** each task $j$ of $\pi$ **do**
    **if** all the predecessor tasks of task $j$ are finished **then**
      Add task $j$ to $\pi^*$
    **end if**
  **end for**
  **for** each task $j$ of $\pi^*$ **do**
    **for** each resource $k$ **do**
      **if** $G_{kn} \geq \eta_{jn}$ **then**
        Add task $j$ to $\pi_k{}^*$
      **end if**
    **end for**
  **end for**
  **for** each resource $k$ **do**
    **if** resource $k$ in $R$ and $\pi_k{}^* \neq \Phi$ **then**
      Select task $j$ of $\pi_k{}^*$ with the highest priority value
      Save (task $j, t_{s(g)}, t_{s(g)} + d_j$, resource $k$) into $Tn$
      Remove resource $k$ from $R^*$

(Continued)

```
            Remove task j form π
            Add task j in P
            for each π_k* do
                if task j in π_k* then
                    Remove task j from π_k*
                end if
            end for
        end if
    end for
    P' ← P
    for each task j of P do
        if task j is the earliest completed task of P' then
            Remove task j from P
            Add task j in F
            t_f(g) ← s_j + d_j
        end if
    end for
    g ← g + 1
    t_s(g) ← t_f(g−1)
end while
C_max ← t_f(n)
End Procedure
```

### *5.5 Neighborhood Structures*

An efficient neighborhood structure will significantly improve the performance of the IGEP algorithm. Six neighborhood structures {M1, M2, M3, M4, M5, M6} are proposed. The detail flowchart of six neighborhood structures in Fig. 6.

Gene swap ($M1$): A chromosome *chrom* is selected by a certain probability. Then two different genes ($chrom_{(a)}$ and $chrom_{(b)}$) are randomly chosen from *chrom* to swap. Note: To avoid the generation of illegal chromosomes, it is necessary to limit position $b$. If $chrom_{(a)}$ is an element from the function set $F$, then $chrom_{(b)}$ can only be selected from individual head genes; otherwise, $chrom_{(b)}$ is selected from the non-functional genes of the chromosome.

Gene forward inserts ($M2$): A chromosome *chrom* is selected by a certain probability. Then two genes ($chrom_{(a)}$ and $chrom_{(b)}$) are randomly selected from *chrom*. The latter gene $chrom_{(b)}$ is inserted in front of the former gene $chrom_{(a)}$. Note: $chrom_{(a)}$ and $chrom_{(b)}$ both are not the first gene in *chrom*. Meanwhile, to avoid the generation of illegal chromosomes, if $chrom_{(a)}$ is the gene at head positions, and the last gene at the head position is an element from the function set $F$, then $chrom_{(b)}$ can only select the gene from head positions.

Gene backward inserts ($M3$): A chromosome *chrom* is selected by a certain probability. Then two genes ($chrom_{(a)}$ and $chrom_{(b)}$) are chosen randomly. The former gene $chrom_{(a)}$ is inserted in front of the latter gene $chrom_{(b)}$. Note: $chrom_{(a)}$ and $chrom_{(b)}$ both are not the first gene in *chrom*. Meanwhile, to avoid the generation of illegal chromosomes, if $chrom_{(a)}$ is the gene at head positions, then $chrom_{(b)}$ can only select the gene from head positions.

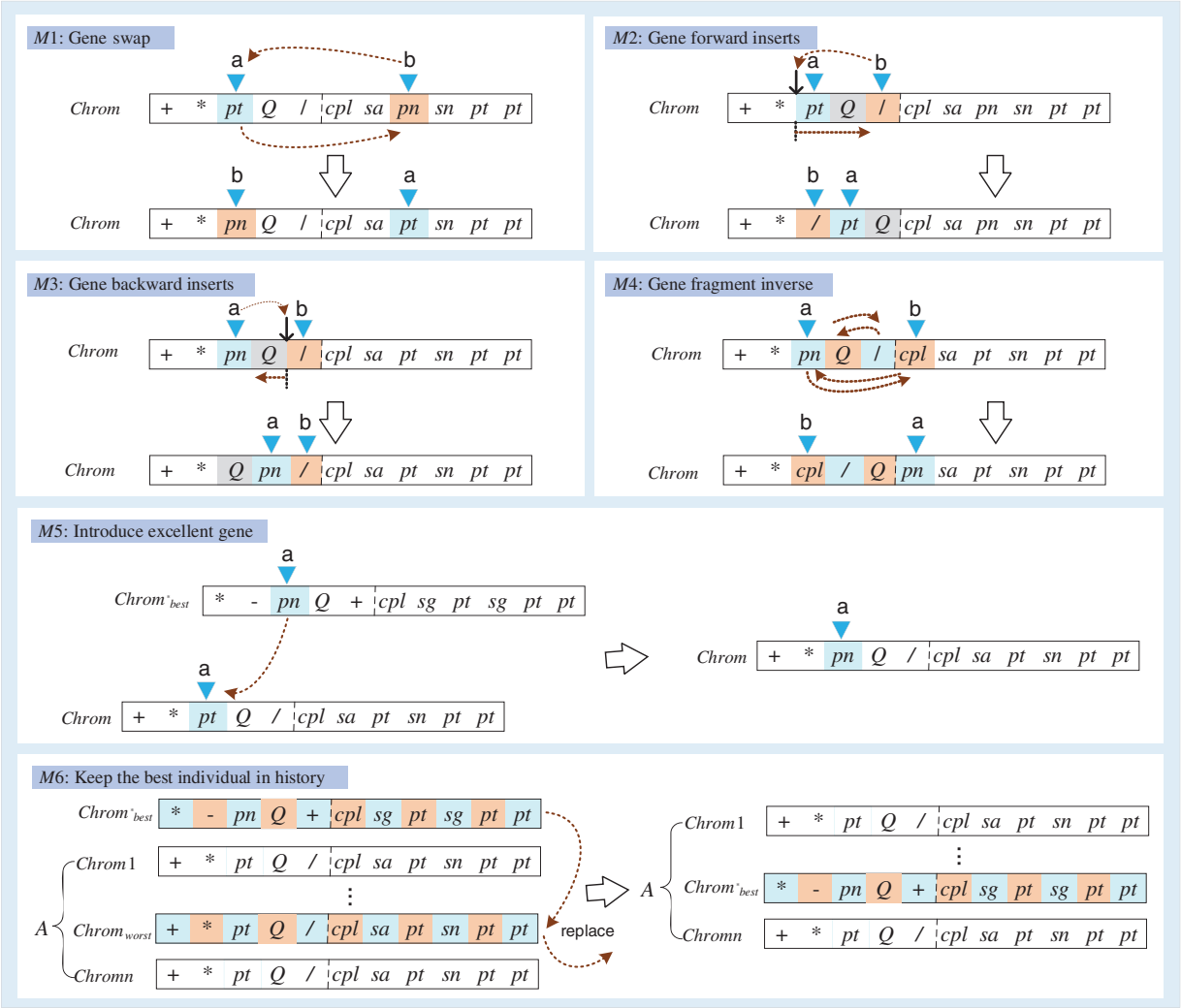


**Figure 6:** The flowchart of six neighborhood structures

Gene fragment inverse ($M4$): A chromosome *chrom* is selected by a certain probability. Then two genes ($chrom_{(a)}$ and $chrom_{(b)}$) are chosen randomly. All genes $\{chrom_{(a)}, chrom_{(a+1)}, \ldots, chrom_{(b)}\}$ are updated in reversed order. Note: To avoid the generation of illegal chromosomes, if $chrom_{(a)}$ is the gene at head positions, then $chrom_{(b)}$ can only select the gene from head positions.

Introduce excellent gene ($M5$): A chromosome *chrom* is selected by a certain probability. Then a gene position $a$ is randomly selected. The gene $chrom_{(a)}$ is replaced by $chrom^*_{best(a)}$ from the historical optimal individual $chrom^*_{best}$. A new chromosome $chrom_{new}$ is generated. If the fitness value of $chrom_{new}$ is higher than that of *chrom*, then replace the *chrom* in the population with $chrom_{new}$.

Keep the best individual in history ($M6$): If the historical optimal chromosome is not in the population, the worst chromosome $chrom_{worst}$ is replaced by the historical optimal chromosome $chrom^*_{best}$.

### *5.6 IGEP Framework*

The proposed IGEP algorithm can be roughly divided into four steps. The first step is to determine the elements of the function set and terminal set, and to set parameter values, such as chromosome length, mutation probability, etc. The second step is to divide the data set into the training set and testing set with a ratio of 2:3. The third step is to train the best individual by the training set data. The fourth step is to use the testing set data to test the performance and verify the performance of the IGEP algorithm. Fig. 7 shows the flowchart of the IGEP algorithm.

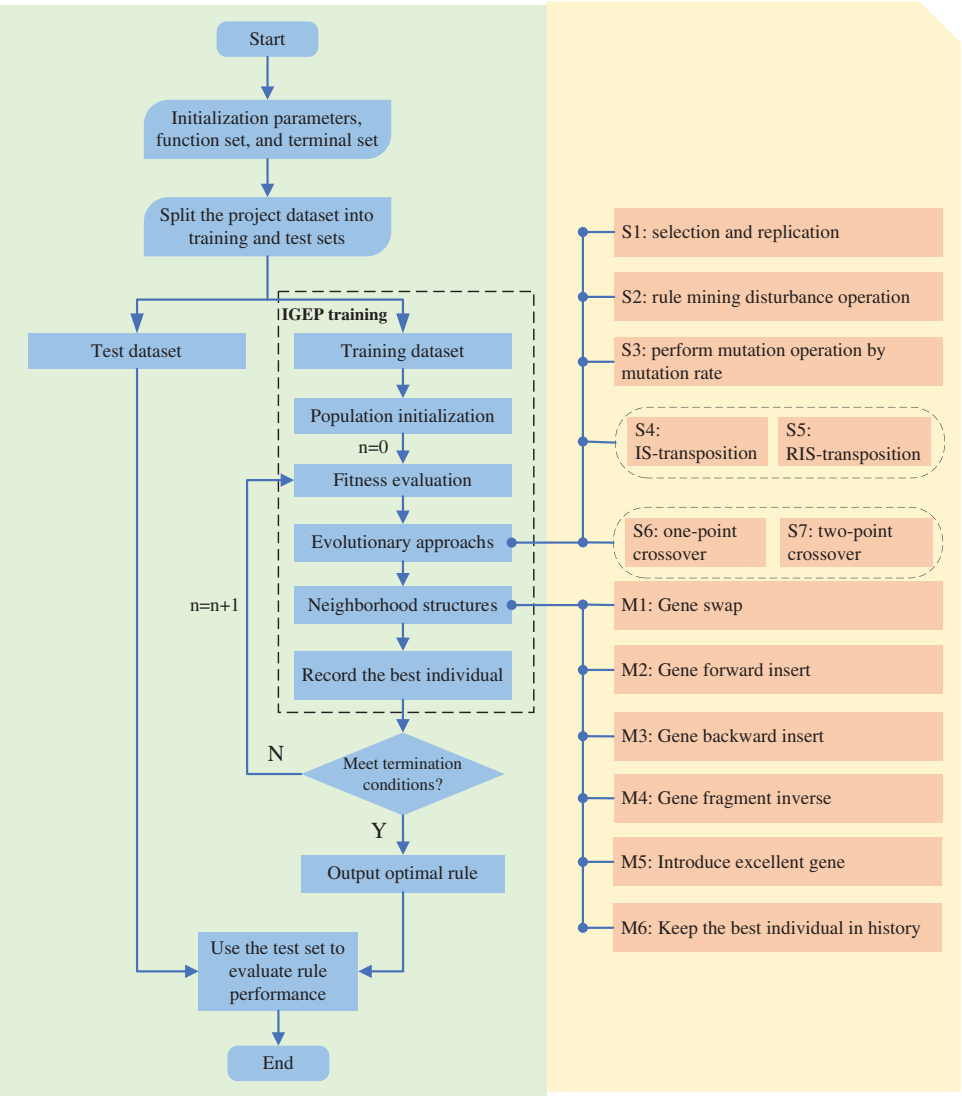


**Figure 7:** The flowchart of the IGEP algorithm

## 6 Computational Results and Analysis

To evaluate the performance of the IGEP algorithm, the benchmark case set (iMOPSE) created by Myszkowski et al. [2] in 2015 is adopted. The iMOPSE contains 36 instances. The IGEP algorithm

is implemented in Python 3.10 and operated on a core R7-5800H processor with 3.3 GHz and 16 GB RAM.

### *6.1 Parameter Setting*

In this section, the parameters are determined. The proposed IGEP algorithm with the long length of the head is easier to obtain excellent dispatching rules. But the computational complexity will increase significantly. After a series of experiments, when the length of the head h is equal to 7, the proposed IGEP algorithm can maintain superior performance and low computational complexity. Otherwise, the parameters such as population size, number of iterations, the preset value of rule mining disturbance, preset probability of rule mining, and mutation rate also significantly impact the performance of the proposed IGEP algorithm.

Before the experiment, the results of IGEP algorithm under different combination of numerical parameters were compared. The comparative method adopts the calculation of relative deviation. The group of parameters with the smallest relative deviation value was selected as the input parameters of the experiment. The parameters of the IGEP algorithm are listed in Table 4.

**Table 4:** The parameters of the IGEP algorithm

| Notations | Definition | Value | Notations | Definition | Value |
|---|---|---|---|---|---|
| *F* | Function set, $F=\{+, -, *, /, Q,$ max, min$\}$ | | *V* | Terminal set, *V* = {*pt, pn, sn, pa, sa, sg, rn, cpl, cpn, lf, tw*} | |
| *size* | Population size | 20 | *P_ss* | IS-transposition rate | 0.1 |
| *C* | The upper limit of iterations | 50 | *p_sr* | RIS-transposition rate | 0.1 |
| *M* | The upper limit of fitness value | 300 | *P_r1* | One-point crossover rate | 0.1 |
| *h* | Length of chromosome head | 7 | *P_r2* | Two-point crossover rate | 0.1 |
| *d_value* | Set the value of the rule mining disturbance operation | 3 | *P_i* | The rate of gene inverse | 0.1 |
| *P_dv* | The rate of the rule mining disturbance operation | 0.3 | *P_e* | The rate of gene position exchange | 0.1 |
| *P_m* | The mutation rate | 0.1 | *P_rl* | The rate of introducing an excellent gene | 0.3 |

To illustrate the performance of the newly dispatching rule generated by the IGEP algorithm, this paper selects six relatively typical dispatching rules. These six dispatching rules are described as follows:

1) Shortest processing time priority rule (SPT) [45], this dispatching rule takes the task duration *pt* as the evaluation criterion. The smaller *pt* value means the higher priority of the task. It is a typical dispatching rule based on the project network.
2) Longest processing time priority rule (LPT) [45], this dispatching rule is the opposite of the SPT rule. The task with a large *pt* value has a higher priority.

3) Longest of the remaining critical path priority rule (LRCP) [46], this dispatching rule only considers the task's remaining critical path length *cpl* value. And the task with a large *cpl* value has a higher priority.
4) Minimum slack time priority rule (MINSLK) [44]. In this rule, the slack time *tw* of the task is taken as the evaluation criterion. The task with a smaller *tw* value is given a higher priority.
5) Latest finishing time priority rule (LLFT), it considers the value of the task's latest finishing time *lf*. The purpose is to assign tasks with smaller *lf* values as early as possible.
6) The most number of immediate successors priority rule (MIS) [44], this rule gives a higher priority to a task with a large number of immediate tasks. A task with a large *sn* value has a higher priority.

According to different factors considered, dispatching rules can be roughly divided into four types, such as project network-based dispatching rules (NBR), critical path-based dispatching rules (CPBR), resource-based dispatching rules (RBR), and hybrid dispatching rules (CR) [44]. The SPT rule, the LPT rule, and the MIS rule are the project network-based dispatching rules. The LRCP rule, the MINSLK rule, and the LLFT rule are critical path-based dispatching rules.

### *6.2 The iMOPSE Benchmark Dataset*

The iMOPSE benchmark dataset is selected for multi-skill resource-constrained project scheduling problem research. The dataset contains 36 instances with 100 or 200 tasks and 5, 10, 20, or 40 resources. The number of skill types is 9, 14, or 15. The dataset was created by Myszkowski et al. [2] in 2015 and can be found on website http://imopse.ii.pwr.edu.pl.

### *6.3 The Discovered Dispatching Rules via IGEP*

The purpose of the IGEP algorithm in this paper is to explore the optimal dispatching rule for solving the multi-skill resource-constrained project scheduling problem. According to the approximate ratio of 2:3, 16 instances are randomly selected as the training set. The remaining 20 instances are the testing set. The training set is used to discover the optimal dispatching rule during the exploration of the IGEP algorithm. The testing set is used to verify the performance of the optimal dispatching rule.

Fig. 8 shows ten discovered dispatching rules under 10 independent runs of the proposed IGEP algorithm. Eight discovered dispatching rules positively correlate with the remaining critical path length *cpl*. Six discovered dispatching rules positively correlate with the maximum skill level requirement *sg*. This means that the task with large *cpl* and *sg* has priority to be allocated resources and to be started.

It can be seen from the 10 new dispatching rules that *pn* and *sn* are completely lost. By analyzing the case set data, it is found that all tasks have only one direct predecessor task, and the number of direct successor tasks of most tasks is 0 after statistics. In addition, *pa* and *sa* have poor performance in 10 new dispatching rules. However, both *sg* and *cpl* have achieved good performance. Therefore, it is bold to make an assumption that the relatively simple predecessor relation network of the case leads to poor performance of the attributes related to the predecessor relation network, which is gradually replaced by the attribute values related to duration in the iterative process.

| | Chromosome | | | | | | | | | | | | | | Dispatching rule |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1) | * | *sg* | *cpl* | *sg* | / | *tw* | *lf* | *sa* | *pt* | *sn* | *sa* | *pn* | *sg* | *sn* | *lf* | $sg*cpl$ |
| 2) | *min* | *min* | *Q* | *min* | *cpl* | * | *cpl* | *cpl* | *cpn* | *sg* | *sa* | *pt* | *sn* | *sa* | *cpl* | $\min(cpl, \sqrt{cpn*sg})$ |
| 3) | * | *max* | *sg* | *sa* | *min* | *cpl* | *pt* | *cpn* | *sn* | *pt* | *sa* | *pt* | *tw* | *pa* | *lf* | $\max(sa, \min(cpl, pt))*sg$ |
| 4) | *min* | * | *cpn* | *sg* | *cpl* | *sg* | *pa* | *sa* | *pt* | *sn* | *pt* | *rn* | *sg* | *rn* | *sa* | $\min(sg*cpl, cpn)$ |
| 5) | * | *max* | *cpl* | *Q* | *cpn* | *sg* | *max* | *tw* | *cpn* | *sg* | *lf* | *pa* | *pa* | *rn* | *tw* | $\max(\sqrt{sg}, cpn)*cpl$ |
| 6) | * | + | / | *max* | *tw* | *cpl* | / | *pt* | *pa* | *tw* | *cpl* | *pt* | *pa* | *lf* | *pa* | $\frac{(\max(pt, pa)+tw)}{tw}$ |
| 7) | * | *max* | *cpl* | + | *min* | *pt* | / | *cpn* | *cpl* | *cpn* | *rn* | *rn* | *tw* | *sn* | *cpl* | $\max(pt+\frac{cpn}{rn}, \min(cpn, cpl))*cpl$ |
| 8) | + | *sa* | * | *rn* | / | - | *cpl* | *pa* | *sg* | *pa* | *lf* | *cpl* | *cpl* | *sg* | *rn* | $sa+\frac{rn*(pa-sg)}{cpl}$ |
| 9) | *Q* | + | *sg* | *cpl* | *min* | *pt* | *pn* | *pn* | *sg* | *cpl* | *sn* | *lf* | *cpn* | *lf* | *pn* | $\sqrt{sg+cpl}$ |
| 10) | / | / | *rn* | *cpl* | *min* | *Q* | *Q* | *lf* | *lf* | *pa* | *sa* | *cpl* | *sn* | *pt* | *pt* | $\frac{cpl*rn}{\min(\sqrt{lf}, \sqrt{tw})}$ |

**Figure 8:** Ten discovered dispatching rules acquired from 10 independent runs

### *6.4 Comparison among Ten Discovered Dispatching Rules*

One-way analysis of variance (ANOVA) was used to analyze the statistical differences in the performance of 10 dispatching rules. The actual project completion time $C_{\max}$ has significant differences. Therefore, the experiment uses the relative percentage deviation (RPD) to measure the performance of 10 discovered dispatching rules. The computational formula of RPD is shown in expression (14).

$$RPD_{j,i} = \frac{C_{j,i} - \min C_j}{\max C_j - \min C_j} \times 100\% \tag{14}$$

where $C_{j,i}$ represents the $C_{\max}$ value of the instance $j$ via the dispatching rule $i$. Max$C_{\max}$ and min$C_{\max}$ represent the maximum $C_{\max}$ value and minimum $C_{\max}$ value of instance $j$ via all dispatching rules, respectively. $RPD_{j,i}$ is the RPD value of instance $j$ via dispatching rule $i$. In the one-way ANOVA of this experiment, the single factor refers to the discovered dispatching rules. The response variable is the RPD value. The ANOVA results are shown in Table 5.

**Table 5:** ANOVA results for ten discovered dispatching rules

| Source | Degrees of freedom | Adj SS | Adj MS | F-value | *P*-value |
|---|---|---|---|---|---|
| Factor | 9 | 7.925 | 0.8805 | 8.76 | 0.000 |
| Error | 350 | 35.191 | 0.1005 | | |
| A combined | 359 | 43.115 | | | |

It can be seen from Table 5 that the *P*-value is approximately 0. Hence, all dispatching rules have statistical differences. This shows that the performance of the ten dispatching rules is significantly different. The pairwise multiple comparisons display the detailed differences among ten dispatching rules and find the high-quality ones, as shown in Fig. 9. Fig. 9 shows the difference in the performance of each discovered dispatching rule.

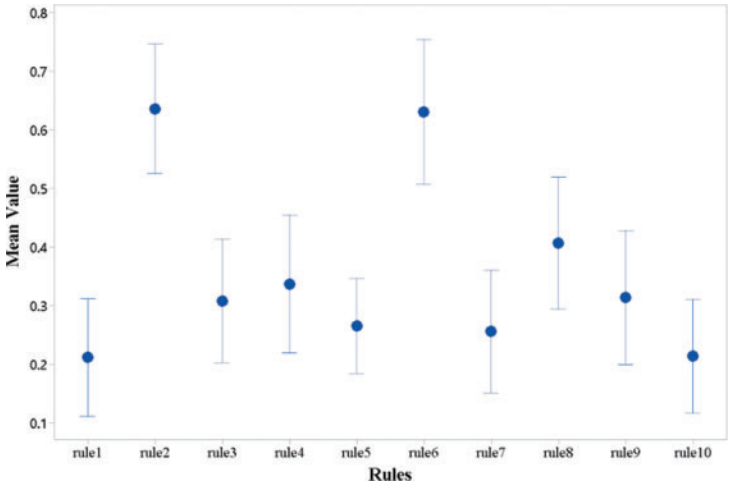


**Figure 9:** Pairwise multiple comparisons for ten discovered dispatching rules

It can be seen that the group means of *rule*1 and *rule*10 are significantly better than the group means of other discovered dispatching rules in Fig. 9. The group means of *rule*3, *rule*4, *rule*5, *rule*7, *rule*8, and *rule*9 are smaller than the group means of *rule*1 and *rule*10. The group means of *rule*2 and *rule*6 are significantly larger than the group means of the other eight discovered dispatching rules. The mean value of RPD for ten discovered dispatching rules are 0.211, 0.636, 0.308, 0.336, 0.265, 0.63, 0.255, 0.406, 0.313, 0.213. The *rule*1 with 0.211 mean value of RPD is the best discovered dispatching rule. Meanwhile, the mean value of RPD of *rule*3, *rule*4, *rule*5, *rule*7, *rule*8, *rule*9, and *rule*10 is the closest dispatching rule to *rule*1. To verify the statistical performance of these eight discovered dispatching rules, One-way analysis of variance (ANOVA) is used to analyze the performance of *rule*1, *rule*3, *rule*4, *rule*5, *rule*7, *rule*8, *rule*9, and *rule*10. To verify the statistical performance of these eight discovered dispatching rules, One-way analysis of variance (ANOVA) is used to analyze the performance of *rule*1, *rule*3, *rule*4, *rule*5, *rule*7, *rule*8, *rule*9, and *rule*10. As can be seen in Table 6, the *P*-value exceeds 0.5. Therefore, this difference was not statistically significant for *rule*1, *rule*3, *rule*4, *rule*5, *rule*7, *rule*8, *rule*9, and *rule*10.

**Table 6:** ANOVA results for eight discovered dispatching rules

| Source | Degrees of freedom | Adj SS | Adj MS | F-value | *P*-value |
|---|---|---|---|---|---|
| Factor | 7 | 0.0182 | 0.0026 | 0.8504 | 0.5464 |
| Error | 280 | 0.8548 | 0.0031 | | |
| A combined | 287 | 0.8729 | | | |

### *6.5 Comparison with the Typical Dispatching Rules*

In order to verify the effectiveness and excellent performance of the IGEP algorithm, eight discovered dispatching rules are compared and analyzed with six typical dispatching rules. Table 7 shows the comparison results of the completion time $C_{max}$ and the relative percentage deviation value RPD among the above-mentioned dispatching rules.

**Table 7:** The comparison results of $C_{max}$ and RPD among dispatching rules

| | Rule1 | Rule3 | Rule4 | Rule5 | Rule7 | Rule8 | Rule9 | Rule10 | SPT | LPT | LRCP | MINSLK | LLFT | MIS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 100_10_26_15 | 259/0.23 | 250/0.14 | 259/0.23 | 249/0.13 | **236/0** | 251/0.15 | 259/0.23 | 240/0.04 | 335/1 | 321/0.86 | 263/0.27 | 285/0.49 | 257/0.21 | 257/0.21 |
| 100_10_27_9_D2 | 223/0.17 | 213/0.04 | 215/0.07 | **210/0** | 226/0.21 | 216/0.08 | 219/0.12 | 212/0.03 | 273/0.84 | 285/1 | 246/0.48 | 241/0.41 | 232/0.29 | 251/0.55 |
| 100_10_47_9 | 257/0.08 | 258/0.09 | 270/0.32 | 257/0.08 | 272/0.36 | 256/0.06 | **253/0** | 260/0.13 | 304/0.96 | 306/1 | 287/0.64 | 265/0.23 | 275/0.42 | 271/0.34 |
| 100_10_48_15 | **245/0** | 250/0.09 | 249/0.07 | 248/0.05 | 248/0.05 | 247/0.03 | 255/0.17 | **245/0** | 300/0.95 | 303/1 | 249/0.07 | 249/0.07 | 262/0.29 | 257/0.21 |
| 100_10_64_9 | 245/0.03 | 244/0.01 | 248/0.07 | **243/0** | 245/0.03 | 267/0.32 | 249/0.08 | 267/0.32 | 308/0.88 | 287/0.59 | 253/0.14 | 317/1 | 254/0.15 | 249/0.08 |
| 100_10_65_15 | 251/0.04 | **247/0** | 252/0.05 | 252/0.05 | 252/0.05 | 253/0.07 | 249/0.02 | 248/0.01 | 338/1 | 308/0.67 | 268/0.23 | 265/0.20 | 299/0.57 | 321/0.81 |
| 100_20_22_15 | 137/0.14 | 135/0.10 | 136/0.12 | **130/0** | 132/0.04 | 135/0.10 | 135/0.10 | 134/0.08 | 180/1 | 163/0.66 | **130/0** | 145/0.30 | 147/0.34 | 142/0.24 |
| 100_20_23_9_D1 | 180/0.13 | 180/0.13 | 180/0.13 | 180/0.13 | 180/0.13 | 176/0.06 | 180/0.13 | 175/0.05 | **172/0** | 235/1 | 181/0.14 | 177/0.08 | 174/0.03 | 173/0.02 |
| 100_20_46_15 | 218/0.63 | 218/0.63 | 218/0.63 | 188/0.22 | 173/0.01 | 182/0.14 | 180/0.11 | **172/0** | 215/0.59 | 201/0.40 | 188/0.22 | 212/0.55 | 245/1 | 236/0.88 |
| 100_20_47_9 | 126/0.04 | 145/0.38 | 126/0.04 | **124/0** | **124/0** | 151/0.49 | 127/0.05 | 144/0.36 | 170/0.84 | 156/0.58 | 165/0.75 | 179/1 | 147/0.42 | 164/0.73 |
| 100_20_65_15 | 226/0.21 | 226/0.21 | 259/0.53 | 274/0.68 | 272/0.66 | 233/0.27 | **205/0** | 242/0.36 | 216/0.11 | 260/0.54 | 254/0.48 | 307/1 | 295/0.88 | 293/0.86 |
| 100_20_65_9 | 128/0.07 | 140/0.48 | 155/1 | 127/0.03 | **126/0** | 128/0.07 | 127/0.03 | **126/0** | 152/0.90 | 155/1 | 129/0.10 | 145/0.66 | 141/0.52 | 138/0.41 |
| 100_5_20_9_D3 | 389/0.02 | 391/0.05 | **388/0** | 390/0.03 | 390/0.03 | 392/0.06 | 389/0.02 | 392/0.06 | 452/1 | 439/0.80 | 391/0.05 | 390/0.03 | 412/0.38 | 413/0.39 |
| 100_5_22_15 | 523/0.48 | 490/0.04 | 517/0.40 | 501/0.19 | 507/0.27 | 495/0.11 | 512/0.33 | **487/0** | 562/1 | 556/0.92 | 517/0.40 | 518/0.41 | 517/0.40 | 516/0.39 |
| 100_5_46_15 | **539/0** | 548/0.07 | 562/0.17 | 558/0.14 | 570/0.23 | 590/0.38 | 543/0.03 | 547/0.06 | 675/1 | 586/0.35 | 600/0.45 | 585/0.34 | 592/0.39 | 590/0.38 |
| 100_5_48_9 | 496/0.03 | 498/0.05 | 497/0.04 | 497/0.04 | 519/0.25 | 532/0.37 | 498/0.05 | **493/0** | 542/0.46 | 529/0.34 | 564/0.67 | 579/0.81 | 584/0.86 | 599/1 |
| 100_5_64_15 | 504/0.20 | 512/0.31 | 504/0.20 | 530/0.55 | 522/0.44 | 499/0.13 | 518/0.39 | **489/0** | 527/0.51 | 555/0.88 | 544/0.73 | 564/1 | 559/0.93 | 560/0.95 |
| 100_5_64_9 | 484/0.13 | 484/0.13 | 481/0.03 | 481/0.03 | **480/0** | 489/0.29 | 482/0.06 | **480/0** | 511/1 | 510/0.97 | 483/0.10 | 485/0.16 | 491/0.35 | 495/0.48 |
| 200_10_128_15 | 476/0.10 | 468/0.03 | 478/0.12 | 479/0.12 | **464/0** | 485/0.17 | 490/0.21 | 465/0.01 | 585/1 | 555/0.75 | 536/0.60 | 547/0.69 | 562/0.81 | 564/0.83 |
| 200_10_135_9_D6 | **554/0** | 569/0.11 | 564/0.08 | 574/0.15 | 590/0.27 | 581/0.21 | 585/0.24 | 561/0.05 | 621/0.51 | 685/1 | 583/0.22 | 570/0.12 | 573/0.15 | 595/0.31 |
| 200_10_50_15 | 508/0.28 | 502/0.20 | 508/0.28 | 498/0.15 | 489/0.04 | 506/0.25 | 530/0.55 | **486/0** | 530/0.55 | 542/0.70 | 522/0.45 | 524/0.47 | 566/1 | 561/0.94 |
| 200_10_50_9 | **486/0** | 488/0.03 | 529/0.61 | 491/0.07 | 487/0.01 | 489/0.04 | 489/0.04 | 487/0.01 | 504/0.26 | 556/1 | 500/0.20 | 504/0.26 | 549/0.90 | 536/0.71 |
| 200_10_84_9 | 509/0.03 | 508/0.02 | 509/0.03 | 510/0.05 | 510/0.05 | 510/0.05 | **507/0** | 509/0.03 | 538/0.51 | 568/1 | 511/0.07 | 517/0.16 | 536/0.48 | 536/0.48 |
| 200_10_85_15 | 479/0.04 | **477/0** | 479/0.04 | 479/0.04 | **477/0** | 486/0.18 | 480/0.06 | **477/0** | 507/0.60 | 503/0.52 | 479/0.04 | 496/0.38 | 524/0.94 | 527/1 |
| 200_20_145_15 | **235/0** | 249/0.21 | 239/0.06 | 238/0.04 | 237/0.03 | 241/0.09 | 238/0.04 | 239/0.06 | 287/0.76 | 286/0.75 | 240/0.07 | 303/1 | 302/0.99 | 273/0.56 |
| 200_20_150_9_D5 | **900/0** | **900/0** | **900/0** | **900/0** | **900/0** | 902/0.25 | **900/0** | 904/0.50 | **900/0** | **900/0** | **900/0** | **900/0** | 908/1 | 908/1 |
| 200_20_54_15 | 261/0.03 | **259/0** | 260/0.01 | 292/0.45 | 261/0.03 | 264/0.07 | **259/0** | 260/0.01 | 322/0.85 | 329/0.95 | 292/0.45 | 318/0.80 | 323/0.86 | 333/1 |
| 200_20_55_9 | **248/0** | 249/0.02 | **248/0** | 249/0.02 | **248/0** | **248/0** | 250/0.04 | 249/0.02 | 298/1 | 283/0.70 | 251/0.06 | 250/0.04 | 266/0.36 | 268/0.40 |
| 200_20_97_15 | **336/0** | **336/0** | **336/0** | **336/0** | **336/0** | **336/0** | **336/0** | **336/0** | 391/1 | 357/0.38 | **336/0** | **336/0** | 367/0.56 | 361/0.45 |
| 200_20_97_9 | **239/0** | 244/0.06 | 241/0.02 | 242/0.04 | 241/0.02 | 242/0.04 | 243/0.05 | 242/0.04 | 277/0.47 | 320/1 | 262/0.28 | 268/0.36 | 265/0.32 | 265/0.32 |
| 200_40_130_9_D4 | **513/0** | **513/0** | **513/0** | **513/0** | **513/0** | **513/0** | **513/0** | **513/0** | **513/0** | **513/0** | **513/0** | **513/0** | **513/0** | **513/0** |
| 200_40_133_15 | **133/0** | 141/0.11 | **133/0** | 134/0.01 | 134/0.01 | 140/0.10 | 134/0.01 | 138/0.07 | 183/0.69 | 205/1 | 142/0.12 | 148/0.21 | 146/0.18 | 148/0.21 |
| 200_40_45_15 | **159/0** | **159/0** | **159/0** | 199/0.65 | **159/0** | **159/0** | **159/0** | **159/0** | 175/0.26 | **159/0** | 221/1 | 221/1 | 183/0.39 | 183/0.39 |
| 200_40_45_9 | **135/0** | 136/0.02 | **135/0** | 137/0.04 | 138/0.06 | 139/0.08 | 145/0.20 | 137/0.04 | 165/0.59 | 186/1 | 138/0.06 | 137/0.04 | 151/0.31 | 152/0.33 |
| 200_40_90_9 | **131/0** | **131/0** | 152/0.34 | 132/0.02 | 148/0.27 | 135/0.06 | 140/0.15 | 133/0.03 | 193/1 | 180/0.79 | 135/0.06 | 133/0.03 | 149/0.29 | 148/0.27 |
| 200_40_91_15 | 132/0.06 | 138/0.15 | 130/0.03 | **128/0** | **128/0** | 152/0.36 | 136/0.12 | 132/0.06 | 183/0.82 | 195/1 | **128/0** | 137/0.13 | 178/0.75 | 152/0.36 |
| Average | 329.56/0.09 | 330.44/0.11 | 334.14/0.16 | 332.50/0.12 | 331.50/0.10 | 333.89/0.14 | 330.94/0.10 | **327.22/0.07** | 372.33/0.69 | 374.36/0.72 | 344.47/0.27 | 353.61/0.40 | 359.56/0.52 | 359.67/0.51 |

It can be seen from Table 7, eight discovered dispatching rules obtain the best $C_{max}$ than six typical dispatching rules. 14, 12, and 12 instances of *rule*1, *rule*7, and *rule*10 obtained the best $C_{max}$, respectively. Six typical dispatching rules with a single attribute have a bad performance. The LRCP rule is the best one among the six typical dispatching rules. In sum, the discovered dispatching rules play a good performance in solving MS-RCPSP problem.

To further study the differences in the performance of the dispatching rules, a one-factor analysis of variance (ANOVA) is performed on 36 instances. The single factor refers to the fourteen dispatching rules. The response variable is the RPD value. The ANOVA results are shown in Table 8.

**Table 8:** ANOVA results for fourteen dispatching rules

| Source | Degrees of freedom | Adj SS | Adj MS | F-value | *P*-value |
|---|---|---|---|---|---|
| Factor | 13 | 26.31 | 2.02389 | 36.61 | 0.000 |
| Error | 490 | 27.09 | 0.05528 | | |
| A combined | 503 | 53.40 | | | |

The ANOVA result in Table 8 shows that the *P*-value is close to 0. This indicates that the statistical performance of the fourteen dispatching rules is significantly different. To clearly show the difference among those dispatching rules, the mean square error control chart is applied to describe the difference, as shown in Fig. 10.

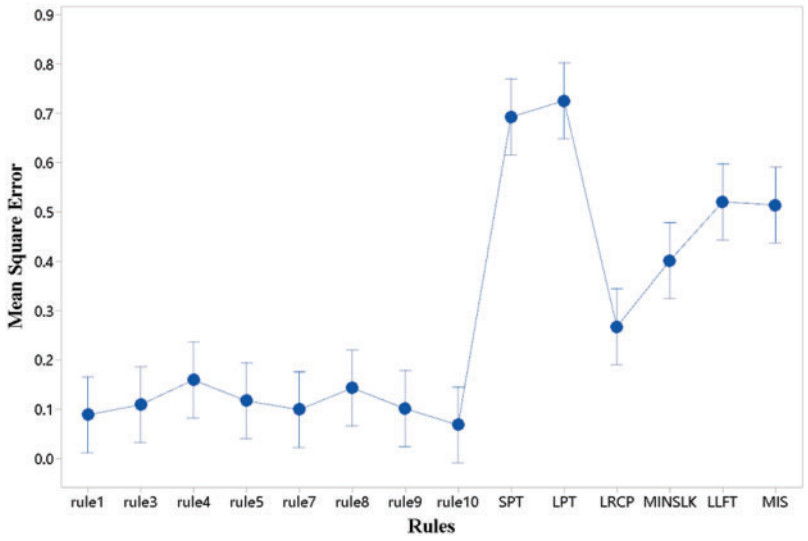


**Figure 10:** Mean square error control chart

Fig. 10 shows the mean square error values of fourteen dispatching rules, which are 0.088, 0.109, 0.159, 0.117, 0.099, 0.143, 0.101, 0.068, 0.692, 0.725, 0.267, 0.401, 0.52, and 0.514, respectively. *rule*10 with the smallest mean square error is the best dispatching rule, followed by *rule*1 and *rule*7. These mean square error values of eight dispatching rules mined by the IGEP algorithm are much smaller than that of six typical dispatching rules. It demonstrates that the dispatching rule trained by the IGEP algorithm is much better than typical dispatching rules for solving MS-RCPSP.

## 7 Conclusion and Future Research

For solving the multi-skill resource-constrained project scheduling problem (MS-RCPSP), this paper constructs a mix-integer mathematical model and proposes an improved gene expression programming algorithm (IGEP) with a backward traversal decoding mechanism to explore the newly dispatching rules. These newly dispatching rules can easily be applied to the real project and production environment. The experimental analysis shows that the newly discovered dispatching rules play a better performance than the typical dispatching rules. This illustrates that the proposed IGEP algorithm with the merit of artificial intelligence and unsupervised learning is effective in exploring the newly dispatching rules for solving the MS-RCPSP problem. Compared with complex real project scenarios and existing research, there are many research directions that are worthy of in-depth study in future research.

1) The skill proficiency of a resource grows with the long operations, which can increase productivity, and the resource can perform the more demanding tasks. Thus, adding learning mechanisms to resources is closer to some realistic production and maintenance scenarios.
2) A reasonable resource allocation plan can result in significant cost savings in real project scenarios. Therefore, having a reasonable resource allocation plan while ensuring the project completion time is a direction well worth our research.

**Funding Statement:** This paper presents work funded by the National Natural Science Foundation of China (Nos. 51875420, 51875421, 52275504).

**Conflicts of Interest:** The authors declare that they have no conflicts of interest to report regarding the present study.

## References

1. Lin, J., Zhu, L., Gao, K. Z. (2020). A genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem. *Expert Systems with Applications, 140*. https://doi.org/10.1016/j.eswa.2019.112915
2. Zhang, Z. K., Tang, Q. H., Chica, M. (2021). A robust MILP and gene expression programming based on heuristic rules for mixed-model multi-manned assembly line balancing. *Applied Soft Computing, 109,* 107513. https://doi.org/10.1016/j.asoc.2021.107513
3. Myszkowski, P. B., Skowroński, M. E., Sikora, K. (2015). A new benchmark dataset for multi-skill resource-constrained project scheduling problem. *Proceedings of the 2015 Federated Conference on Computer Science and Information Systems*, pp. 129–138. Lodz, Poland. https://doi.org/10.15439/2015F273
4. Myszkowski, P. B., Siemieński, J. J. (2016). GRASP applied to multi–skill resource–constrained project scheduling problem. *Department of Computational Intelligence*, Wroclaw University of Technology, Wroclaw, Poland. https://doi.org/10.1007/978-3-319-45243-237
5. Dziwiński, P., Bartczuk, Ł (2020). A new hybrid particle swarm optimization and genetic algorithm method controlled by fuzzy logic. *IEEE Transactions on Fuzzy Systems, 28,* 1140–1154. https://doi.org/10.1109/TFUZZ.2019.2957263
6. Li, L., Tang, Q. H., Zheng, P., Zhang, L. P. (2016). An improved self-adaptive genetic algorithm for scheduling steel-making continuous casting production. *Proceedings of the 6th International Asia Conference on Industrial Engineering and Management Innovation*, pp. 399–410. Tianjin, China. https://doi.org/10.2991/978-94-6239-148-2_40

7. Li, D., Zhang, D. D., Zhang, N., Zhang, L. P. (2022). A novel hybrid algorithm for scheduling multipurpose batch plants. *Computer Aided Chemical Engineering, 51,* 961–966. https://doi.org/10.1016/B978-0-323-95879-0.50161-2
8. Huang, R. X., Ning, J. Y., Mei, Z. H., Fang, X. D., Yi, X. M. et al. (2021). Study of delivery path optimization solution based on improved ant colony model. *Multimedia Tools and Applications, 80,* 28975–28987. https://doi.org/10.1007/s11042-021-11142-1
9. Guan, B. X., Zhao, Y. H., Li, Y. (2021). An improved ant colony optimization with an automatic updating mechanism for constraint satisfaction problems. *Expert Systems with Applications, 164,* 114021. https://doi.org/10.1016/j.eswa.2020.114021
10. He, M. L., Wei, Z. X., Wu, X. H., Peng, Y. T. (2021). An adaptive variable neighborhood search ant colony algorithm for vehicle routing problem with soft time windows. *IEEE Access, 9,* 21258–21266. https://doi.org/10.1109/access.2021.3056067
11. Fescioglu-Ünver, N., Kokar, M. M. (2011). Self controlling tabu search algorithm for the quadratic assignment problem. *Computers & Industrial Engineering, 60,* 310–319. https://doi.org/10.1016/j.cie.2010.11.014
12. Wang, Y. L., Wu, Z. P., Guan, G., Li, K., Chai, S. H. (2021). Research on intelligent design method of ship multi-deck compartment layout based on improved taboo search genetic algorithm. *Ocean Engineering, 225,* 108823. https://doi.org/10.1016/j.oceaneng.2021.108823
13. Hrizi, H. (2019). Improving the wave iterative method by metaheuristic algorithms. *Journal of Computational Electronics, 18(4),* 1365–1371. https://doi.org/10.1007/s10825-019-01394-4
14. Shao, W., Guo, G. B. (2018). Multiple-try simulated annealing algorithm for global optimization. *Mathematical Problems in Engineering, 2018,* 9248318. https://doi.org/10.1155/2018/9248318
15. Wang, K. P., Li, X. Y., Gao, L., Li, P. P., Gupta, S. M. (2021). A genetic simulated annealing algorithm for parallel partial disassembly line balancing problem. *Applied Soft Computing, 107,* 107404. https://doi.org/10.1016/j.asoc.2021.107404
16. Tang, Q. H., Li, Z. X., Zhang, L. P., Floudas, C. A. (2014). A hybrid particle swarm optimization algorithm for large-sized two-sided assembly line balancing problem. *ICIC Express Letters, 8,* 1981–1986.
17. Chen, C. H., Li, C. L. (2021). Process synthesis and design problems based on a global particle swarm optimization algorithm. *IEEE Access, 9,* 7723–7731. https://doi.org/10.1109/access.2021.3049175
18. Gu, Q. H., Liu, Y. Y., Chen, L., Xiong, N. X. (2022). An improved competitive particle swarm optimization for many-objective optimization problems. *Expert Systems with Applications: An International Journal, 189,* 116118. https://doi.org/10.1016/j.eswa.2021.116118
19. Zhang, Z. K., Tang, Q. H., Han, D. Y., Li, Z. X. (2022). Multi-manned assembly line balancing with sequence-dependent set-up times using an enhanced migrating birds optimization algorithm. *Engineering Optimization,* 1–20. https://doi.org/10.1080/0305215X.2022.2067992
20. Niroomand, S., Hadi-Vencheh, A., Sahin, R., Vizvári, B. (2015). Modified migrating birds optimization algorithm for closed loop layout with exact distances in flexible manufacturing systems. *Expert Systems with Applications, 42,* 6586–6597. https://doi.org/10.1016/j.eswa.2015.04.040
21. Wen, L., Gao, L., Li, X. Y., Zeng, B. (2021). Convolutional neural network with automatic learning rate scheduler for fault classification. *IEEE Transactions on Instrumentation and Measurement, 70,* 1–12. https://doi.org/10.1109/TIM.2020.304879
22. Wen, L., Li, X. Y., Gao, L. (2021). A new reinforcement learning based learning rate scheduler for convolutional neural network in fault classification. *IEEE Transactions on Industrial Electronics, 68,* 12890–12900. https://doi.org/10.1109/TIE.2020.3044808
23. Wen, L., Wang, Y., Li, X. Y. (2022). A new cycle-consistent adversarial networks with attention mechanism for surface defect classification with small samples. *IEEE Transactions on Industrial Informatics, 18(2),* 8988–8998. https://doi.org/10.1109/TII.2022.3168432

24. Cheng, L. X., Tang, Q. H., Zhang, L. P., Zhang, Z. K. (2022). Multi-objective Q-learning-based hyper-heuristic with bi-criteria selection for energy-aware mixed shop scheduling. *Swarm and Evolutionary Computation, 69,* 100985. https://doi.org/10.1016/j.swevo.2021.100985
25. Zhou, Q., Li, J. H., Dong, R. P., Zhou, Q. H., Yang, B. X. (2023). Optimization of multi-execution modes and multi-resource-constrained offshore equipment project scheduling based on a hybrid genetic algorithm. *Computer Modeling in Engineering & Sciences, 134(2),* 1263–1281. https://doi.org/10.32604/cmes.2022.020744
26. Myszkowski, P. B., Skowronski, M. E., Olech, L. P., Oslizlo, K. (2015). Hybrid ant colony optimization in solving multi-skill resource-constrained project scheduling problem. *Soft Computing, 19,* 3599–3619. https://doi.org/10.1007/s00500-014-1455-x
27. Maghsoudlou, H., Afshar-Nadjafi, B., Niaki, S. T. A. (2016). A multi-objective invasive weeds optimization algorithm for solving multi-skill multi-mode resource constrained project scheduling problem. *Computers and Chemical Engineering, 88,* 157–169. https://doi.org/10.1016/j.compchemeng.2016.02.018
28. Cui, L. Q., Liu, X. B., Lu, S. J., Jia, Z. L. (2021). A variable neighborhood search approach for the resource-constrained multi-project collaborative scheduling problem. *Applied Soft Computing, 107*. https://doi.org/10.1016/j.asoc.2021.107480
29. Li, Y. Y., Lin, J., Wang, Z. J. (2021). Multi-skill resource constrained project scheduling using a multi-objective discrete jaya algorithm. *Applied Intelligence, 52(5),* 5718–5738. https://doi.org/10.1007/s10489-021-02608-8
30. Zhu, L., Lin, J., Li, Y. Y., Wang, Z. J. (2021). A decomposition-based multi-objective genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem. *Knowledge-Based Systems, 225*. https://doi.org/10.1016/j.knosys.2021.107099
31. Haupt, R. (1988). A survey of priority rule-based scheduling. In: *Operations research spektrum*, vol. 11, pp. 3–16. Federal Republic of Germany.
32. Almeida, B. F., Correia, I., Saldanha-da-Gama, F. (2016). Priority-based heuristics for the multi-skill resource constrained project scheduling problem. *Expert Systems with Applications, 57,* 91–103. https://doi.org/10.1016/j.eswa.2016.03.017
33. Ferreira, C. (2001). Gene expression programming in problem solving. WSC6 tutorial. http://www.gene-expression-programming.com
34. Koza, J. R. (1993). Genetic programming: On the programming of computers by means of natural selection. *Biosystems, 33,* 69–73.
35. Peng, Y. Z., Yuan, C., Qin, X., Huang, J. T., Shi, Y. B. (2014). An improved gene expression programming approach for symbolic regression problems. *Neurocomputing, 137,* 293–301. https://doi.org/10.1016/j.neucom.2013.05.062
36. Zhong, J. H., Feng, L., Ong, Y. S.(2017). Gene expression programming: A survey. *IEEE Computational Intelligence Magazine, 12,* 54–72. https://doi.org/10.1109/MCI.2017.2708618
37. Zhang, L. P., Hu, Y. F., Tang, Q. H., Wang, C. J. (2022). Effective dispatching rules mining based on near-optimal schedules in intelligent job shop environment. *Journal of Manufacturing Systems, 63,* 424–438. https://doi.org/10.1016/j.jmsy.2022.04.019
38. Zhang, L., Li, Z., Wu, D. D., Krolczyk, G. (2019). Mathematical modeling and multi-attribute rule mining for energy efficient job-shop scheduling. *Journal of Cleaner Production, 241,* 118289. https://doi.org/10.1016/j.jclepro.2019.118289
39. Zhang, L. P., Tang, Q. H., Zheng, P. (2016). Adaptive dispatching rule for job shop scheduling problem via gene expression programming. ICIC Express Letters, *10,* 923–928.
40. Nie, L., Gao, L., Li, P. G., Zhang, L. P. (2011). Application of gene expression programming on dynamic job shop scheduling problem. *Proceedings of the 2011 15th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 291–295. Laussane, Switzerland. https://doi.org/10.1109/CSCWD.2011.5960088

41. Zhang, L. P., Hu, Y. F., Tang, Q. H., Li, J., Li, Z. X. (2021). Data-driven dispatching rules mining and real-time decision-making methodology in intelligent manufacturing shop floor with uncertainty. *Sensors, 21*. https://doi.org/10.3390/s21144836
42. Zhang, L. P., Tang, Q. H., Wu, Z. J., Wang, F. (2017). Mathematical modeling and evolutionary generation of rule sets for energy-efficient flexible job shops. *Energy, 138(3),* 210–227. https://doi.org/10.1016/j.energy.2017.07.005
43. Zhang, H., Zhou, J. L., Wu, X. (2021). An evolutionary algorithm for non-destructive reverse engineering of integrated circuits. *Computer Modeling in Engineering & Sciences, 127(3),* 1151–1175. https://doi.org/10.32604/cmes.2021.015462
44. Klein, R. (2000). Bidirectional planning: Improving priority rule-based heuristics for scheduling resource-constrained projects. *European Journal of Operational Research, 127,* 619–638. https://doi.org/10.1016/S0377-2217(99)00347-1
45. Chand, S., Huynh, Q. N., Singh, H. K., Ray, T., Wagner, M. (2018). On the use of genetic programming to evolve priority rules for resource constrained project scheduling problems. *Information Sciences, 432,* 146–163. https://doi.org/10.1016/j.ins.2017.12.013
46. Browning, T. R., Yassine, A. A. (2010). Resource-constrained multi-project scheduling: Priority rule performance revisited. *International Journal of Production Economics, 126,* 212–228. https://doi.org/10.1016/j.ijpe.2010.03.009