applied sciences

MDPI

Article

Multi-Objective Multi-Skill Resource-Constrained Project Scheduling Considering Flexible Resource Profiles

Xu Luo [1,2], Shunsheng Guo [1,2], Baigang Du [1,2,*], Xinhao Luo [1,2] and Jun Guo [1,2]

[1] School of Mechanical and Electronic Engineering, Wuhan University of Technology, Wuhan 430070, China; luoxu1983@whut.edu.cn (X.L.)
[2] Hubei Digital Manufacturing Key Laboratory, Wuhan 430070, China
* Correspondence: dbg767@163.com

Abstract: This paper addresses a novel multi-skill resource-constrained project scheduling problem with flexible resource profiles (F-MSRCPSP), in which the resource allocation of each activity consists of a certain number of discrete resources and is allowed to be adjusted over its duration. The F-MSRCPSP aims, therefore, to determine the flexible resource profile of each activity to minimize the make-span and total cost simultaneously. Then, a hybrid multi-objective fruit fly optimization algorithm is proposed to handle the concerned problem. In the proposed algorithm, two flexible parallel and serial schedule generation schemes are introduced, aiming to schedule activities and adjust allocated resource combinations. Additionally, two heuristic strategies are proposed to effectively select suitable resource combinations for activities. Moreover, a series of operators has been developed, including the rejoining operator, empirical re-arrangement operator, and empirical re-selection operator. These operators aim to accelerate the convergence speed and enhance the exploration of the proposed algorithm. Finally, the orthogonal test is used to select the optimal parameter combination, and comparative experiments based on tests with different scales are conducted, along with a *t*-test. The experimental results demonstrate that MOFOA-HS is effective in solving the F-MSRCPSP.

Keywords: multi-skill resource-constrained project scheduling; flexible resource profiles; multi-objective fruit fly optimization algorithm; heuristic strategy; improved operation operators

check for updates

Citation: Luo, X.; Guo, S.; Du, B.; Luo, X.; Guo, J. Multi-Objective Multi-Skill Resource-Constrained Project Scheduling Considering Flexible Resource Profiles. *Appl. Sci.* **2024**, *14*, 1921. https://doi.org/10.3390/app14051921

Academic Editor: Paolo Renna

Received: 8 January 2024
Revised: 5 February 2024
Accepted: 6 February 2024
Published: 26 February 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

As the most common production scheduling problem, the resource-constrained project scheduling problem (RCPSP) is widespread in various fields [1–3]. The RCPSP is meant to schedule activities without preemption under precedence constraints and limited resource availability to optimize the given objectives, such as the make-span and total cost [4–8]. In realistic situations, both the manpower and multipurpose machine master several skills, and the level of each skill is differentiated [9]. The execution of each activity requires a resource that masters a certain skill at a standard level [10]. Thus, the multi-skill resource-constrained project scheduling problem (MSRCPSP) is proposed to deal with these situations.

As an extension of the RCPSP, the MSRCPSP has attracted more and more attention of researchers recently [11]. Almeida et al. [6] developed a MSRCPSP with a new heuristic framework to cooperate with resource weight and activity grouping. Lin et al. [12] proposed a genetic programming hyper-heuristic approach to handle the MSRCPSP, where genetic programming as a high-level strategy managed the low-level heuristics. Dai et al. [13] used a function of the starting time and a specific deteriorating date to express the duration of each activity for the MSRCPSP. Tian et al. [14] studied a MSRCPSP with skill switches, where a new encoding scheme was used to reduce search space and a greedy-like decoding scheme was used to reorder activities to reduce skill switches. Hosseinian and Baradaran [15] proposed a MSRCPSP considering transfer times. Najafzad et al. [16] addressed a MSRCPSP considering manpower under shift differential payments and equipment with time-of-use

electricity tariffs. Ahmadpour and Ghezavati [17] studied a MSRCPSP considering project members' skill factor.

In the above studies on MSRCPSPs, most of them assumed that the resource allocation of the activity is normally given in advance and constant over its duration. However, it is often the case that in real projects, the resource allocation for the activity is unknown beforehand; it is generated in real-time according to the process of the scheduling plan. Relevant literature about RCPSPs considering flexible resource allocation is rather scarce. Naber and Kolisch [18] introduced a concept of the activity's "resource profile", in which resource quantities allocated to activities may vary for the entire duration, and the duration of each activity indeed resulted from the resource quantities at each time period during its processing. Based on this, Naber [19] studied a RCPSP with a flexible resource profile in continuous periods. Tritschler et al. [20] proposed a RCPSP considering a flexible resource profile in discrete time periods. However, there is one thing to point out: the above literature on flexible resource profiles is entirely limited to a single resource, without considering multi-skill resources. Considering the flexible resource profiles in MSRCPSP, it is not only more in line with a real project but also has greater flexibility in generating a scheduling plan, so as to obtain a more perfect scheduling plan. To the best of our knowledge, this paper is the first attempt to present the multi-skill resource-constrained project scheduling problem with flexible resource profiles (F-MSRCPSP).

Given the MSRCPSP is an NP-hard problem [21], as an extension of the MSRCPSP, the F-MSRCPSP is more complicated. It not only needs to consider the selection of multi-skill resources, but it must also adjust the resource allocation of each time period for each activity on the basis of the MSRCPSP. In recent years, many meta-heuristics algorithms have been applied to solve RCPSP-like optimization problems [22–31]. Among them, the multi-objective fruit fly optimization algorithm (MOFOA), as an effective optimization algorithm, has shown promising potential for solving complex engineering optimization problems. Inspired by the successful hybrid strategies and applications in the foregoing research, this paper proposes a hybrid MOFOA combined with heuristic strategies (MOFOA-HS). The solution representation scheme is designed to include an activity sequence list and a resource pattern list. Two flexible parallel and serial schedule generation schemes are proposed to arrange activities in sequence to enter the scheduling plan and adjust resource combinations for activities. Then, two heuristics are employed to select appropriate resource combinations for activities. In addition, a series of operators are adopted to enhance the exploration by searching the feasible solution space in different ways.

The main contributions of this paper are as follows:

(1) The existing problem formulation of the RCPSP is extended by considering the utilization of multi-skilled resources with flexible resource profiles. This is called a multi-skill resource-constrained project scheduling problem with flexible resource profiles (F-MSRCPSP).

(2) A new MOFOA-HS is proposed for the F-MSRCPSP. The solution representation scheme is designed to include an activity sequence list and a resource pattern list. Two parallel and serial schedule generation schemes are proposed, where two heuristic strategies are applied to select appropriate resource combinations for activities.

The rest of this paper is organized as follows: Section 2 describes the F-MSRCPSP. The proposed MOFOA-HS is discussed in Section 3. Section 4 presents the results and analysis of a series of experiments. Finally, some conclusions and prospects for future work are described in Section 5.

2. Problem Description and Model Setting

In this section, the problem description is first presented. Then, the model of the F-MSRCPSP is proposed. Moreover, an example is given to further illustrate the problem.

2.1. Problem Description

To enhance the description of the model, the symbols employed in this paper are explicitly defined in Table 1 for better comprehension.

Table 1. The notations used in the optimization model for the F-MSRCPSP.

Indices	
j	Index of the activity, where $j = 1, 2, \cdots, n_j$.
r	Index of the resource, where $r = 1, 2, \cdots, n_r$.
t	Index of the time period, where $t = 1, 2, \cdots, T$.
k	Index of the skill where, $k = 1, 2, \cdots, n_k$.
p	Index of the resource combination.
Parameters	
A	The set of activities.
R	The set of resources.
S	The set of skills.
n_j	Number of activities.
n_r	Number of resources.
n_k	Number of skills.
T	Discrete time horizon.
A_j	Activity j.
R_r	Resource r.
T_t	Time period t.
S_k	Skill k.
p_{jt}	Resource combination p of A_j at T_t.
c_r	The fixed-unit salary of R_r.
l_{jr}	Minimum working period of R_r in the execution process of A_j.
f_{max}	Project make-span.
AK_j^k	The requirement of S_k for A_j.
RK_r^k	The requirement of S_k that R_r supplies per time period.
$s_j f_j$	Start and finished period of A_j.
$\underline{q}_j\ \overline{q}_j$	The lower and upper usage bounds allocated to A_j for S_k.
l_{jr}	Working length of R_r in the execution cycle of A_j.
φ_{kt}	The set of current leftover resources that masters R_k at T_t.
w_{jkt}	Completed requirement of A_j for S_k at T_t.
Decision Variables	
D_{ij}	$D_{ij} = 1$, if A_i is the predecessor of A_j; else $D_{ij} = 0$.
x_{rkt}	Decision variable, $x_{rkt} = 1$, if R_r, use S_k at T_t; else $x_{rkt} = 0$.
y_{rt}	Decision variable, $y_{rt} = 1$, if R_r is assigned at T_t; else $y_{rt} = 0$.
z_{jrt}	Decision variable, $z_{jrt} = 1$, if R_r is assigned to A_j at T_t; else $z_{jrt} = 0$.

A project consists of a set of activities A with $A = \left\{A_1, A_2, \ldots, A_{n_j}\right\}$, where each activity $A_j \in A$ is processed without preemption. An activity cannot be executed until all of its predecessor activities are completed due to precedence constraints between some activities. The activities are accomplished by a set of renewable resources R with $R = \{R_1, R_2, \ldots, R_{n_r}\}$. Each resource $R_r \in R$ masters several skills with certain familiarity levels from the set of skills S with $S = \left\{S_1, S_2, \ldots, S_{n_k}\right\}$. Each activity $A_j \in A$ requires a certain unit of skill requirement rather than resources. Hence, the resource requirement of each activity is now transformed into the skill requirement. For each resource $R_r \in R$, it can be allocated to an activity A_j only if it masters the required skill of A_j. In addition, the unit salary c_r of each resource R_r is constant no matter which skill is performed.

The execution of each activity requires a resource combination p_j, which consists of several resources that match the skill demand of the activity. The resource combination may vary in terms of resource quantity or specific composition in each discrete time period $T_t \in T$ over the duration of the activity. The schedule for the F-MSRCPSP is to determine a start time period and a resource profile, which is composed of resource combinations

at each time period. The duration of the activity is unpredictable and depends on the allocated resource combination per time period during its processing. The resource profile observes the following constraints:

(1) The quantity of resource combination in the resource profile should be between the lower and upper usage bounds of the activity. In addition, it also cannot exceed the current leftover quantity of resources that meets the skill demand of the activity in each discrete time period over its duration:

$$\underline{q}_j \le q_{jkt} = \sum_{r \in R} z_{jrt} \le min(\overline{q}_j, \varphi_{kt} = n_{R_k} - \sum_{r \in R_k} y_{rt}); \forall t \in [s_j, f_j], j \in A \quad (1)$$

(2) The resource combination should remain unchanged for at least one contiguous time period:

$$l_{jr} = 1; \forall j \in A, r \in R \quad (2)$$

(3) The total planned skill requirement allocated to the activity must reach or exceed the skill requirement, because exceeding the skill requirement vs. the activity demands might be necessary in order to guarantee the previous constraint:

$$\sum_{t \in [s_j, f_j]} \sum_{r \in p_{jt}} x_{rkt} \cdot z_{jrt} \cdot RK_r^k \ge AK_j^k; \forall k \in S, j \in A \quad (3)$$

2.2. Problem Formulation

The mathematical model of the F-MSRCPSP is as follows.

$$Min\ Makespan = \max_{j \in A}(f_j) \quad (4)$$

$$Min\ Cost = \sum_{r=1}^{n_r} \sum_{t=1}^{T} c_r y_{rt} \quad (5)$$

Subject to:

$$\sum_k x_{rkt} \le 1; \forall j \in A, r \in R, t \in T \quad (6)$$

$$AK_j^k RK_r^k x_{rkt} z_{jrt} > 0; \forall j \in A, r \in R, k \in S, t \in [s_j, f_j] \quad (7)$$

$$0 < f_i < s_j; \forall j \in A, i \in P_j \quad (8)$$

$$z_{jrt} = 1; \forall j \in A, \exists r \in R, t \in T \quad (9)$$

$$\sum_{r=1}^{n_r} \sum_{k=1}^{n_k} x_{rkt} z_{jrt} \le 1; \forall t \in T \quad (10)$$

The first objective function in Equation (4) minimizes the project make-span. The second objective function in Equation (5) minimizes the total cost, which consists of the cost of using resources in each time period. Constraint (6) guarantees that only one skill is required to complete an activity, and constraint (7) ensures that only the resources that master the required skill of the activity can be allocated to the activity in each discrete period. The precedence constraints among activities of the project are ensured by constraint (8). Each activity should be processed once, and each resource can only use one skill to deal with one activity in one time period. The constraints are shown by constraints (9) and (10), respectively.

To further illustrate the F-MSRCPSP, an illustrative project with five activities, three resources, and three skills is described as follows. Table 2 represents the skill distribution and corresponding mastery level of each resource, as well as its unit salary. Table 3 shows the lower and upper usage bounds, the skill requirements, and their predecessors. The comparison Gantt chart of a feasible solution to the MSRCPSP and F-MSRCPSP is depicted in Figure 1. In the MSRCPSP, a single resource is allocated to an activity and remains

constant during the processing of the activity, such as allocate R_2 to A_1, R_3 to A_3, R_1 to A_5, and so on. In the F-MSRCPSP, a resource combination could be allocated to an activity without violating constraints and may change with the implementation of activities and the use of resources, such as for the resource combination, A_1 changes from R_2 to $[R_2, R_3]$ during the 12th time period, A_3 changes from R_1 to R_3 during the 7th time period, and A_5 also changes the resource combination in the 8th and 11th time period, respectively. Compared with the MSRCPSP, the F-MSRCPSP can obtain a better scheduling plan with a shorter make-span and lower total cost.

Table 2. Example of resource.

		Skill		
Resource	**Salary**	S_1	S_2	S_3
R_1	56	0	1	3
R_2	53.6	2	0	3
R_3	28.9	2	1	0

Table 3. Example of activity.

	Usage Bound		Skill			
Activity	$\underline{q}_{jk}$	$\overline{q}_{jk}$	S_1	S_2	S_3	**Predecessors**
A_1	1	2	18	0	0	A_0
A_2	1	1	12	0	0	A_0
A_3	1	2	0	7	0	A_0
A_4	1	1	0	0	12	A_0
A_5	1	2	0	8	0	A_4

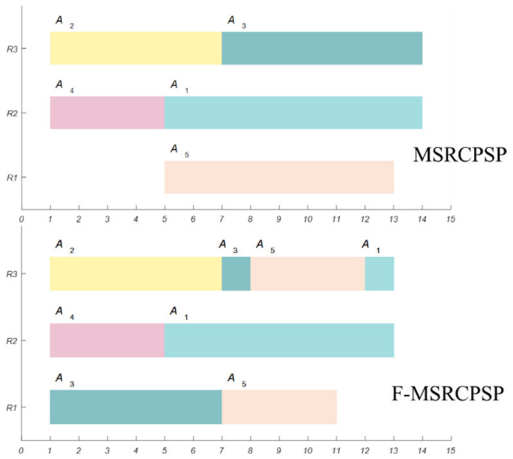


Figure 1. Gantt chart of two models.

3. The MOFOA-HS for the Multi-Objective F-MSRCPSP

In this section, we present a hybrid multi-objective fruit fly optimization algorithm combined with heuristic strategies (MOFOA-HS) to solve the concerned problem. In the MOFOA-HS, each individual is represented with the solution representation in Section 3.1 and evaluated with the schedule generation scheme in Section 3.2 by applying heuristic strategies in Section 3.3. Population initialization is in Section 3.4, and a series of operators are adopted to enhance exploration and exploitation of the population in Section 3.5. The main procedure of the MOFOA-HS is detailed as follows:

Step 1: Parameter setting. Population size NS, number of searches per individual in the local search maximum S, the learning rate of the experience base c.

Step 2: Population initialization. The double-layer solution representation is based on activity sequence and resource allocation; the length of both layers is the number of activities.

Step 3: Rejoining operator. S individuals are generated by searching the solution space near each individual for S times, and such generated individuals construct the new population.

Step 4: Contraposition selection operator. Randomly select an activity to replace the two heuristic strategies for resource combination selection with the other for each individual in the new population.

Step 5: Local balanced operator. Calculate the objective values of the new population. Merge original and new population and sort them by a non-dominated sorting technique (NST). Then, select the first NS individuals to form the population and continue the following search.

Step 6: Empirical rearrangement operator. The operator is implemented by the experience base, which is completed by the dominant individuals of the initial population. The experience base consists of two parts: the activity sequence experience base (ASB) and the resource pattern knowledge base (RPB). At every iteration, update the experience base of the activity sequence with dominant individuals of the current population. Then, select a continuous segment in the activity sequence list of each individual and reorder it using the information of the activity sequence experience base, which forms the new population.

Step 7: Empirical reselection operator. Similarly, update the experience base of resource patterns with dominant individuals of the original population. Then, reselect the resource pattern of each activity for each individual of the new population according to the empirical value of resource pattern choice in the resource pattern experience base on a roulette strategy.

Step 8: Global balanced operator. After completing the above two searches, each individual generates an individual through experience learning, resulting in a new population of NS. Therefore, we also need to evaluate NS new individuals. Then, update the population by selecting NS individuals from the merged population with NST, and record the Pareto set simultaneously.

Step 9: Update the Pareto set. Merge original and newly generated Pareto set, and select non-dominated individuals as new Pareto set.

Step 10: Iteration criteria. With the iteration of the population, it can be guaranteed that the Pareto set will be better than the previous generation or remain unchanged. Therefore, when the Pareto set remains unchanged in a certain number of consecutive iterations, it can be considered that the algorithm has converged, and the iteration stops. If the Pareto set is superior to the previous generation of the Pareto set, this indicates that the population is still close to the theoretical optimal solution. Go back to Step 3.

3.1. Solution Representation

The F-MSRCPSP consists of two sub-problems: activity sequencing and resource allocating. Each solution consists of two lists, i.e., an activity sequence list (λ) and a resource pattern list (σ), as shown in Figure 2.

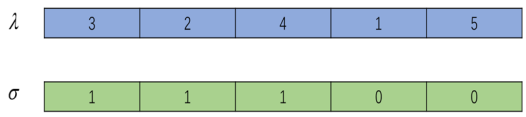


Figure 2. Representation of a feasible solution.

The parameter λ determines the order to process the activity, where each element denotes a specific activity. σ determines which strategy the activity chooses from two designed heuristics strategies in Section 3.3. In detail, each activity chooses the greedy selection strategy (GSS) when the resource pattern value is 0; otherwise, select the non-greedy selection strategy (NGSS). In order not to violate all the constraints, it is ensured that the predecessors of each activity in the list must be in front of it during initialization.

3.2. Schedule Generation Scheme

Each solution obtains two results through two scheduling generation schemes in Sections 3.2.1 and 3.2.2. To determine the dominant solution, a comparison method is utilized. If one solution clearly outperforms another across all criteria, it is selected as the superior choice. However, if there is no definitive dominance, both solutions are retained. Importantly, the scheduling generation schemes implemented ensure that all generated solutions are capable of producing feasible scheduling outcomes.

3.2.1. Flexible Parallel Schedule Generation Scheme (FPSGS)

The current partial schedule with an associated decision point is extended in each iteration based on the time period. Each iteration is a decision point at T_t with a set A_t^f of finished activities that have been completed up to T_{t-1}, a set A_t^a of active activities that have already started the execution but are not yet completed at T_t, as well as a set A_t^e of eligible activities that have not yet started and whose predecessors have been completed up to T_{t-1}. For each activity A_j, $\underline{p}_j$ is the minimum cost resource combination p of A_j; $\underline{q}_{jk}$ and $\overline{q}_{jk}$ represent the lower and upper usage bounds, respectively; q_{jkt} denotes the resource quantity; ρ_j indicates the current allocated resource combination; w_{jkt} expresses the completed requirement; ξ_{jkt} shows the remaining requirement; and φ_{kt} means the set of current leftover resources. In addition, $\xi_{\rho k}$ states the requirement of S_k that p supplies per time period. After each operation, variables and sets are constantly updated. The pseudo code of the FPSGS is depicted as in Algorithm 1.

Step 1: Initialize T_t, A_t^e, A_t^a, A_t^f.

Step 2: Execute the flexible resource profiles' parallel schedule. Judge whether all activities have been completed: if yes, the resource scheduling plan of all activities is completed; if not, continue to execute the flexible resource profiles' parallel schedule at the current time period.

Step 3: Update A_t^e. Judge whether all predecessor activities of A_j in sequence of λ have been completed. If so, move A_j from λ to A_t^e; if not, stop judging and go to Step 4.

Step 4: Continue active activities. In order to prevent the execution interruption caused by resource preemption during the activity, the basic principle of resource allocation is conservative in this step. Therefore, the operation of this step is to remain constant or delete part of the resources in the resource combination p'_j of the last time period. The A_j will continue the p'_j. In particular, there are some activities that adopt the GSS. The ξ_{jkt} is less than the $\xi_{\rho k}$ of p'_j, and the quantity of p'_j is greater than its lower usage bound, which means that the activity could be completed with fewer resources. For the resource of these activities, it is necessary to recombine the p_j: Firstly, obtain the set Ω that consists of all possible combinations whose number of resources is larger than $\underline{q}_j$ from p_j. Then, eliminate the combinations whose $\xi_{\rho k}$ is less than ξ_{jkt} from Ω with GSS. Finally, obtain p_j from the Ω. Move the activity from A_t^a to A_t^f if the remaining requirement is met by p_j.

Step 5: Start eligible activities. Use the remaining available resources to start the activity with $\underline{q}_j$. For activities that adopt the GSS, judge the size relationship between the quantity of φ_{kt} and $\underline{q}_j$. If $n_{\varphi_{kt}} \geq \underline{q}_j$, obtain the set Ω that consists of all possible combinations whose resource number is equal to $\underline{q}_j$ from φ_{kt}, then extra allocate p_j from Ω to the activity. If the NGSS is adopted, allocate $\underline{p}_j$ to A_j after ensuring that all resources of $\underline{p}_j$ are available. Update related sets and variables, and move A_j from A_t^e to A_t^a if p_j is not empty.

Step 6: Utilize leftover resources. After completing the resource allocation in Step 4 and 5, some eligible activities are added to the schedule, and all the active activities continue to be executed. However, due to the non-greedy manner, there are available resources that are not fully utilized. Hence, in this step, we take advantage of these remaining resources by expanding the resource combination for activities that belong to A_t^a. In order for the activities to participate in secondary resource allocation to utilize leftover resources, there must be some combination of adopting the GSS for the available leftover resources

remaining, and the quantity of p_j is no more than $\overline{q}_j$. First, in order to control the number of resource combinations not to exceed $\overline{q}_{jk}$ after expansion, the set Ω consists of all possible combinations whose quantity of resources is in the range of 1 and $min(n_{\varphi_{kt}}, \overline{q}_j - n_{p_j})$ from φ_{kt}. If the activity can be completed at T_t, that is to say, the maximum $\xi_{\rho k}$ in the Ω is not less than the ξ_{jkt}, then filter out the resource combination p whose $\xi_{\rho k}$ is less than ξ_{jkt}. Afterwards, additionally allocate the combination from Ω with the GSS to the activity. Move the activity from A_t^a to A_t^f if ξ_{jkt} is met.

Step 7: Enter the next time period. The resource allocation at the current time period T_t has ended and enters the next time period T_{t+1}.

Algorithm 1 The pseudo code of the FPSGS

1. Initializing parameters: $T_t = 1; A_t^e = []; A_t^a = []; A_t^f = []$
2. **While** there are activities not completed yet **do**
3. **For** A_j in sequence of λ **do**

   If $A_{j'} \notin A_t^f, \exists A_{j'} \in P_j$ **then**
   Break
   Else
   move A_j from λ to A_t^e
4. **For** A_j in sequence of A_t^a **do**

   $p_j = p'_j$
   If $\sigma[j] == 0$ **and** A_j could be completed with fewer resources **then**
   For p in sequence of Ω **do**
   If $\xi_{pk} < \xi_{jkt}$ **then**
   Eliminate the p from Ω
   Get p_j from Ω
   If A_j is completed **then**
   Move A_j from A_t^a to A_t^f
5. **For**A_j in sequence of A_t^e **do**

   $p_j = \varnothing$
   If $\sigma[j] == 0$ **then**
   If $n_{\varphi_{kt}} \geq \underline{q}_j$ **then**
   Get p_j from Ω
   Else if All resources of $\underline{p}_j$ are available **then**
   $p_j = \underline{p}_j$
   If $p_j \neq \varnothing$ **then**
   Move A_j from A_t^e to A_t^a
6. **For**A_j in sequence of A_t^a **do**

   If A_j could continue to allocate resources **then**
   If $max(\xi_{pk}) \geq \xi_{jkt}, \forall p \in \Omega$ **then**
   For p in sequence of Ω **do**
   If $\xi_{pk} < \xi_{jkt}$ **then**
   Eliminate the p from Ω
   Get the combination from Ω with GSS
   If $\xi_{jkt} \leq 0$ **then**
   Move A_j from A_t^a to A_t^f
7. $T_t = T_t + 1$

To further illustrate FPSGS, take the example in Section 2.2. Initialize $T_t = 1$, $A^a = ()$, $A^f = ()$, $A^e = (4, 3, 2)$, because there are precedence constraints between A_4 and A_5. Skip

Step 4, because $A^a = ()$. During Step 3, firstly allocate resources to A_4 due to their position in λ. As $\sigma(4) = 1$, allocate R_2 to A_4 with NGSS. Allocate R_1 to A_3 from R_1 and R_3 with NGSS, because $\sigma(3) = 0$, and start eligible activities with the lower usage bound. Similarly, allocate the remaining resource R_2 to A_2. End this step with $A^e = ()$ and $A^a = (4, 3, 2)$. Then, skip Step 6, because there are no resources left. In time period 2, continue the previous resource combination for A^a. A_4 is completed and transfers from A^a to A^f in time period 4. Then, in time period 5, due to the completion of A_4, A_5 could be eligible. Update $A^e = (5, 1)$ and $A^a = (3)$. Allocate R_2 to A_1, because both R_1 and R_3 required for A_5 are occupied. A_2 is completed in time period 6. In time period 7, allocate R_3 to A_5 in Step 5, and A_3 is completed. In time period 8, utilize leftover R_2 for A_5 in Step 6, because $\sigma(5) = 0$, and A_5 does not reach its upper usage bound. In time period 11, in Step 4, A_5 could be completed with fewer resources, so the occupation of R_1 is released. The A_1 utilizes leftover A_3 in Step 6 and is completed in time period 12. Given λ = (4,2,3,1,5) and σ = (0, 0, 0, 1, 0), the Gantt chart generated using the FPSGS is shown in Figure 3.

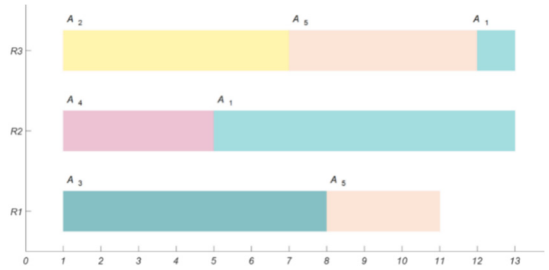


Figure 3. The Gantt chart generated by the FPSGS.

3.2.2. Flexible Serial Schedule Generation Scheme (FSSGS)

In each iteration, the current partial schedule is extended by scheduling one activity. Each iteration is a decision point of A_j with the latest end time period T_{pl} and the earliest time period T_{re} of each activity. The remaining parameters are the same as FPSGS. The pseudo code of the FSSGS is shown in Algorithm 2.

Step 1: Initialize λ, A^e. Select the first activity in order from the activity sequence list λ as A_j and delete it from λ. Then, initialize the set A^e: select the activities without predecessor activity constraints or whose predecessor activities have been completed from λ and add them to A^e.

Step 2: Find the earliest start time without violating the constraints. The A_j must be started after all its predecessor activities are completed if there are predecessor activities. Since the quantity of available resources should not be less than the lower usage bound, the earliest start time of A_j should be the largest of the next time period of the latest end time period of all the predecessor activities. Meanwhile, the earliest time period should also meet the lower usage bound.

Step 3: Resource allocation. Judge whether the activity has been completed. If not, resources are allocated to activities in different ways.

(a) Greedy selection strategy. For the A_j that adopts the GSS, judge whether the quantity of φ_{kt} is not less than $\underline{q}_j$. If so, obtain the set Ω, which consists of all possible combinations whose quantity of resources is in the range of $\underline{q}_j$ and $min(n_{\varphi_{kt}}, \overline{q}_j)$ from φ_{kt}. Filter the p whose ξ_{pk} is less than ξ_{jkt} when the activity could be completed, then allocate p_j from Ω to A_j. Otherwise, abandon all previous resource allocation of A_j and reallocate the resource profile from the next time period.

(b) Non-greedy allocation strategy. For the A_j adopting the NGSS, allocate the $\underline{p}_j$ to A_j after ensuring that all resources in $\underline{p}_j$ are available. If there are any resources in $\underline{p}_j$ occupied by other activities, abandon all previous resource allocation of A_j, and reallocate resource profile from the next time period.

Algorithm 2 The pseudo code of the FSSGS

1. $A^e = []$
For $A_{j'}$ in sequence of λ **do**
  If $A^e \cap P_{j'} \neq \varnothing$ **then**
    Break
  Else
   $A^e = [A^e, A_{j'}]$
2. **If** $P_j \neq \varnothing$ **then**
 $T_{pl} = \max_{j' \in P_j}(f_{j'}) + 1$
  Else
    $T_{pl} = 1, T_{re} = 1$
 While $n_{\varphi_{kt}} < \underline{q}_j$ **do**
    $T_{re} = T_{re} + 1$
  $T_t = max(T_{pl}, T_{re})$
3. **While** $\xi_{jkt} > 0$ **do**

(a) **if** $\sigma[j] == 0$ **then**

if $\varphi_{kt} \geq \underline{q}_j$ **then**
if $\varphi_{kt} \geq \overline{q}_j$ **and** $max(\xi_{pk}) \geq \xi_{jkt}, \forall p \in \Omega$ **then**
for p in sequence of Ω **do**
if $\xi_{pk} < \xi_{jkt}$ **then**
Eliminate the p from Ω
Get p_j from Ω
else
Initialize the resource profile and requirement of the activity

(b) **else**

if All resources of $\underline{p}_j$ are available **then**
$p_j = \underline{p}_j$
else
Initialize the resource profile and requirement of the activity
$T_t = T_t + 1$

To further illustrate the FSSGS, take the example in Section 2.2 again. Firstly, resource scheduling is carried out for A_2 due to its position in λ. In the Step 2, the latest end time period T_{pl} of all the predecessor activities is 1, and the earliest time period T_{re} to meet the lower usage bound is 1, so the earliest start time $T_t = max(T_{pl}, T_{re})$ is 1. As $\sigma(2) = 1$, allocate R_3 to A_2 from R_2, R_3, and $[R_2, R_3]$ with the NGSS. Then, repeat Step 3a until the A_2 is completed in time period 6. Then, for A_4, T_t is also 0, and allocate R_2 to A_4 with NGSS, because $\sigma(4) = 1$. Repeat Step 3b until the A_4 is completed in time period 4. For A_3, allocate R_1 in time period 1–6, and then allocate R_3, because the occupation of R_3 is released in time period 7, and R_3 is considered the optimal combination among R_1, R_3, and $[R_1, R_3]$ with the NGSS. After A_3 is completed, A_1 starts in T_t which is 5. Allocate R_2 in time periods 5–6, and then allocate $[R_2, R_3]$, because the occupation of R_3 is released in time period 7. Finally, the A_5 starts in time period, 7 because T_{re} is 7 and T_{pl} is 5 due to precedence constraints between A_4 and A_5. Similarly, only allocate R_1 in time period 7–10, and then allocate $[R_1, R_3]$, because the occupation of R_3 is released in time period 11. Finally, the schedule results in a make-span of 12. Given λ= (2,4,3,5,1) and σ= (0,0,0,1,0), the Gantt chart generated using the FPSGS is shown in Figure 4.

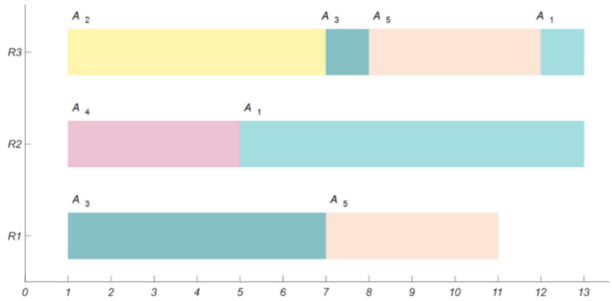


Figure 4. The Gantt chart generated by the FSSGS.

3.3. Heuristic Strategies for Resource Combination Selection

The idea of a flexible resource profile is not to allocate a constant resource combination to an activity before its execution but to keep adjusting the resource combination at different time periods during its execution according to a combination of factors, i.e., the remaining requirement of the activity, the implementation of other parallel activities, and the quantities and supple value of available resources. Therefore, it is inevitable that there will be resource conflict, in which parallel activities compete for a certain resource at the same time period. The greedy selection strategy (GSS) is designed to solve the resource conflict and avoid the randomness of resource combinations' allocation to activities.

The GSS is bound to ensure that a feasible resource combination is allocated to the activity, so it may appear that an inappropriate resource combination is allocated to the activity. For example, R_1 and R_3 have the same skill requirement in S_2, and both of them are 1, but the salary is different, i.e., 56 and 28.9, respectively. Therefore, for an activity that needs S_2, it is obvious that R_3 should be selected if only one is selected. However, if R_1 is idle and R_3 is occupied at the current time period, the GSS will inevitably select R_1, which greatly increases the cost. Therefore, we propose a non-greedy selection strategy (NGSS) to delay the scheduling of the activities.

3.3.1. Greedy Selection Strategy

The idea of the GSS is to calculate the two weight values of each feasible resource combination and then select the best one from all feasible resource combinations by using TOPSIS [32] to analyze the weight values of each combination. The two calculation formulas of the weight value are defined as follows.

Calculation formula (1):

Calculate the overall cost $a = \sum_{r \in \rho_{jt}} c_r$ of the resource combination p_{jt}. The larger a is, the higher the cost value of p_{jt}, and the lower the priority; otherwise, the priority is higher.

Calculation formula (2):

$$\text{Calculate the complexity} b = \sum_{k \in S} \left\{ \frac{\sum_{j' \in A} \sum_{r \in \rho_{jt}} RK_r^k \cdot \xi_{j'kt} \cdot z_{jrt}}{\sum_{r \in R'} RK_r^k} \right\} \tag{11}$$

where R' is the set of idle resources in the current time period. It includes three aspects of resource combination: the diversity and mastery level of skills mastered by each resource of the combination, as well as the degree of skill demand of the active and eligible activities of the project. The larger b is, the higher the impact on the start or execution of other activities, and the lower the priority; otherwise, the lower the impact, the higher the priority.

3.3.2. Non-Greedy Selection Strategy

In order to achieve the goal of non-greedy selection, it is not necessary to start as soon as possible but at as low a cost as possible. We determine a fixed resource combination with the lowest cost and then give up scheduling if there are occupied resources. By analyzing

the characteristics of the F-MSRCPSP, we obtain a property that the total cost is minimized if and only if the resource combination with the lowest sum of unit salary for each activity is allocated.

Property 1. *Cost*(Π) *is the lowest cost, where* Π *expresses a solution if and only if* $\forall j, \Pi(j) = \arg\min_{p_j \in \Omega_j} (\frac{AK_j^k}{\sum_{r \in p_j} RK_r^k} \cdot \sum_{r \in p_j} c_r)$.

Proof.

Sufficiency.

$$\begin{aligned} Cost(\Pi) &= \sum_j C_{\Pi(j)} = \sum_j \sum_t \sum_{r \in p_j} c_r \\ &= \sum_j (d_j \cdot \sum_{r \in p_j} c_r) = \sum_j (\frac{AK_j^k}{\sum_{r \in p_j} RK_r^k} \cdot \sum_{r \in p_j} c_r) = \sum_j \min_{p_j \in \Omega_j} (\frac{AK_j^k}{\sum_{r \in p_j} RK_r^k} \cdot \sum_{r \in p_j} c_r) \end{aligned} \tag{12}$$

Therefore, *Cost*(Π) is the optimal cost.

Necessity. By contradiction, suppose there exists a solution Π', such that $Cost(\Pi') < Cost(\Pi)$.

$$Cost(\Pi') = \sum_j \sum_t \sum_{r \in p'_j} c_r = \sum_j \frac{AK_j^k \cdot \sum_{r \in p'_j} c_r}{\sum_{r \in p'_j} RK_r^k} < Cost(\Pi) = \sum_j \frac{AK_j^k}{\sum_{r \in p_j} RK_r^k} \cdot \sum_{r \in p_j} c_r \tag{13}$$

which means $\exists\, j$, such that $\frac{AK_j^k}{\sum_{r \in p'_j} RK_r^k} \cdot \sum_{r \in p'_j} c_r < \frac{AK_j^k}{\sum_{r \in p_j} RK_r^k} \cdot \sum_{r \in p_j} c_r$.

This contradicts the condition that $\forall j, \Pi(j) = \arg\min_{p_j \in \Omega_j} (\frac{AK_j^k}{\sum_{r \in p_j} RK_r^k} \cdot \sum_{r \in p_j} c_r)$ and completes the proof. □

Therefore, the NGSS means that the specific resource combination is given for each activity before execution, and it remains unchanged in each time period of the whole execution. In addition, all resources of the given resource combination need to be idle before they can be allocated to the activity. Once allocated to the activity, they will be occupied until the activity is completed. The principle of minimum cost according to the above is the only criterion for the selection of the resource combination that can reduce the cost. Flatten the Pareto front curve, and approach the optimal solution more closely.

$$\underline{p}_j = \arg\min_{p_j \in \Omega_j} (\frac{AK_j^k}{\sum_{r \in p_j} RK_r^k} \cdot \sum_{r \in p_j} c_r) \tag{14}$$

The generated minimum cost resource combination associated with each activity is unchanged during the execution of the algorithm and can be computed in advance.

3.4. Population Initialization

During population initialization, a multiple individual strategy is adopted. The population consists of *NS* individuals, which are represented by a two-list, i.e., activity sequence list λ and resource pattern list σ. In addition, employ the Pareto archive (PA) to store and update the Pareto solutions generated by each iteration. In order to realize the uniform distribution of the solution in the solution space but not affect the quality of the solution, we adopt the random and the heuristic strategies, as well as propose an activity list extension algorithm to generate the activity sequence list.

For the resource pattern list, there are three initialization methods: GSS is chosen for all activities of the solution ($\sum_{j\in J}\sigma(j)=0$); all activities of the solution choose NGSS ($\sum_{j\in J}\sigma(j)=n_j$); some activities choose GSS, and the others choose NGSS ($0<\sum_{j\in J}\sigma(j)<n_j$). Randomly select one of the three initialization methods to a generate resource pattern list.

The basic idea of the activity list extension algorithm is that an activity is selected in sequence from the candidate set and inserted into the list, where the constraint is not violated until all the activities are ordered. The candidate set A' is obtained by disorderly arranging the active set A. Before starting the algorithm, we need to classify all activities according to P_j and S_j of activities and divide them into the following four activity categories: the set $S1(P_j\neq\varnothing, S_j=\varnothing)$ with predecessor constraints only, the set $S2(P_j=\varnothing, S\neq\varnothing)$ with successor constraints only, the set $S3(P_j=\varnothing, S_j=\varnothing)$ with both predecessor and successor constraints, and the set $S4(P_j\neq\varnothing, S_j\neq\varnothing)$ with no constraint. The pseudo code of the activity sequence list extension algorithm is shown in Algorithm 3.

Algorithm 3 Activity sequence list extension algorithm

1. $\lambda=[]$
2. **For** A_j in sequence of A' **do**
3. **If** $A_j\in S1$ **then**
   If $P_j\cap\lambda\neq\varnothing$ **then**
   $I_j\in[\max_{j'\in P_j\cap\lambda} I_{j'}+1, n_\lambda+1]$
   Else
   $I_j\in[1, n_\lambda+1]$
4. **If** $A_j\in S2$ **then**
   If $S_j\cap\lambda\neq\varnothing$ **then**
   $I_j\in[1, \min_{j'\in S_j\cap\lambda} I_{j'}+1]$
   Else
   $I_j\in[1, n_\lambda+1]$
5. **If** $A_j\in S3$ **then**

$I_j\in[1, n_\lambda+1]$

6. **If** $A_j\in S4$ **then**
   If $P_j\cap\lambda\neq\varnothing$ **then**
   If $S_j\cap\lambda\neq\varnothing$ **then**
   $I_j\in[\max_{j'\in P_j\cap\lambda} I_{j'}+1, \min_{j''\in S_j\cap\lambda} I_{j''}]$
   Else
   $I_j\in[\max_{j'\in P_j\cap\lambda} I_{j'}+1, n_\lambda+1]$
   Else
   If $S_j\cap\lambda\neq\varnothing$ **then**
   $I_j\in[1, \min_{j'\in S_j\cap\lambda} I_{j'}]$
   Else
   $I_j\in[1, n_\lambda+1]$
7. $\lambda=[\lambda(1:I_j-1), A_j, \lambda(I_j:n_\lambda)]$

Step 1: Initialize the activity list. The activity list λ is initialized as $\varnothing$.

Step 2: Determine where the activity is inserted, and extend the activity list. Select an activity from A' that is the random one of A in order to achieve the purpose of randomness and judge its activity category. Due to the different activity categories of 3–6, four disjunct cases apply.

Step 3: The activity A_j belongs to $S1$. The insertion position I_j of A_j can be any position after the last predecessor activity of A_j if it is in λ. Otherwise, it can be anywhere.

Step 4: The activity A_j belongs to $S2$. The insertion position I_j of A_j can be any position before the first successor activity of A_j if it is in λ. Otherwise, it can be anywhere.

Step 5: The activity A_j belongs to $S3$. The insertion position I_j of A_j can be anywhere in λ.

Step 6: The activity A_j belongs to $S4$. The insertion position I_j of A_j should be any position anywhere between the last predecessor activity and the first successor activity of A_j in λ. If there is only no successor activity in λ, the procedure is similar to Step 3; if there is only no predecessor activity in λ, the procedure is similar to Step 4; if neither, the procedure is similar to Step 5.

Step 7: Insert the activity A_j into the specified position I_j in the list, and then go back to Step 3 until all activities are in λ.

3.5. Improved Operation Operators

3.5.1. Rejoining Operator

An effective method to implement a neighborhood search of the activity sequence is proposed, which employs an operator by rejoining any activity in λ without violating the activity constraint. Randomly select an activity A_j from λ, and rejoin the activity to any position within the scope that is determined by the category of the activity. Then, the following four disjunctive cases apply:

$$I'_j \in (\max_{j' \in P_j}(I_{j'}), J+1],\ A_j \in S1 \tag{15}$$

$$I'_j \in [1, \min_{j \in S_{j'}}(I_{j'})],\ A_j \in S2 \tag{16}$$

$$I'_j \in (\max_{j' \in P_j}(I_{j'}), \min_{j \in S_{j'}}(I_{j'})],\ A_j \in S3 \tag{17}$$

$$I'_j \in [1, J+1],\ A_j \in S4 \tag{18}$$

Figure 5 shows a project with four situations. λ and the local search of λ are shown, where the black arrows represent a selectable location rejoined in λ for four categories of activities.

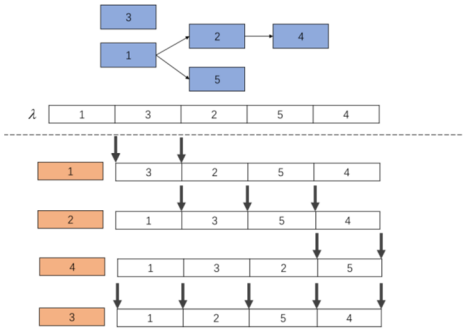


Figure 5. The rejoining operator.

3.5.2. Empirical Re-Arrangement Operator

The initialization of the activity sequence experience base is as follows:

$$ASB_0(j, j') = \frac{\sum_{n=1}^{NE}\left(I_{n,j'} - I_{n,j}\right)}{NE}; \forall j, j' \in A \tag{19}$$

where 0 represents the initialization status of ASB, NE is the number of superior individuals in the Pareto set to provide the experience value about the activity sequence, and $I_{n,j}$ is the position of A_j in the λ of the $n-th$ individual.

Record the relative position of one activity and the other activities in ASB. In each generation, relative positions are updated with the experiences of the activity sequence provided by NE superior individuals among the population, which is as follows:

$$ASB_g(j,j') = (1-c)ASB_{g-1}(j,j') + \frac{c}{NE}\sum_{n=1}^{NE}\left(I_{n,j'} - I_{n,j}\right) \tag{20}$$

where g represents the generation number, and c is the updating rate of the experience base.

Each individual adjusts its activity sequence to obtain a new activity list λ' under the guidance of ASB. The step is as follows:

Step 1: Randomly generate two numbers $1 \le p < q \le n_j$.

Step 2: Generate λ' by inheriting the subsequence from λ as Equation (21).

$$\pi'(j) = \pi(j); \forall j \in [1, p-1] \cup [q+1, n_j], \pi' \in \lambda', \pi \in \lambda \tag{21}$$

Step 3: Calculate the sum RP_j of the relative position value in the knowledge base of the activity λ' to the non-sorted activities as Equation (22).

$$RP_j = \sum_{j'} ASB\left(\pi(j), \pi\left(j'\right)\right); \forall j, j' \in [p, q] \tag{22}$$

Step 4: Fill in the position from p to q in λ' from small to large according to the RP_j.

3.5.3. Empirical Re-Selection Operator

The experiential possibility of the resource pattern sub-knowledge base (RPB) is initialized as follows:

$$RPB_0 = \frac{\sum_{n=1}^{NE}\rho_n}{NE} \tag{23}$$

where ρ_n is the resource pattern list of the $n-th$ individual in NE superior individuals, and RPB_0 represents the probability of elite individuals selecting the GSS or the NGSS in the initial population phase.

In each generation, update the experiences of the resource pattern provided by superior individuals among the population as follows:

$$RPB_g = (1-c)RPB_{g-1} + \frac{c}{NE}\sum_{n=1}^{NE}\rho_n \tag{24}$$

4. Experiment Study

This paper chooses the intelligent multi-objective project scheduling environment (iMOPSE) benchmark dataset [21] for numerical tests, which is available at http://imopse.ii.pwr.wroc.pl/ (accessed on 8 January 2024). In order to adapt to the F-MSRCPSP problem, the following modifications are made to the definition of parameters in instances: the duration of the activity in instances is equal to AK_j^k; the skill level of resources is equal to RK_r^k; the $\underline{q}_{jk}$ and $\overline{q}_{jk}$ are randomly generated; and it is ensured that $\overline{q}_{jk}$ will not exceed the number of resources with corresponding skills. In addition, the dataset consists of 42 project instances, including six small sample instances with 10 and 15 activities, as well as 36 large sample instances with 78, 100, and 200 activities. Due to the lack of medium sample instances, we use intercept activities and preserve other parameters of certain large sample instances to generate them. The hardware and software platform parameters are listed as follows: Windows 10, AMD Ryzen 7 4800U CPU, 1.8 GHz, 16G RAM, and MATLAB R2016b. We select and modify 21 instances with various scales as experimental samples. In all the subsequent tests, each algorithm will run independently ten times.

4.1. Parameter Setting

There are three key parameters in the MOFOA-HS, i.e., the number of individuals (*NS*), the number of searches per individual in the rejoining operator (*S*), and the learning rate of the experiences (*c*). To investigate the influence of these parameters on the MOFOA-HS, the design-of-experiment (DOE) method [33] is employed. Table 4 shows the four levels of each parameter, and the orthogonal array $L_{16}(4^3)$ is selected. This is shown in Table 5.

Table 4. Factor levels.

Parameters	Factor Level			
	1	**2**	**3**	**4**
NS	10	50	100	150
S	1	5	10	15
c	0.05	0.1	0.15	0.2

Table 5. Orthogonal array and RVs.

Experiment No.	Factor Level			RV (%)
	1	**2**	**3**	
1	1	1	1	0.011
2	1	2	2	0.016
3	1	3	3	0.000
4	1	4	4	0.027
5	2	1	2	0.045
6	2	2	1	0.059
7	2	3	4	0.050
8	2	4	3	0.077
9	3	1	3	0.067
10	3	2	4	0.092
11	3	3	1	0.111
12	3	4	2	0.100
13	4	1	4	0.062
14	4	2	3	0.048
15	4	3	2	0.110
16	4	4	1	0.123

For each parameter combination in Table 5, the MOFOA-HS is performed 10 times independently on the instance 50_10_15_13, which is a moderate-scale instance. $PA_i (i = 1, 2, \ldots, 16)$ represents the obtained Pareto solutions, and the final solutions (FS) are the union of all sets PA_i with non-dominated sorting. Then, the relevance value (RV) of each parameter combination is calculated as $RV(i) = |PA_i \cap FS|/|FS|$. Thus, a larger RV means a better correlation. The relevance ranks of each parameter can be determined in Table 6. Moreover, the changing trend of each factor parameter level for 50_10_15_13 can be depicted in Figure 6.

Table 6. The significant rank of each parameter for instance 50_10_15_13.

Level	*NS*	*S*	*c*
1	0.01356	0.04618	0.07614
2	0.05792	0.05396	0.06813
3	0.09137	0.06785	0.04809
4	0.08592	0.08201	0.05764
Delta	0.07781	0.03584	0.02805
Rank	1	2	3

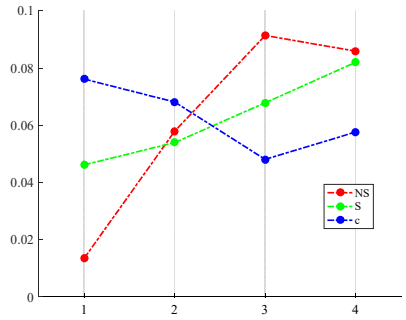


Figure 6. Level trend of the parameters.

From Figure 6, it can be seen that a greater *NS* and a smaller *S* are preferred, which implies that good results need strong exploration and moderate exploitation. As for *c*, a value that is too large may cause premature convergence. According to the above analysis, the recommended parameter combination for the following numerical experiments is $NS = 100$, $S = 15$, and $c = 0.05$.

4.2. Validation of the Mathematical Model

The software CPLEX 12.9.0 is used to verify the correctness of the F-MSRCPSP model. First, the nonlinear variables are linearized in the F-MSRCPSP model. Then, the ε-constraint method is adopted for multi-objective optimization of the transformed linear model. Since it is difficult for the CPLEX solver to reach the ideal solution in a reasonable amount of time in the larger scale case, the small instance 10_3_4_3 is employed to verify the correctness of the mathematical model in this paper. It can be seen that the optimal solution obtained by the F-MSRCPSP is the same as the exact solution solved by the CPLEX solver, as shown in Figure 7, which validates the effectiveness of the proposed model.

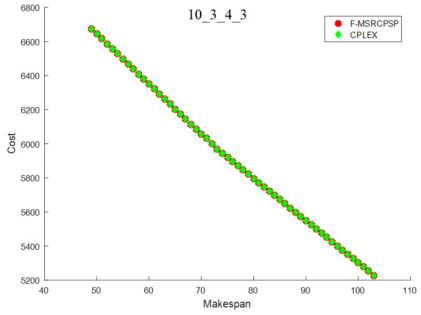
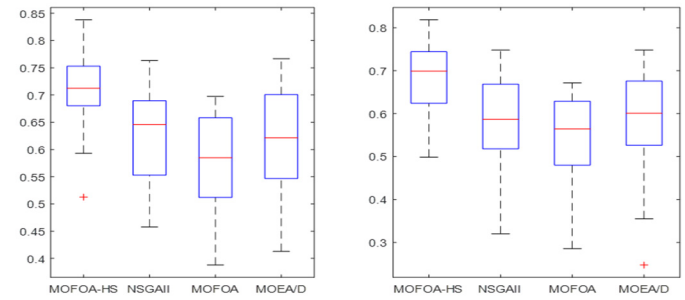


Figure 7. Comparisons of the obtained Pareto fronts.

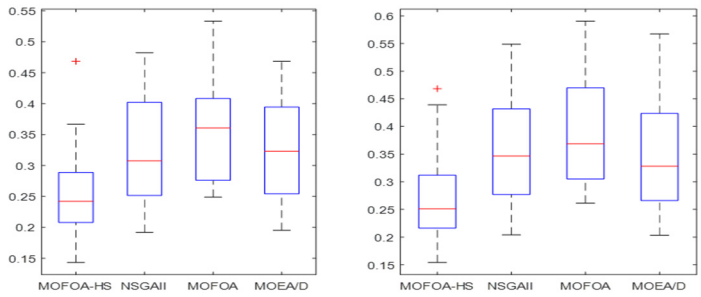
4.3. Comparative Analysis

Next, we compare the MOFOA-HS to other algorithms including the non-dominated sorting genetic algorithm(NSGA-II) [34], the multi-objective fruit fly optimization algorithm (MOFOA) [10], and the multi-objective evolutionary algorithm based on decomposition (MOEA/D) [35]. Two evaluation indicators, Inverted Generational Distance (IGD) [36] and Hypervolume (HV) [37], are used to evaluate the performance of the algorithms. For fair comparisons, the NSGA-II, MOFOA, and MOEA/D also adopt the same solution representation, schedule generation scheme, and initialization mechanism as the MOFOA-HS. Based on the above parameter setting, comparative experiments are carried out by running 10 times independently in all the instances.

The results of the comparative experiments are listed in Table 7, where the best results for each instance are marked in bold. Then, we use HV and IGD evaluation indicators to analyze the above obtained results. The results of the comparative experiment are determined in Tables 8 and 9. Figure 8 illustrates the boxplots of HV and IGD values among the MOFOA-HS, NSGAII, MOFOA, and MOEA/D in all instances.



(**a**) Best and average HV values for comparisons



(**b**) Best and average IGD values for comparisons

Figure 8. Boxplots of HV and IGD values for comparisons.

Table 7. Computational results of four algorithms.

Instance	MOFOA-HS		NSGAII		MOFOA		MOEA/D	
	Make-Span	Cost	Make-Span	Cost	Make-Span	Cost	Make-Span	Cost
10_3_4_3	64.5	5920.0	71.8	6044.1	70.7	6076.4	72.0	6034.3
10_5_7_5	27.1	4348.3	30.1	4379.1	28.5	4480.4	29.0	4420.5
10_7_8_6	25.0	4853.5	28.4	4920.9	28.5	4930.1	28.1	4873.4
15_3_4_3	64.7	7682.4	67.5	7734.6	66.6	7816.5	67.1	7743.5
15_6_10_9	45.9	4718.7	53.2	4722.3	51.0	4763.5	54.1	4755.1
15_9_11_9	35.5	6401.7	39.9	6525.7	39.3	6670.1	39.9	6492.6
25_4_8_9	74.7	22,245.8	78.2	22,392.0	78.1	22,419.4	78.8	22,342.2
25_5_3_15	69.3	24,576.6	72.6	24,778.6	73.6	24,759.6	73.8	24,712.6
25_5_9_9	77.7	10,731.0	82.2	11,040.2	82.6	11,089.2	81.7	11,015.4
50_4_14_9	177.3	36,463.7	199.7	36,949.8	196.6	37,154.3	195.9	37,024.5
50_5_4_14	194.3	60,726.8	203.5	61,233.9	197.8	61,974.0	203.6	61,167.7
50_10_15_13	104.1	29,148.4	116.0	29,229.1	115.9	30,487.8	113.8	29,423.8
78_5_44_15	245.4	55,637.0	260.8	56,947.0	269.8	56,799.5	258.6	56,685.0
78_5_52_9	207.6	13,390.4	226.8	13,483.5	222.6	13,918.6	226.1	13,443.8
78_10_42_15	158.7	57,923.0	178.2	59,373.3	177.0	60,398.7	173.0	59,620.7
100_5_22_14	353.1	73,721.0	366.9	73,872.4	369.1	74,172.3	364.9	73,937.5
100_10_64_9	197.0	36,470.8	247.5	37,616.3	234.9	39,363.7	236.5	38,151.7
100_15_65_9	187.3	47,130.3	216.8	51,331.7	223.6	53,176.0	213.7	51,605.0
200_10_50_9	362.8	112,705.0	391.0	121,011.7	409.4	122,590.3	405.1	119,423.3
200_15_54_15	314.8	120,070.0	339.4	126,526.3	339.3	130,886.0	329.5	128,378.3
200_20_145_15	213.2	104,455.3	242.3	112,647.7	260.1	113,600.7	245.3	112,363.7

Table 8. HV results of four algorithms.

Instance	MOFOA-HS		NSGAII		MOFOA		MOEA/D	
	Best	**Avg**	**Best**	**Avg**	**Best**	**Avg**	**Best**	**Avg**
10_3_4_3	0.5126	0.5125	0.5041	0.4864	0.5009	0.4827	0.5063	0.4968
10_5_7_5	0.7022	0.6921	0.4625	0.4321	0.5447	0.4857	0.6858	0.6563
10_7_8_6	0.6535	0.6252	0.6460	0.6156	0.6466	0.6221	0.6026	0.5818
15_3_4_3	0.5881	0.5774	0.5417	0.5229	0.5161	0.5004	0.5533	0.5389
15_6_10_9	0.8582	0.8450	0.8399	0.8240	0.8546	0.8371	0.8483	0.8219
15_9_11_9	0.7218	0.7153	0.6994	0.6712	0.7066	0.6632	0.7066	0.6722
25_4_8_9	0.4619	0.4222	0.3215	0.2001	0.2095	0.0571	0.3021	0.2259
25_5_3_15	0.6007	0.4918	0.4761	0.3179	0.3294	0.2458	0.4113	0.2303
25_5_9_9	0.7461	0.7320	0.6779	0.6537	0.6463	0.6264	0.6890	0.6713
50_4_14_9	0.6752	0.6620	0.4512	0.3159	0.3764	0.2529	0.4450	0.3065
50_5_4_14	0.7852	0.7779	0.7388	0.7212	0.6887	0.6712	0.7332	0.7186
50_10_15_13	0.8380	0.8182	0.7632	0.7479	0.6549	0.6305	0.7667	0.7481
78_5_44_15	0.7500	0.7419	0.6818	0.6673	0.6222	0.6048	0.6993	0.6848
78_5_52_9	0.7416	0.7218	0.7091	0.6697	0.6685	0.6176	0.7173	0.6783
78_10_42_15	0.7993	0.7832	0.7287	0.7044	0.6588	0.6345	0.7212	0.7039
100_5_22_14	0.6872	0.6645	0.6713	0.5795	0.6331	0.5295	0.6662	0.5769
100_10_64_9	0.7607	0.7286	0.7232	0.6830	0.6856	0.6401	0.7371	0.6931
100_15_65_9	0.8022	0.7928	0.7381	0.7241	0.6811	0.6459	0.7400	0.7238
200_10_50_9	0.6980	0.6777	0.5473	0.5253	0.4743	0.4454	0.5509	0.5320
200_15_54_15	0.7513	0.7498	0.6843	0.6714	0.6160	0.5991	0.6858	0.6725
200_20_145_15	0.7291	0.7153	0.6485	0.6352	0.5848	0.5606	0.6398	0.6288

Table 9. IGD results of four algorithms.

Instance	MOFOA-HS		NSGAII		MOFOA		MOEA/D	
	Best	**Avg**	**Best**	**Avg**	**Best**	**Avg**	**Best**	**Avg**
10_3_4_3	0.4683	0.4683	0.4690	0.4733	0.4735	0.4776	0.4688	0.4714
10_5_7_5	0.2406	0.2497	0.5114	0.5309	0.3664	0.3963	0.2576	0.2728
10_7_8_6	0.2876	0.3088	0.2887	0.3123	0.2889	0.3099	0.3374	0.3451
15_3_4_3	0.3688	0.3776	0.4120	0.4229	0.4363	0.4448	0.4011	0.4116
15_6_10_9	0.1151	0.1247	0.1284	0.1379	0.1163	0.1310	0.1263	0.1416
15_9_11_9	0.2394	0.2429	0.2508	0.2644	0.2419	0.2648	0.2466	0.2635
25_4_8_9	0.3863	0.4226	0.5383	0.6065	0.6213	0.6747	0.5491	0.5961
25_5_3_15	0.3246	0.4248	0.4308	0.5311	0.5242	0.5581	0.4513	0.5511
25_5_9_9	0.2019	0.2120	0.2594	0.2768	0.2697	0.2907	0.2565	0.2647
50_4_14_9	0.2862	0.2982	0.4854	0.5458	0.5382	0.5905	0.4759	0.5370
50_5_4_14	0.1827	0.1858	0.2197	0.2271	0.2502	0.2614	0.2214	0.2291
50_10_15_13	0.1431	0.1539	0.1920	0.2038	0.2724	0.2890	0.1951	0.2033
78_5_44_15	0.2113	0.2171	0.2699	0.2779	0.3168	0.3229	0.2542	0.2628
78_5_52_9	0.2256	0.2387	0.2477	0.2747	0.2795	0.3171	0.2401	0.2685
78_10_42_15	0.1757	0.1857	0.2183	0.2414	0.2958	0.3088	0.2539	0.2569
100_5_22_14	0.2966	0.3089	0.3037	0.3648	0.3313	0.4013	0.3100	0.3661
100_10_64_9	0.2081	0.2297	0.2352	0.2680	0.2672	0.3012	0.2229	0.2571
100_15_65_9	0.1789	0.1851	0.2306	0.2384	0.2773	0.2940	0.2258	0.2356
200_10_50_9	0.2809	0.2911	0.3982	0.4159	0.4691	0.4833	0.3931	0.4077
200_15_54_15	0.2227	0.2232	0.2839	0.2940	0.3348	0.3406	0.2851	0.2946
200_20_145_15	0.2362	0.2449	0.3075	0.3124	0.3604	0.3686	0.3230	0.3280

It can be clearly seen that the computational results of MOFOA-HS are significantly better than those of the other algorithms. Specifically, MOFOA-HS has a larger HV value and a smaller IGD value in both the average and best situation. This indicates that MOFOA-HS has a better comprehensive performance than others to a great extent in all the instances.

What is more, the performance of MOFOA-HS, NSGAII, MOFOA, and MOEA/D are analyzed with a *t*-test according to the above HV and IGD values.

Example 1. *The mean value of HV of MOFOA-HS is more than that of the other algorithms.*

$$H_0 : \theta_{mean} > \theta'_{mean} \tag{25}$$

$$H_1 : \theta_{mean} \leq \theta'_{mean} \tag{26}$$

Example 2. *The mean value of IGD of MOFOA-HS is less than that of the other algorithms.*

$$H_0 : \theta_{mean} < \theta'_{mean} \tag{27}$$

$$H_1 : \theta_{mean} \geq \theta'_{mean} \tag{28}$$

Regarding the above example, H_0 is the null hypothesis, and H_1 is the alternative hypothesis; θ and θ', respectively, represent the evaluation indicator values of MOFOA-HS and the other algorithms. All tests are carried out at the default $\alpha = 0.05$ significance level, and the test results for the examples are shown in Table 10, where "Y" and "N" mean, respectively, that the null hypothesis is accepted, and the null hypothesis fails to be accepted.

Table 10. *t*-test results at a = 0.05 significance level.

Instance		HV			IGD		
		NSGA-II	MOFOA	MOEA/D	NSGA-II	MOFOA	MOEA/D
10_3_4_3	*p*-value	0.31	0.29	0.01	0.75	0.79	1.00
	Null hypothesis	Y	Y	N	Y	Y	Y
10_5_7_5	*p*-value	0.96	1.00	0.95	0.14	0.93	1.00
	Null hypothesis	Y	Y	Y	Y	Y	Y
10_7_8_6	*p*-value	0.48	0.06	0.01	0.66	0.94	1.00
	Null hypothesis	Y	Y	N	Y	Y	Y
15_3_4_3	*p*-value	0.72	0.62	0.78	0.77	0.81	0.61
	Null hypothesis	Y	Y	Y	Y	Y	Y
15_6_10_9	*p*-value	0.98	0.44	1.00	0.53	0.82	0.78
	Null hypothesis	Y	Y	Y	Y	Y	Y
15_9_11_9	*p*-value	0.87	0.70	0.47	0.78	0.91	0.92
	Null hypothesis	Y	Y	Y	Y	Y	Y
25_4_8_9	*p*-value	0.41	0.69	0.67	0.83	0.95	0.85
	Null hypothesis	Y	Y	Y	Y	Y	Y
25_5_3_15	*p*-value	0.93	1.00	0.95	0.91	1.00	0.84
	Null hypothesis	Y	Y	Y	Y	Y	Y
25_5_9_9	*p*-value	0.07	0.60	0.90	0.92	0.80	0.31
	Null hypothesis	Y	Y	Y	Y	Y	Y
50_4_14_9	*p*-value	0.79	0.88	0.61	0.57	1.00	0.86
	Null hypothesis	Y	Y	Y	Y	Y	Y
50_5_4_14	*p*-value	0.97	0.00	0.20	0.20	1.00	0.78
	Null hypothesis	Y	N	Y	Y	Y	Y
50_10_15_13	*p*-value	0.81	0.80	0.99	0.61	1.00	0.44
	Null hypothesis	Y	Y	Y	Y	Y	Y
78_5_44_15	*p*-value	0.29	0.76	0.15	0.89	0.93	0.91
	Null hypothesis	Y	Y	Y	Y	Y	Y
78_5_52_9	*p*-value	1.00	0.27	1.00	0.17	1.00	0.29
	Null hypothesis	Y	Y	Y	Y	Y	Y
78_10_42_15	*p*-value	0.54	0.44	0.13	0.99	1.00	1.00
	Null hypothesis	Y	Y	Y	Y	Y	Y
100_5_22_14	*p*-value	1.00	1.00	1.00	0.07	0.94	0.20
	Null hypothesis	Y	Y	Y	Y	Y	Y
100_10_64_9	*p*-value	0.15	0.15	0.08	0.96	0.97	0.96
	Null hypothesis	Y	Y	Y	Y	Y	Y

Table 10. *Cont.*

Instance		HV			IGD		
		NSGA-II	MOFOA	MOEA/D	NSGA-II	MOFOA	MOEA/D
100_15_65_9	*p*-value	0.67	0.03	0.58	0.74	1.00	0.77
	Null hypothesis	Y	N	Y	Y	Y	Y
200_10_50_9	*p*-value	0.45	0.40	0.88	0.79	0.99	0.83
	Null hypothesis	Y	Y	Y	Y	Y	Y
200_15_54_15	*p*-value	0.11	0.00	0.01	0.96	1.00	0.96
	Null hypothesis	Y	N	N	Y	Y	Y
200_20_145_15	*p*-value	0.87	0.13	0.55	0.08	1.00	0.97
	Null hypothesis	Y	Y	Y	Y	Y	Y

As the *t*-test results show, there are only six tests that fail to accept the null hypothesis for 22, and all tests accept the null hypothesis for 22. In general, based on the above experimental results, the MOFOA-HS algorithm is more effective than NSGAII, MOFOA, and MOEA/D in solving the proposed F-MSRCPSP.

5. Conclusions and Future Works

This study proposes a novel F-MSRCPSP, where the execution of an activity requires a combination of multiple resources. In the F-MSRCPSP, the resource combination may vary over a time period, which results in changes in the duration and cost of the activity. To solve the concerned problem, a hybrid multi-objective fruit fly optimization algorithm with heuristic strategies is proposed. A novel solution representation based on an activity sequence list and a resource pattern list is designed. Two scheduling generation schemes, i.e., the FPSGS and the FSSGS, are introduced. Two heuristic strategies, i.e., the GSS and the NGSS, are adopted for specific resource allocation. Then, an initialization method is designed to balance the uniform distribution of the population. Additionally, a series of operators are designed to accelerate the convergence speed and strengthen the exploration of the algorithm, such as a rejoining operator, a contraposition selection operator, an empirical rearrangement and reselection operator, a local and global balanced operator, and so on. We have compared the MOFOA-HS to other excellent algorithms, i.e., NSGAII, MOFOA, and MOEA/D. The results of the comparative experiment and *t*-test statistical analysis show that the proposed MOFOA-HS algorithm obtains the best performance among the compared algorithms.

In future work, it is important to consider the preemptive F-MSRCPSP. In real-world projects, we often encounter a complex and uncertain environment, which can result in a reduction in available resources and the need to temporarily add external activities. Furthermore, the integration of solutions derived from heuristic algorithms should be explored as an initial solution to accelerate the exact solution of the mathematical model.

Author Contributions: Methodology, X.L. (Xu Luo) and B.D.; Software, X.L. (Xinhao Luo); Validation, J.G.; Investigation, J.G.; Writing—original draft, X.L. (Xu Luo) and X.L. (Xinhao Luo); Supervision, S.G. and B.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Yang, S.; Fu, L. Critical chain and evidence reasoning applied to multi-project resource schedule in automobile R&D process. *Int. J. Proj. Manag.* **2014**, *32*, 166–177.
2. Isah, M.A.; Kim, B.S. Integrating Schedule Risk Analysis with Multi-Skilled Resource Scheduling to Improve Resource-Constrained Project Scheduling Problems. *Appl. Sci.* **2021**, *11*, 650. [CrossRef]
3. Arabnejad, V.; Bubendorfer, K.; Ng, B. Scheduling deadline constrained scientific workflows on dynamically provisioned cloud resources. *Future Gener. Comput. Syst.* **2017**, *75*, 348–364. [CrossRef]
4. Zaman, F.; Elsayed, S.; Sarker, R.; Essam, D.; Coello, C.A.C. An evolutionary approach for resource constrained project scheduling with uncertain changes. *Comput. Oper. Res.* **2021**, *125*, 105104. [CrossRef]
5. Almeida, B.F.; Correia, I.; Saldanha-da-Gama, F. A biased random-key genetic algorithm for the project scheduling problem with flexible resources. *Top* **2018**, *26*, 283–308. [CrossRef]
6. Almeida, B.F.; Correia, I.; Saldanha-da-Gama, F. Priority-based heuristics for the multi-skill resource constrained project scheduling problem. *Expert Syst. Appl.* **2016**, *57*, 91–103. [CrossRef]
7. Chand, S.; Singh, H.; Ray, T. Evolving heuristics for the resource constrained project scheduling problem with dynamic resource disruptions. *Swarm Evol. Comput.* **2019**, *44*, 897–912. [CrossRef]
8. Du, B.; Tan, T.; Guo, J.; Li, Y.; Guo, S. Energy-cost-aware resource-constrained project scheduling for complex product system with activity splitting and recombining. *Expert Syst. Appl.* **2021**, *173*, 114754. [CrossRef]
9. Afshar-Nadjafi, B. Multi-skilling in scheduling problems: A review on models, methods and applications. *Comput. Ind. Eng.* **2021**, *151*, 107004. [CrossRef]
10. Wang, L.; Zheng, X. A knowledge-guided multi-objective fruit fly optimization algorithm for the multi-skill resource constrained project scheduling problem. *Swarm Evol. Comput.* **2018**, *38*, 54–63. [CrossRef]
11. Snauwaert, J.; Vanhoucke, M.A. classification and new benchmark instances for the multi-skilled resource-constrained project scheduling problem. *Eur. J. Oper. Res.* **2023**, *307*, 1–19. [CrossRef]
12. Lin, J.; Zhu, L.; Gao, K. A genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem. *Expert Syst. Appl.* **2020**, *140*, 112915. [CrossRef]
13. Dai, H.; Cheng, W.; Guo, P. An improved tabu search for multi-skill resource-constrained project scheduling problems under step-deterioration. *Arab. J. Sci. Eng.* **2018**, *43*, 3279–3290. [CrossRef]
14. Tian, Y.; Xiong, T.; Liu, Z.; Mei, Y.; Wan, L. Multi-Objective multi-skill resource-constrained project scheduling problem with skill switches: Model and evolutionary approaches. *Comput. Ind. Eng.* **2022**, *167*, 107897. [CrossRef]
15. Hosseinian, A.H.; Baradaran, V. A multi-objective multi-agent optimization algorithm for the multi-skill resource-constrained project scheduling problem with transfer times. *RAIRO-Oper. Res.* **2021**, *55*, 2093–2128. [CrossRef]
16. Najafzad, H.; Davari-Ardakani, H.; Nemati-Lafmejani, R. Multi-skill project scheduling problem under time-of-use electricity tariffs and shift differential payments. *Energy* **2019**, *168*, 619–636. [CrossRef]
17. Ahmadpour, S.; Ghezavati, V. Modeling and solving multi-skilled resource-constrained project scheduling problem with calendars in fuzzy condition. *J. Ind. Eng. Int.* **2019**, *15*, 179–197. [CrossRef]
18. Naber, A.; Kolisch, R. MIP models for resource-constrained project scheduling with flexible resource profiles. *Eur. J. Oper. Res.* **2014**, *239*, 335–348. [CrossRef]
19. Naber, A. Resource-constrained project scheduling with flexible resource profiles in continuous time. *Comput. Oper. Res.* **2017**, *84*, 33–45. [CrossRef]
20. Tritschler, M.; Naber, A.; Kolisch, R. A hybrid metaheuristic for resource-constrained project scheduling with flexible resource profiles. *Eur. J. Oper. Res.* **2017**, *262*, 262–273. [CrossRef]
21. Myszkowski, P.B.; Skowronksi, M.E.; Olech, L.P.; Oślizło, K. Hybrid ant colony optimization in solving multi-skill resource-constrained project scheduling problem. *Soft Comput.* **2015**, *19*, 3599–3619. [CrossRef]
22. Berthaut, F.; Pellerin, R.; Hajji, A.; Perrier, N. A path relinking-based scatter search for the resource-constrained project scheduling problem. *Int. J. Proj. Organ. Manag.* **2018**, *10*, 1–36. [CrossRef]
23. Joy, J.; Rajeev, S.; Narayanan, V. Particle swarm optimization for resource constrained-project scheduling problem with varying resource levels. *Procedia Technol.* **2016**, *25*, 948–954. [CrossRef]
24. Liu, J.; Liu, Y.; Shi, Y.; Li, J. Solving resource-constrained project scheduling problem via genetic algorithm. *J. Comput. Civ. Eng.* **2020**, *34*, 04019055. [CrossRef]
25. Tao, S.; Dong, Z.S. Multi-mode resource-constrained project scheduling problem with alternative project structures. *Comput. Ind. Eng.* **2018**, *125*, 333–347. [CrossRef]
26. VO, T.M.N. Centrifugal pump design: An optimization. *Eurasia Proc. Sci. Technol. Eng. Math.* **2022**, *17*, 136–151.
27. Roslon, J.H.; Kulejewski, J.E. A hybrid approach for solving multi-mode resource-constrained project scheduling problem in construction. *Open Eng.* **2019**, *9*, 7–13. [CrossRef]
28. Tran, V.T.; Le, M.H.; Vo, M.T.; Le, Q.T.; Hoang, V.H.; Tran, N.T.; Nguyen, T.T. Optimization design for die-sinking EDM process parameters employing effective intelligent method. *Cogent Eng.* **2023**, *10*, 2264060. [CrossRef]
29. Nemati-Lafmejani, R.; Davari-Ardakani, H.; Najafzad, H. Multi-mode resource constrained project scheduling and contractor selection: Mathematical formulation and metaheuristic algorithms. *Appl. Soft Comput.* **2019**, *81*, 105533. [CrossRef]

30. Pellerin, R.; Perrier, N.; Berthaut, F. A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. *Eur. J. Oper. Res.* **2020**, *280*, 395–416. [CrossRef]
31. Sun, L.; Shi, W.; Wang, J.; Mao, H.; Tu, J.; Wang, L. Research on Production Scheduling Technology in Knitting Workshop Based on Improved Genetic Algorithm. *Appl. Sci.* **2023**, *13*, 5701. [CrossRef]
32. Maghsoudlou, H.; Afshar-Nadjafi, B.; Niaki, S.T.A. A multi-objective invasive weeds optimization algorithm for solving multi-skill multi-mode resource constrained project scheduling problem. *Comput. Chem. Eng.* **2016**, *88*, 157–169. [CrossRef]
33. Du, B.; Guo, S. Production planning conflict resolution of complex product system in group manufacturing: A novel hybrid approach using ant colony optimization and Shapley value. *Comput. Ind. Eng.* **2016**, *94*, 158–169. [CrossRef]
34. Hou, Y.; Wu, N.Q.; Zhou, M.C.; Li, Z. Pareto-optimization for scheduling of crude oil operations in refinery via genetic algorithm. *IEEE Trans. Syst. Man Cybern. Syst.* **2015**, *47*, 517–530. [CrossRef]
35. Cao, J.; Zhang, J.; Zhao, F.; Chen, Z. A two-stage evolutionary strategy based MOEA/D to multi-objective problems. *Expert Syst. Appl.* **2021**, *185*, 115654. [CrossRef]
36. Sun, Y.; Yen, G.G.; Yi, Z. IGD indicator-based evolutionary algorithm for many-objective optimization problems. *IEEE Trans. Evol. Comput.* **2018**, *23*, 173–187. [CrossRef]
37. Groz, B.; Maniu, S. Hypervolume subset selection with small subsets. *Evol. Comput.* **2019**, *27*, 611–637. [CrossRef] [PubMed]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.