REVISED Manuscript (text UNmarked) Click here to view linked References

Problem-specific multi-objective invasive weed optimization algorithm for reconnaissance mission scheduling problem

Abstract

With the progress of technology, the multi-agent system is successfully applied in many applications. In this paper, we investigate the problem of multi-agent system reconnaissance mission scheduling, which is the core of the reconnaissance decision support system and can be modeled as an extension of Multi-Mode Multi-Skill Resource-Constrained Project Scheduling Problem. Three objectives are considered in this paper: (1) minimizing the reconnaissance mission's makespan, (2) minimizing the total cost of allocating reconnaissance agents, and (3) maximizing the total quality of all reconnaissance tasks. An effective problem-specific multi-objective invasive weed optimization algorithm (PS-MOIWO) is proposed for solving the problem. Firstly, a new chromosome structure guaranteeing the feasibility of solutions and an initialization method are proposed. Secondly, we propose a self-adaptive penalty-based constraint handling technique to describe the fitness of each individual and adopt a novel non-dominated sorting method to rank the population. Thirdly, by using the problem-specific knowledge, a local search procedure is developed and incorporated into the PS-MOIWO framework to enhance the exploitation ability. Based on the Taguchi method, algorithm's suitable parameter combinations are determined. Simulation results based on a set of newly generated reconnaissance instances and the comparisons with some existing algorithms demonstrate the proposed algorithm's effectiveness.

Keywords: invasive weed optimization, reconnaissance, multi-mode, multi-skill, RCPSP

1. Introduction

The widespread usage of robotic systems has brought a lot of convenience to humans. But in real-world applications, there are still many scenarios where robotic systems cannot replace humans. In such cases, manned and unmanned systems need to work together to deal with complex tasks. A system with several manual and unmanned equipment can be modeled as a multi-agent system, for the agents in a multi-agent system could equally well be robots, humans, or human teams. The reconnaissance mission is an example where manned and unmanned systems are involved simultaneously. Therefore, the human resource and unmanned equipment that performs the reconnaissance mission is usually modeled as a multi-agent system. As shown in Fig. 1, a reconnaissance mission consists of several reconnaissance tasks and a network that denotes the precedence relations between tasks, the agents assigned to a reconnaissance task make exploration in that area to gain information about natural features and human activities. The precedence relation means one task can be started to be processed only if all its predecessor tasks are finished. Usually, there are not enough agents to perform all the tasks which can perform according to the precedence relations at the same time. We need to schedule the start time of each task and assign agents to each task, which is the reconnaissance mission scheduling problem investigated in this paper. Reconnaissance

missions can be conducted in military operations (Kim et al., 2017), where the action of enemies and geographical information of the combat area are collected by military reconnaissance multi-agent system. Forest fire management (Laszlo et al., 2018) also contains reconnaissance on fire information, where firefighters need to work with drones to collect fire information. Reconnaissance mission also exists in environmental protection (Chen et al., 2019) and agricultural applications (Bloss, 2014). Choi & Ahn (2020) proposed to use imitation learning to plan UAV path for reconnaissance. John A. Richards (2019) focused on using autonomous multi-sensor platforms to conduct intelligence, surveillance, and reconnaissance mission. Several features of reconnaissance mission are summarized as follows:

- The reconnaissance mission consists of several tasks with precedence relations.
- Geographical locations are different for different tasks.
- Each task requires one or more execution abilities (skills) to be performed. As shown in Fig. 2, a reconnaissance task requires at most three types of skills.
- Each task has a start point and a finish point. The start point specifies the location for reconnaissance agents to enter the task area. The finish point specifies the location for reconnaissance agents to leave the task area.
- Reconnaissance agents can provide the skill required to perform tasks. Considering the proficiency level of human resources and equipment status are different, different agents usually provide both different types and different amounts of skills.
- Each task is specified with the longest processing time. The processing time of a task is a non-linear function of the agent allocation result.
- The makespan of reconnaissance mission, the cost, and the quality of performing a reconnaissance mission are three important indicators to evaluate a reconnaissance mission schedule.

The precedence relations between tasks are of great importance to the successful execution of the reconnaissance mission. As in forest fire management and some military applications, it could be dangerous to enter an area when the path's information is unclear, which may cause damage or loss of the agents. Thus the precedence relations between tasks must be respected during the execution of the reconnaissance mission. Then, a detailed explanation of Fig. 1 is given, the green point denotes the start point, and the red point denotes the finish point. The yellow lines are the paths between different tasks. Some agents are available to conduct the reconnaissance mission. Reconnaissance mission scheduling is to schedule a given set of tasks with a finite set of reconnaissance agents, while the makespan, the cost, and the quality are optimized. Such scheduling process needs to answer the following questions: for a given task, when the task starts, which agent is assigned to perform that task, and what kind of capabilities each agent provides. Reconnaissance mission scheduling is the core of the reconnaissance decision support system, which plays a vital role in the system's performance.

In operation research, Resource-Constrained Project Scheduling Problem (RCPSP) is a research hotspot that has received the attention of many researchers (Zheng et al., 2017). It consists of finding a schedule for the project of activities with precedence constraints under the limited resource availability (Abdolshah, 2014). The most common goal of the basic version of RCPSP is to minimize the duration (makespan) of the project without violating the priority constraints. Many similarities

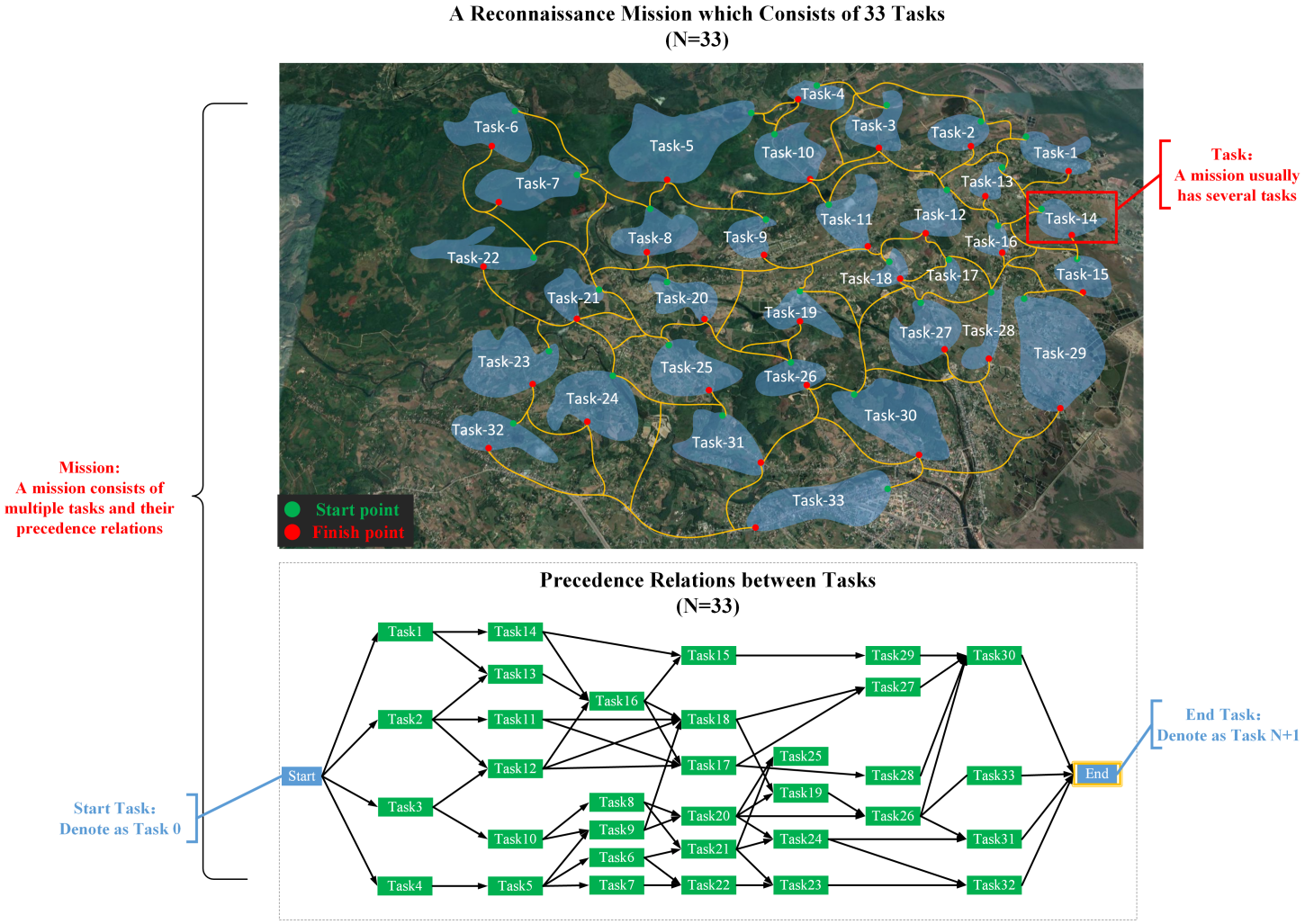


Figure 1: Reconnaissance mission example.

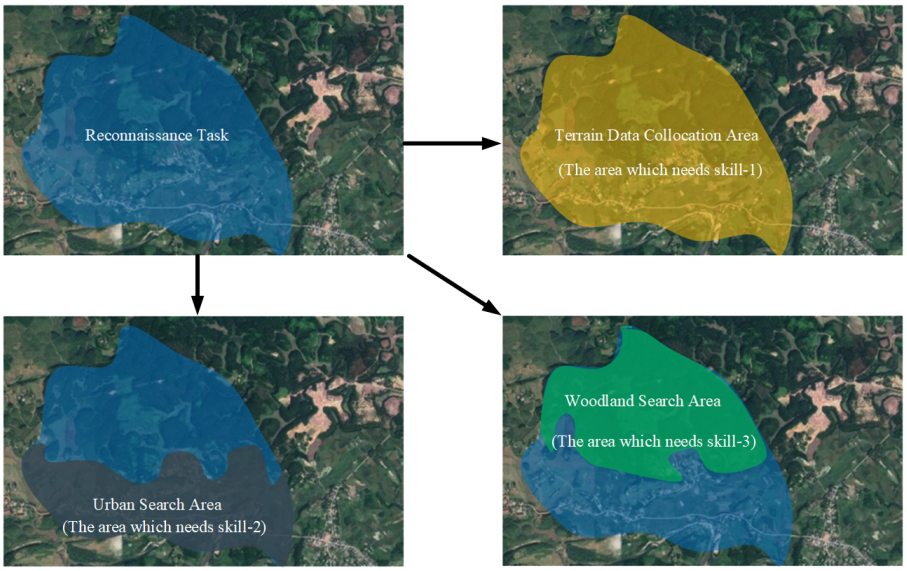


Figure 2: Reconnaissance task division by skill requirement.

exist between the reconnaissance scheduling problem and RCPSP. Both of the two problems consist of several tasks (activities) with precedence relations. The tasks in the reconnaissance mission require some abilities to be performed, the "some ability" corresponds to the phrase of "multi-skill" for RCPSP. The agent which could be used to perform reconnaissance tasks is similar to the definition of "resource" in RCPSP. The multiple possible processing time of the reconnaissance task can be viewed as the "multi-mode" feature of RCPSP. Because of the above similarities between the project scheduling and reconnaissance scheduling, the reconnaissance mission scheduling problem is modeled as an extension of Multi-Mode Multi-Skill Resource-Constrained Project Scheduling Problem (MMMS-RCPSP). In this paper, we present the mathematical formulation of the reconnaissance mission scheduling problem. Because the problem investigated in this paper is NP-hard, to solve the problem, we propose a problem-specific multi-objective invasive weed optimization algorithm (PS-MOIWO). The experiment shows the proposed algorithm outperforms other comparison algorithms.

The contributions of this paper are fourfold: (1) A non-linear integer programming model of reconnaissance mission scheduling problem is established. (2) A problem-specific multi-objective invasive weed optimization algorithm is proposed. (3) A self-adaptive penalty based constraint handling technique is developed. (4) The experiment results prove the effectiveness of the proposed algorithm.

The rest of the paper is organized as follows. Section 2 presents a short summary of existing publications. The mathematical formulation of the reconnaissance mission scheduling problem is presented in Section 3. In Section 4, a problem-specific multi-objective invasive weed optimization algorithm (PS-MOIWO) is proposed. Section 5 shows the experiment result. Section 6 concludes the article and presents our future research plan.

2. Related work

In this paper, the reconnaissance scheduling problem is modeled as an extension of multi-skill multi-mode RCPSP, the agent in the reconnaissance mission has the multi-skill feature. Therefore we can refer to the relevant research on MS-RCPSP. The traditional RCPSP deals with the resource with a single skill. But in many applications, the resource may have more than one type of skill, which is multi-skill RCPSP (MS-RCPSP) (Das & Acharyya, 2011; Skowroński et al., 2013). Néron (2002) firstly defined MS-RCPSP, Bellenguez-Morineau & Néron (2007) proposed a branch and bound algorithm for multi-skill project scheduling problems. In most applications of the project scheduling problem, finding its optimal solution is time-consuming. Therefore designing heuristic or meta-heuristic method attracts the interest of many researchers. Kazemipoor et al. (2002) used a simulated annealing algorithm to solve MS-RCPSP. Chen & Zhang (2012) developed a novel approach with an event-based scheduler and an ant colony optimization algorithm to deal with software project scheduling problem, which is an extension of MS-RCPSP. Using different neighborhood generation methods, two types of Tabu Search for MS-RCPSP have been proposed in Skowroński et al. (2013). Myszkowski developed a hybrid ant colony optimization (HAntCO) approach in Myszkowski et al. (2015), and an application of Greedy Randomized Adaptive Search Procedure (GRASP) in Myszkowski & Siemieński (2016) for MS-RCPSP. Almeida et al. (2016) proposed the concept of resource weight and task group. Based on these concepts, they proposed a new parallel heuristic framework to solve MS-RCPSP. Their algorithm is very efficient in finding feasible solutions. Myszkowski et al. (2017) incorporated task-resources prioritizing method into a co-evolution based approach to solve MS-RCPSP. Myszkowski et al. (2018) used an indirect representation coding method to transform the discrete searching space to continuous space to deal with MS-RCPSP, and a hybrid differential evolution algorithm was proposed in their paper. Hosseinian et al. (2019) proposed a new mixed-integer formulation for the time-dependent MS-RCPSP considering the resource's learning effect.

Multi-mode resource-constrained project scheduling (MM-RCPSP) is another extension of RCPSP, in which each task has several execution modes. Different modes represent different resource requirements and processing time. Reconnaissance mission scheduling problem has the multi-skill feature and has the multi-mode feature, so the related work of MM-RCPSP is presented in this part. Alcaraz et al. (2003) extended the representation and operators previously designed for the single-mode version of the project scheduling problem, and a genetic algorithm was developed to deal with MM-RCPSP. A priority-based heuristic algorithm with polynomial computational complexity was firstly proposed in Hsu & Kim (2005) to make resource allocation in MM-RCPSP. A bi-population genetic algorithm was proposed in Van Peteghem & Vanhoucke (2010), in which the two separate populations provide better parallelism for algorithm implementation and improve the quality of solutions. Cheng et al. (2015) focused on MM-RCPSP with non-preemptive task splitting. A precedence tree-based branch-and-bound algorithm was proposed to find the optimal solution. In most cases, the algorithm takes a lot of time to obtain the optimal solution. A model considering the disruptions in MM-RCPSP was proposed in Chakrabortty et al. (2016). They developed an adaptive metaheuristic procedure based on a neural network to find feasible solutions. Zoraghi et al. (2017) formulated a novel model for the MM-RCPSP with material ordering problem and used hybrid metaheuristic algorithms to obtain a feasible solution. The alternative project structures were taken into consideration in Tao & Dong (2018). A bi-objective linear integer programming model was proposed to trade-off total makespan and cost, a hybrid metaheuristic that combines tabu search and NSGA-II was designed. Chakrabortty et al. (2019) proposed to use a modified version of the variable neighborhood search algorithm to solve the MM-RCPSP. The experiment

showed that their algorithm obtains very high-quality solutions. The waterway ship scheduling problem was modeled as MM-RCPSP in Hill et al. (2019), an integer programming approach was presented to obtain the optimal solution.

Neither MS-RCPSP nor MM-RCPSP can be used to solve the reconnaissance mission scheduling problem independently. Very few papers study multi-mode multi-skill RCPSP. To the best of the author's knowledge, there are only three related papers (Kadrou & Najid, 2006; Santos & Tereso, 2011; Maghsoudlou et al., 2016). Kadrou & Najid (2006) proposed a heuristic algorithm for multi-mode multi-skill RCPSP. The idea of the heuristic is to enumerate some schedulable combinations of tasks with an appropriate resource assignment and schedule from them the one having the best value for an evaluation criterion, the objective is to minimize the makespan of the project. Santos & Tereso (2011) presented a heuristic algorithm for multi-mode multi-skill RCPSP and a software application using that algorithm. In their algorithm, a penalty function was included for tardiness beyond the specified time of allocation to make mode selection. Minimizing the makespan of the project is the only objective. The above two algorithms can only deal with single objective cases, but the reconnaissance mission scheduling problem investigated in this paper has three objectives. Maghsoudlou et al. (2016) proposed a new multi-objective invasive weed optimization algorithm to deal with the multi-mode multi-skill RCPSP. There are some differences between the problem in Maghsoudlou et al. (2016) and reconnaissance mission schedule: (1) Maghsoudlou et al. (2016) assumed the exact information of each mode is known, the same as the other traditional multi-mode RCPSP. But the mode information of the reconnaissance mission needs to be calculated. Suppose we transform the reconnaissance mission scheduling problem to multi-mode RCPSP, because the processing time is a non-linear function of agent allocation result and the inequality of agents' skills. In that case, the mode number of a task is between 3312 to 56231. But in research papers (Chakrabortty et al., 2016; Zoraghi et al., 2017; Tao & Dong, 2018; Chakrabortty et al., 2019) dealing with MM-RCPSP, the numbers of mode for a task are all less than 18. Obviously, the mode number of the problem investigated in this paper is significantly higher than that of traditional MM-RCPSP. The time cost of solving reconnaissance mission scheduling problems with traditional MM-RCPSP method approximates greedy search. (2) Maghsoudlou et al. (2016) assumed the transfer time between different tasks is zero, contrary to the reconnaissance task. (3) Maghsoudlou et al. (2016) assumed the amount of skill mastered by the same type of resource is always the same. Still, in reconnaissance mission, even two agents of the same type usually master the different amount of skills, which is another reason that leads to the number of mode in reconnaissance mission larger than that of traditional MM-RCPSP. (4) In reconnaissance mission scheduling, there is an upper bound of the number of agents assigned to a reconnaissance task, which is not considered in Maghsoudlou et al. (2016). The solution representation in Maghsoudlou et al. (2016) can not be used for reconnaissance mission scheduling. (5) The project's quality is assumed to be time-dependent in Maghsoudlou et al. (2016), but the reconnaissance's quality is time-relevant.

Invasive weed optimization algorithm (IWO) was firstly proposed by Mehrabian & Lucas (2006). The idea behind IWO is inspired by an agricultural phenomenon: the colonization of invasive weeds. The algorithm has the following features: simple structure, fewer parameters, strong robustness, easy to understand, and easy to program. Because IWO has the above advantages, it has been successfully applied to the cooperative multiple task assignment of UAVs (Ghalenoei et al., 2009), constrained optimization of combustion at a coal-fired utility boiler (Zhao et al., 2009), constrained engineering design (Su et al., 2009), permutation flow-shop scheduling problem (Chen et al., 2013) and other applications.

Overall, there is no paper published modeling the reconnaissance mission scheduling problem

considering the following features simultaneously: (1) The mission, which consists of several tasks with precedence relations, the precedence relations between tasks must be respected when performing the mission. (2) Three objectives are considered, the makespan of the mission, cost, and quality of performing the mission. (3) Each reconnaissance agent has at least one type of skill; given a specific type of skill, the skill units masted by different agents are usually different. (4) The processing time of each task is a problem-specific non-linear function of the agent assignment result. (5) The transfer time between different tasks is considered. Apparently, no algorithm was designed specifically for reconnaissance mission scheduling problems. We need to use problem information to design a more effective algorithm.

3. Problem formulation

The objectives of the reconnaissance mission scheduling problem are threefold: (1) minimizing the reconnaissance mission's makespan, (2) minimizing the total cost of allocating reconnaissance agents, and (3) maximizing the total quality of processing reconnaissance tasks. The precedence relationship between tasks is modeled as a task on node network $G = (V, E)$, where V represents a set of tasks, E indicates the precedence relationship between tasks.

3.1. Assumptions

The following assumptions are made for the reconnaissance mission scheduling problem:

- Tasks are numbered topologically with 0 and $N + 1$ as dummy start and dummy end tasks.
- Preemption is not allowed. That is, if a task is being processed, it must be processed to the end.
- All multi-skill resources used in the reconnaissance mission are reconnaissance agents and are modeled as renewable resources corresponding to the definition in project scheduling.
- Each agent can only contribute one skill to a specific task at the same time.
- Each agent performs a reconnaissance task with a pre-defined cost and quality.
- An task can start to be processed if all the agents assigned to it have been transferred to the start point.

3.2. Notations

The notations used in problem formulation are listed in Table 1. N is the number of non-dummy tasks. A reconnaissance mission has two dummy tasks: start task and end task, start task is denoted as task 0, end task is denoted as task $N + 1$.

Some indices are used:

- i, j: index of tasks, $i, j = 0, 1, 2, ..., N, N + 1$.
- l: index of skills of reconnaissance agent, $l = 1, 2, ..., LN$.
- k: index of reconnaissance agent (resource), $k = 1, 2, ..., K$.
- t: index of time step.

Table 1: The symbols.

Symbols	Description
N	the number of non-dummy tasks.
$V = \{0, \ldots, i, \ldots, j, \ldots, N+1\}$	set of tasks, task 0 and $N+1$ are dummy tasks.
$R = \{1, \ldots, k, \ldots K\}$	set of agents (resources)
$L = \{1, \ldots, l, \ldots LN\}$	set of skills corresponding to reconnaissance capacity.
P_j, P_j^I	tasks which are the direct and indirect predecessor of task j respectively.
S_j, S_j^I	tasks which are the direct and indirect predecessor of task j respectively.
L_j	skills required by task j.
L^k	skills mastered by resource k.
V_k	tasks which require at least one skill mastered by resource k.
A_j^l	size of the area in task j which requires skill l.
R_j	agents which can contribute at least one skill required by task j.
RA_j^l	agents allocated to perform skill l in task j.
MR_j	the maximum number of agents assigned to task j.
u_{kl}	the units of skill l that agent k masters.
c_{kl}	cost of performing skill l by agent k per unit time.
q_{kl}	quality of performing skill l by agent k per unit time.
p_j	the processing time of task j.
p_{max}^j	the longest processing time of task j.
Γ_j	the setup time of task j.
Δ_{ij}	transfer time between task i to task j.
UB	the upper bound of the makespan of a reconnaissance mission.
$T = \{0, \ldots, t, \ldots, UB\}$	set of discrete time steps.
$ES_j, LS_j,$	earliest and latest start times of task j.
s_{jt} (decision variable)	is 1 if task j is to be processed at time t, 0 otherwise.
x_{jklt}(decision variable)	is 1 if resource k processes task j with skill l at time t and the value of t equals to the start time of task j, 0 otherwise.
z_{ijk}(decision variable)	is 1 if resource k is transferred for i to j, 0 otherwise.

3.3. Mathematical formulation

We extend the model of Almeida et al. (2019). The mathematical formulation of the reconnaissance mission scheduling problem can be formulated as follows:

$$f_1 = \min \sum_{t=ES_{N+1}}^{LS_{N+1}} ts_{(N+1)t} \ , \tag{1}$$

$$f_2 = \min \sum_{j\in V} \sum_{l\in L} \sum_{k\in R} \sum_{t\in T} c_{kl} \cdot p_j \cdot x_{jklt} \ , \tag{2}$$

$$f_3 = \min \sum_{j\in V} \sum_{l\in L} \sum_{k\in R} \sum_{t\in T} \frac{p_j^{max}}{\cdot p_j \cdot q_{kl} \cdot x_{jklt}} \ , \tag{3}$$

$$\textbf{s.t.} \quad p_j = \max_{l\in L_j} \{\Gamma_j + \frac{A_j^l}{\sum_{k\in R} \sum_{t\in T} u_{kl} \cdot x_{jklt}}\} \ , \qquad j \in V, \tag{4}$$

$$\sum_{l\in L} \sum_{k\in R} \sum_{t\in T} x_{jklt} \leq MR_j \ , \qquad j \in V, \tag{5}$$

$$\sum_{t=ES_j}^{LS_j} s_{jt} = 1 \ , \qquad j \in V, \tag{6}$$

$$\sum_{l\in L} x_{jklt} \leq s_{jt} \ , \qquad j \in V, k \in R, ES_j \leq t \leq LS_j, \tag{7}$$

$$\sum_{t=ES_j}^{LS_j} ts_{jt} - \sum_{t=ES_i}^{LS_i} (t+p_i)s_{it} - (UB+\Delta_{ij})\cdot z_{ijk} \geq -UB\ , \quad i \in V\backslash\{n+1\}, j \in V\backslash P_i^I, k \in R, \tag{8}$$

$$\sum_{j\in V} \sum_{\tau=\max\{ES_j, t-p_j+1\}}^{\min\{LS_j,t\}} \sum_{l\in L} x_{jkl\tau} \leq 1\ , \qquad k \in R, t \in T, \tag{9}$$

$$\sum_{j\in V} z_{ijk} \leq 1\ , \qquad i \in V, k \in R_i \cap R_j, \tag{10}$$

$$\sum_{i\in V\backslash S_j^I} z_{ijk} \geq \sum_{e\in V\backslash P_j^I} z_{jek}\ , \qquad j \in V\backslash\{0\}, k \in R_j, \tag{11}$$

$$\sum_{t=ES_j}^{LS_j} \sum_{k\in R} u_{kl}\cdot x_{jklt} \geq \frac{A_j^l}{p_j^{max} - \Gamma_j}\ , \qquad j \in V\backslash\{0, n+1\}, l \in L, \tag{12}$$

$$\sum_{l\in L} \sum_{t=ES_j}^{LS_j} x_{jklt} = \sum_{i\in V\backslash\{n+1\}} z_{ijk}\ , \qquad j \in V\backslash\{0, n+1\}, k \in R_i \cap R_j, \tag{13}$$

$$s_{jt} \in \{0,1\}\ , \qquad j \in V, t \in T, \tag{14}$$

$$x_{jklt} \in \{0,1\}\ , \qquad j \in V, k \in R, l \in L, t \in T, \tag{15}$$

$$z_{ijk} \in \{0,1\}\ , \qquad i \in V\backslash\{0\}, j\backslash\{S_j^I\}, k \in R_i \cap R_j. \tag{16}$$

Eqs. (1) to (2) are the objective functions. Eq. (1) is to minimize the makespan of the reconnaissance mission, the value of Eq. (1) is the start time of the dummy end task, which equals the makespan of the reconnaissance mission. Eq. (2) aims to minimize the cost of allocating reconnaissance agents to perform all tasks. Eq. (3) is to minimize the reciprocal of the sum of the task's qualities, which is equal to maximize the sum of qualities of all tasks. Constraint (4) defines how to compute the processing time of a reconnaissance task when agent allocation has been made. $\sum_{k\in R}\sum_{t\in T} u_{kl} \times x_{jklt}$ in constraint (4) denotes the total units of skill l assigned to perform task j. Constraint (5) restricts the maximum number of reconnaissance agents assigned to a task. Constraint (6) ensures that each task can only start once. Constraint (7) ensures that each agent can only provide one skill to a task. Constraint (8) considers the transfer time and precedence relation between reconnaissance tasks while making sure that a task can only be processed when all the agents assigned to it have reached the start point. The precedence relation means that one task can start to be processed if all its predecessors have been finished. Constraints (9) and (10) restrict that each agent can not be assigned to a task more than once. Constraint (11) ensures that if an agent is transferred to a task, the agent's current position must be the predecessor or the indirect predecessor of that task, which avoids inefficient assignments that an agent is assigned to transfer from a successor task to a predecessor task. Constraint (12) means that enough agents are assigned to tasks to make sure the processing time is less than the corresponding longest processing time. Constraint (13) shows the relationship between decision variable x and z, which means that if an agent is assigned to perform a task, it must be transferred from another task. Constraints (14) to (16) define the domain of each decision variable. Apparently, RCPSP is a particular case of reconnaissance mission scheduling problem. RCPSP is proven to be NP-hard in Blazewicz et al. (1983). Therefore, the reconnaissance mission scheduling problem is NP-hard.

4. The proposed PS-MOIWO

4.1. Algorithm framework

The colonial behavior of invasive weeds inspires the traditional IWO algorithm. The vitality of the invasive weeds is tenacious, and people are working hard to clear them. But the constant evolution makes them more adaptable. Given this ability of invasive weeds, Mehrabian used an abstract way to simulate their behavior to obtain a powerful optimization algorithm. Based on Mehrabian's work, the multi-objective version of the population-based invasive weed optimization algorithm known as MOIWO is developed by Kundu et al. (2011). In MOIWO, a group of weeds make up the initial population. Their seeds spread in designated pastures and become new weeds. A new weed colony is formed around the parent weeds. Weeds growing in more arable areas will have higher competency and more survival opportunities. Therefore, higher breeding will be carried out in the vicinity of these weeds. With the increase of iterations, the distance between the newly generated weeds and the parent weeds will gradually decrease. In the beginning, a larger distance can diversify the weeds generated in more extensive search space. With the algorithm's processing, the convergence is accelerated by dynamically reducing the distance between parent weeds and child weeds.

The framework of the proposed PS-MOIWO is presented as follows:

Step 1 Generate P_0 individuals using population initialization algorithm proposed in Section 4.3.

Step 2 Evaluate the population using fitness function proposed in Section 4.6, prioritize the population using the sorting method illustrated in Section 4.7.

Step 3 Generating a new colony of weeds. Let some individuals in the population generate some seeds. The individuals with better adaptability will make more seeds than others. The number of seeds will be calculated using the following equation:

$$seeds_i = \text{ floor } \left(S_{\min} + (S_{\max} - S_{\min}) \cdot \left(\frac{np - \text{rank}_i}{np} \right) \right) \tag{17}$$

where seeds $_i$ represents the number of seeds generated by individual i. rank$_i$ is the position of individual i in the population after sorting. S_{min} and S_{max} denote minimum and maximum numbers of seeds generated by each weed.

Step 4 The generated seeds are scattered in the search space using a number sampled by a standard gaussian distribution with a mean value of zero and a standard deviation which can be calculated as:

$$\sigma_{\text{current}} = (NOG - NOG_{\text{current}})^n \cdot \frac{\sigma_{\text{initial}} - \sigma_{\text{final}}}{NOG^n} + \sigma_{\text{final}} \tag{18}$$

where n is a coefficient of non-linear regulation, NOG is the maximum number of generations, and NOG_{current} is the current generation number.

Step 5 Conduct the local search method proposed in Section 4.4 on each seed.

Step 6 If the number of individuals in a population becomes larger than the maximum size of population (P_{max}), rank the population and preserve the best P_{max} number of individuals.

Step 7 Repeat the process until the stopping criterion is met (maximum number of generations, NOG).

4.2. Solution representation

In this paper, each chromosome is represented as a task vector with a resource matrix. The first part of a chromosome is a random-key based task vector ($\boldsymbol{\pi} = \{\pi_1, \pi_2, \ldots, \pi_N\}$). Each task vector element takes a random value from 0 to 1, representing the priority of the corresponding task. To decode the task vector to a feasible task list, N elements of the task vector are separated and sorted in ascending order. The task list sequence is obtained for activities: each place of the task list is placed with the highest-ranking task while satisfying the precedence constraints. The second part of a chromosome is a resource matrix, which specifies the agents assigned to tasks and each agent's skill to perform. The matrix is denoted as $\boldsymbol{M}$. It has K rows and $\sum_{j \in V} |L_j|$ columns, K is the number of agents, $|L_j|$ represents the number of skills that task j requires to be performed. The row of resource $\boldsymbol{M}$ represents each agent. The column represents the skill requirement of the task. The element in $\boldsymbol{M}$ takes value from 0 to 1. If task-i is in the ip-th position of task list $\boldsymbol{\pi}$ and requires L_j kinds of skills, the $\sum_{j \in \{\pi_1 \ldots \pi_{ip-1}\}} |L_j| + 1$ column in matrix $\boldsymbol{M}$ represents the agents assigned to task-i to perform the first type of skill required by task-i, the $\sum_{j \in \{\pi_1 \ldots \pi_{ip-1}\}} |L_j| + e$ column in matrix $\boldsymbol{M}$ represents the agents assigned to task-i to perform the last type of skill required by task-i. Moreover, the decoding process of an activity vector and resource matrix guarantees the obtained solution's feasibility. For all columns that represent the skill requirements of a specific task, given a row that corresponds to an agent, if the largest value of the elements determined by a certain row and these columns are larger than 0.5 and the agent processed the corresponding skill, then the agent represented by this row is assigned to perform that skill to the task defined by the column of this largest element, otherwise do the same thing to the second-largest value until all the skills that the agent masters are visited. A small-size reconnaissance mission with 5 tasks and 3 agents is presented to illustrate the solution representation and decoding process. The precedence network is shown as Fig. 3. The skill requirement of the mission is shown in Table 2. Note that the skill requirement is the minimum skill requirement that must be satisfied. The skill information of each agent is presented in Table 3. The solution structure is shown in Fig. 4. Fig. 5 illustrates the decoding process for the task vector to get a task list. Based on the task list, the column definition of a resource matrix is shown in Fig. 6. Fig. 7 shows the decoding process for task-2 to get the resource allocation result.

Table 2: The minimum skill requirement of small-size reconnaissance mission example.

Task	max agents	skill-1	skill-2	skill-3
1	2	10	0	0
2	2	5	0	6
3	2	0	4	0
4	2	10	0	8
5	2	5	0	0

Table 3: The skill information of agents in example.

Agent	skill-1	skill-2	skill-3
1	5	0	2
2	5	0	2
3	0	5	8

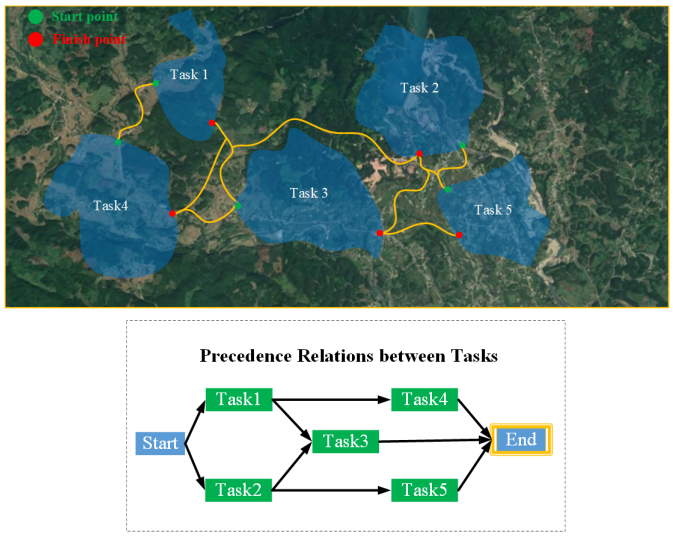


Figure 3: Precedence relation of a small example.

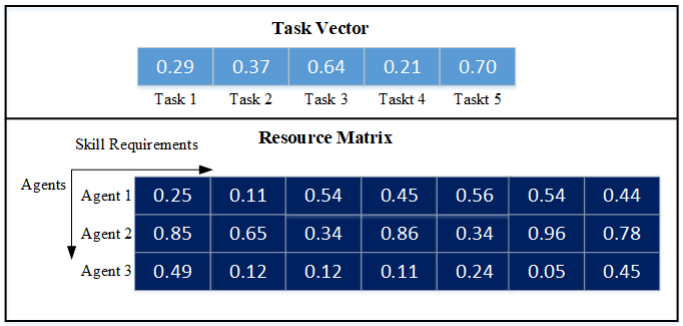


Figure 4: Example of solution representation.

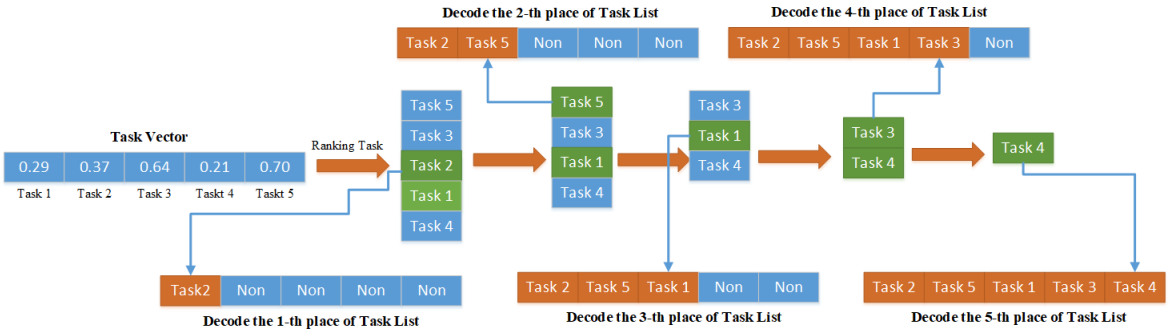


Figure 5: Decoding activity vector to activity list.

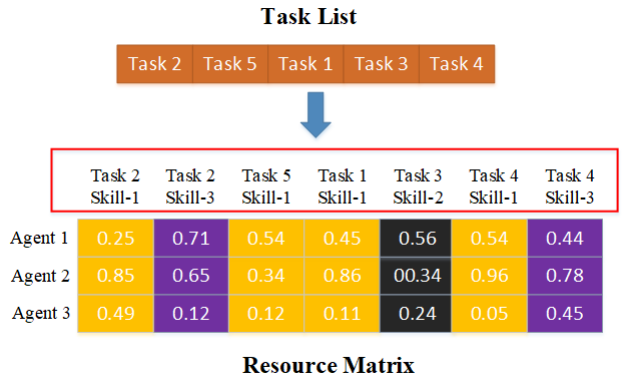


Figure 6: Column definition of resource matrix.

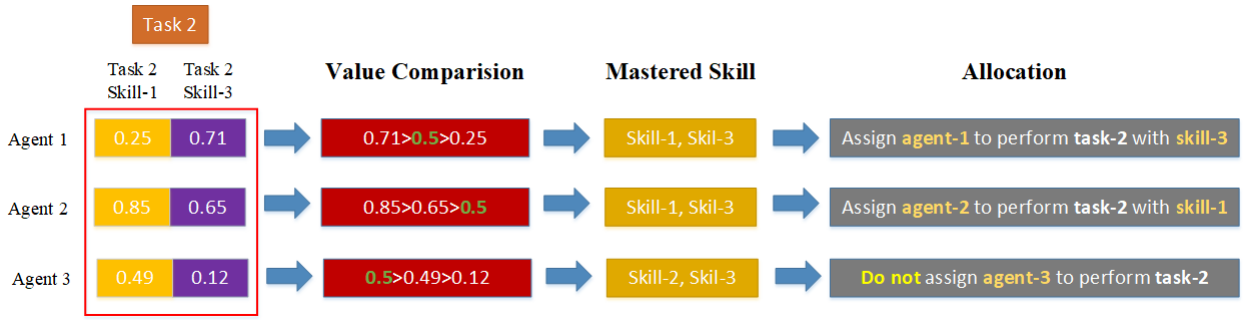


Figure 7: Decoding resource matrix to resource allocation result.

4.3. Population initialization

Reconnaissance mission scheduling is a very complex optimization problem that involves several non-linear constraints. This problem contains some non-smooth objective functions and constraints, which means gradient information is not available. The nonlinearity and multimodality make reconnaissance mission scheduling difficult to solve. According to Yang et al. (2018); Heidari et al. (2017), both empirical observations and numerical statistics suggest that a good quality initial population tends to get better solutions to the optimization problem. Motivated by the above reasons, a heuristic initialization algorithm is proposed to generate the initialization population with good quality and diversity; a serial scheduling generation scheme is adopted. The framework of the population initialization algorithm is presented in Algorithm 1. When an agent is assigned to a task, the corresponding values in the resource matrix are generated using Algorithm 2.

Unlike the traditional multi-mode multi-skill RCPSP, in reconnaissance mission scheduling problem, the original reconnaissance mission information does not contain a specific skill required for each task. For each task, only the area of each type of reconnaissance, a maximum processing time, and the maximum number of agents assigned to the task are given. The minimum skill requirement means that if the allocated skill is less than the minimum skill requirement, a task's processing time should be more than its longest processing time, which is unacceptable. We need to know

the exact skill requirement information when using a serial scheduling scheme to avoid allocating extra resources. Therefore, four different modes are designed to calculate the skill requirement for each task in population initialization. To generate an individual in the initial population, a specific mode should be selected from the four different modes:

Mode 1: Min mode, the skill requirement for each task is specified as the minimum skill requirement.

Mode 2: Max mode, the skill requirement for each task is calculated as Eq. (19), where the maximum number of agents that can be assigned to the task and average skill level of reconnaissance are used.

$$\text{skill requirement} = \frac{MR_j}{|L|} \cdot \frac{\sum_{k \in R} u_{kl}}{|R|}, \quad l \in L_j. \tag{19}$$

Mode 3: Middle-level mode, the skill requirement for each task is calculated as the average of Min mode and Max mode.

Mode 4: Random mode, the population initialization method is not used; the resource matrix is generated randomly.

A resource rule should be selected for the individual when a task is chosen to be allocated with agents. The resource rule is used to give a priority value to each available agent. Rank the agents according to their priority value in descending order. Higher ranked agents have a higher priority to be assigned to the given task. Three resource rules are introduced in this paper:

Resource rule 1: Rank the available agents according to the transfer time to each agent's given task.

Resource rule 2: Priority value is defined as the sum of skills of an agent which the given task requires. Assuming the task j is given, the priority value of available agent k can be calculated as:

$$\text{priority value} = \sum_{l \in L_j} u_{kl} \tag{20}$$

Resource rule 3: The priority value of each available agent is randomly generated in the interval $[0, 1]$.

If an agent k is assigned to perform task j, then a skill rule should be specified. Skill rule is used to determine which type of skill agent k should perform. The skill rule gives a priority value to each type of skill mastered by the agent. The agent should perform the type of skill with a higher priority value. Two skill rules are involved in this paper.

Skill rule 1: The priority value is based on the relative scarcity of the skill units mastered by agent and the task's requirement. Assuming the agent k is assigned to perform task j, the skill priority value of agent k can be calculated as:

$$\text{priority value} = \frac{u_{kl}}{\text{skill } l \text{ requirement of task } j}, \quad l \in L_j. \tag{21}$$

Skill rule 2: The skill priority value of each available agent is randomly generated from interval $[0, 1]$.

4.4. Local search procedure

In each iteration, as presented in Step-5 of PS-MOIWO in Section 4.1, after the generated seeds are scattered in the search space, a local search procedure is performed to improve the quality

Algorithm 1: Population Initialization

```
i = 0 (The number of generated individuals)
while i < population size do
    i = i + 1
    D = ∅ (Store the task which has been allocated with agents)
    Select a task requirement mode randomly, calculate the task requirement.
    if The task requirement mode is not random mode then
        Randomly generate N numbers from [0, 1] to fill the task vector, translate the task vector
          to task list π.
        t = 0
        for λ = 1 to N do
            Let j be the activity in position λ of task list π
            while j ∉ D do
                Update the available agent set AA.
                Randomly select a resource rule and a skill rule.
                if The agent in AA can provide enough skills to cover the requirement of task j
                  then
                    Assign the agents to perform task j using the selected resource rule and skill
                      rule. Using Algorithm 2 to generate the value of the corresponding element in
                      resource matrix.
                else
                    Jump out of the loop, set t to the most recent time step when at least one
                      agent is released.
                end
            end
        end
    else
        Randomly generate N numbers from (0, 1) to fill the task vector, translate the task vector
          to task list π.
        All the elements in resource matrix are randomly generated from (0,1)
    end
end
```

Algorithm 2: Value Generation

input : Agent k is assigned to task j to perform skill l
output: The corresponding value in resource matrix

```
for l* ∈ L_j do
    if l* == l then
        The corresponding value in resource matrix is randomly generated using a Gaussian
          Distribution N(0.75, 0.05^2). If the generated value is not in interval (0.50,1), re-sample
          again.
    else
        The corresponding value in resource matrix is randomly generated using a Gaussian
          Distribution N(0.25, 0.05^2). If the generated value is not in the interval (0,0.49),
          re-sample again.
    end
end
```

of the population. The driving motivation to design a local search procedure is to eliminate the unreasonable allocation represented by the existing seeds to improve their quality. Because of the randomness in PS-MOIWO, there may be two kinds of unreasonable allocations in each seed:

Inside Skill Adjustment: Assuming two agents k_1, k_2 are assigned to the same task to perform skill s_1 and s_2 respectively. If agent k_1 masters skill s_2 and agent k_2 masters skill s_1, and the skill level satisfies $u_{k_1 s_1} < u_{k_2 s_1}, u_{k_1 s_2} > u_{k_2 s_2}$, then adjust agent k_1 to perform skill s_2 and agent k_2 to perform skill s_1 to that task.

Left Neighborhood Adjustment: Assuming agent k_1 is assigned to perform skill s_1 to task t_1, agent k_2 is assigned to perform skill s_2 to task t_2. In the task list, task t_1 is next to task t_2 and t_1 is to the left position of t_2. If the transfer time for k_1 to t_1 is longer than that to t_2, the transfer time for k_2 to t_1 is longer than that to t_2, and the the skill level satisfies $u_{k_1 s_1} < u_{k_2 s_1}, u_{k_1 s_2} > u_{k_2 s_2}$, and t_1 is not direct of indirect predecessor of t_2. Then, adjust agent k_1 to perform skill s_2 to task t_2 and agent k_2 to perform skill s_1 to task t_1.

The implementation details of the local search are given in Algorithm 3.

Algorithm 3: Local Search

```
input : Seed s before local search
output: Seed s after local search
for t ∈ Task List of Seed s do
    Find the agents set A that can meet the criteria of Inside Skill Adjustment.
    Conduct Inside skill adjustment for agents in A.
    if t is not in the first place of the Task List then
        Find the agents set B that can meet the criteria of Left Neighborhood Adjustment.
        Conduct Left Neighborhood Adjustment for agents in B.
    end
end
```

4.5. Constraint handling

Using the solution representation and decoding process in this paper, only 2 types of constraint violations may occur. One is that the allocated skill for a task is less than the task's minimum skill requirement; this type of violation is denoted as C_s. The other is that the number of agents allocated to a task is more than the maximum number of agents for that task; this type of violation is denoted as C_n. RA_j^l denotes the set of agents allocated to perform skill l in task j. SR_{jl}^{min} can be calculate using Eq. (22), which represents the minimum requirement of skill l that must be satisfied for task j. The constraint violation for an individual $\vec{e}$ can be calculated as follows:

$$SR_{jl}^{min} = \frac{A_j^l}{p_j^{max} - \Gamma_j} \tag{22}$$

$$C_s(\vec{e}) = \sum_{j \in V} \sum_{l \in L} [\max\{0, (SR_{jl}^{min} - \sum_{k \in RA_j^l} u_{kl})\}] \tag{23}$$

$$C_n(\vec{e}) = \sum_{j \in V} \sum_{l \in L} [\max\{0, |RA_j^l| - MR_j\}] \tag{24}$$

4.6. Fitness function

A constraint handing technique which named "self-adaptive penalty" is adopted in this paper, in which the penalty term is added to the objective function. The fitness function for each objective contains two parts: a distance value and a penalty function. To define the fitness function, firstly, each objective function of each individual $\vec{e}$ is normalized:

$$\tilde{f}_i(\vec{e}) = \frac{f_i(\vec{e}) - f_i^{\min}}{f_i^{\max} - f_i^{\min}}, \quad i \in \{1, 2, 3\} \tag{25}$$

where f_i^{min} is the minimum value of ith objective function, f_i^{max} is the maximum value of ith objective function. The measure of C_s and C_n is the "skill units of the agent" and the "number of agent", respectively. We need to transfer the measure of C_s and C_n to the same. The average skill units of the agent (denote as au) is used.

$$au = \frac{\sum_{k \in R} \sum_{l \in L} u_{kl}}{|R|} \tag{26}$$

For each individual $\vec{e}$ in population, the degree of constraint violation can be normalized as:

$$C(\vec{e}) = \frac{1}{2} \cdot \frac{C_s(\vec{e}) + au \cdot C_n(\vec{e})}{\max\{C_s(\vec{e}), au \cdot C_n(\vec{e})\}} \tag{27}$$

Using r_f denotes the proportion of feasible individuals in the population:

$$r_f = \frac{\text{number of feasible individual}}{\text{number of the population}} \tag{28}$$

The distance value for ith ($i \in \{1, 2, 3\}$) objective function of individual $\vec{e}$ is defined as:

$$d_i(\vec{e}) = \begin{cases} C(\vec{e}), & \text{if } r_f = 0 \\ \sqrt{\tilde{f}_i(\vec{e})^2 + C(\vec{e})^2}, & \text{if } r_f \neq 0 \end{cases} \tag{29}$$

The ith ($i \in \{1, 2, 3\}$) penalty value pn_i of individual $\vec{e}$ can be calculated by:

$$pn_i(\vec{e}) = (1 - r_f) X_i(\vec{e}) + r_f Y_i(\vec{e}) \tag{30}$$

where

$$X_i(\vec{e}) = \begin{cases} 0, & \text{if } r_f = 0 \\ C(\vec{e}), & \text{if } r_f \neq 0 \end{cases} \tag{31}$$

$$Y_i(\vec{e}) = \begin{cases} 0, & \text{if } \vec{e} \text{ is feasible} \\ \tilde{f}_i(\vec{e}), & \text{if } \vec{e} \text{ is infeasible} \end{cases} \tag{32}$$

The fitness of individual $\vec{e}$ for ith ($i \in \{1, 2, 3\}$) objective in reconnaissance mission scheduling is the sum of $d_i(\vec{e})$ and $pn_i(\vec{e})$.

$$F_i(\vec{e}) = d_i(\vec{e}) + pn_i(\vec{e}), i \in \{1, 2, 3\} \tag{33}$$

For the reconnaissance mission scheduling problem, the population is sorted based on 3 fitness functions F_1, F_2, F_3 by using the sorting method proposed in the next section.

4.7. Non-dominated solutions maintaining

The non-dominated sorting method and crowding distance are used to rank the individual in the proposed PS-MOIWO. Non-dominated sorting is mainly used to sort the solutions in population according to the Pareto dominance principle, which plays a critical role in the selection operation of many multi-objective evolutionary algorithms. A modification of traditional non-dominated sorting, which names "Efficient Non-Dominated Sort using Non-Dominated Tree (ENS-NDT)" and proposed in Gustavsson & Syberfeldt (2018), is adopted in PS-MOIWO to accelerate the process of non-dominated sorting. In ENS-NDT, a novel Non-Dominated Tree structure is modified for quicker non-domination checks. Compared with the traditional non-dominated sorting method, ENS-NDT is more efficient.

For individuals in the same front, to improve the sorted population's diversity, the improved crowding distance proposed in Chu & Yu (2018) is incorporated with the PS-MOIWO to prioritize these individuals. Using the non-dominated sorting method and crowding distance introduced in this part, we can prioritize all the individuals in a population.

4.8. Complexity analysis

In this subsection, to make it easier for readers, we redefine the notations and make them consistent with other papers. Using N to denote the max population size (P_{max}) and M denotes the number of objectives.

Space Complexity: During the operation of PS-MOIWO, it uses one population $O(N)$ to store the current solutions and another population to store non-dominated solutions $O(N)$ in the worst case of each iteration. The size of the population is determined when the first iteration is finished. It will not change during the processing of the algorithm.

Time Complexity: After the initialization, the process of generating a new colony of weeds needs $O(N)$ time. Scattering the seeds in the search space and the local search need $O(N)$ time. In the evaluation of the population, the major computational cost is the non-dominated sorting. According to Gustavsson & Syberfeldt (2018), the non-dominated sorting method used in this paper costs $O(MN^2)$ time. So the time complexity of PS-MOIWO is $O(MN^2)$.

5. Computational experiment

In this section, the computational experiment is conducted to assess the performance of the proposed algorithm. All algorithms mentioned in this section are coded in the C++ programming language. All experiments are conducted on a machine running an Intel Core i5-7400 3.4GHz CPU with 16GB of RAM. We firstly give an introduction to the comparison algorithms and the performance metrics. Secondly, we present some parameters used to describe the test instance and the way to generate the test instance. Then the parameters of each algorithm are determined by using the Taguchi method. We show and analyze the result of the computational experiment at last.

5.1. Comparison algorithm

To the best of the author's knowledge, there is no benchmark or specially designed algorithm for reconnaissance mission scheduling problems in the state of the art literature. Therefore, four well-known and effective multi-objective optimization algorithms are used to compare with the proposed PS-MOIWO.

- NSGA-II(IM) (Blank et al., 2017), which presents an improved version of NSGA-II with several mechanisms and operators. We test these mechanisms and operators with the reconnaissance mission scheduling problem. Firstly, the same TSP solver used in (Blank et al., 2017) is adopted to generate the task list in the initial population for our problem. Then, we view each task's requirement as the capacity of a knapsack and each reconnaissance agent as the goods, select a KNP operator to make resource allocation. Thirdly, the local search proposed in Blank et al. (2017) is conducted for each individual in NSGA-II(IM), and the tournament selection is also used. We combine the operators discussed in Blank et al. (2017) and test their performance on our problem. The best combination is 2OPT-EDGE and PACK-RANDOM. The best operator combination is adopted with NSGA-II(IM) in our experiment.

- NTGA (Laszczyk & Myszkowski, 2019), which focuses on solving multi-objective multi-skill RCPSP. In NTGA, a modified selection operator is presented in combination with classical NSGA-II, and a clone prevention method is used to increase the spread of resulting sets.

- MOPSO (Coello & Lechuga, 2002) is another well-known multi-objective optimization algorithm in which solutions are generated randomly, and then the position of the solution is updated to improve its quality. Non-dominated solutions will be stored in a limited archive called repository. The newly generated solutions will be added to this repository. Then, the dominated solutions will be removed from the repository.

- MOEA/D (Zhang & Li, 2007), which provides an excellent algorithmic framework of evolutionary multi-objective optimization. MOEA/D decomposes the target MOP into several scalar optimization problems and then applies the EA to optimize these subproblems simultaneously. This algorithm has been successfully applied in many engineering applications.

5.2. Performance metric

Because the "true Pareto front" is hard to obtain in multi-objective optimization, we use the data obtained by all algorithms to approximate the "true Pareto front." The Pareto front obtained by an algorithm is denoted as "approximate Pareto front". To analyze the performance of the proposed algorithm and comparison algorithms, we use the following popular performance metrics.

Set Coverage (C-metric): This performance metric is proposed in Zitzler & Thiele (1998). Given to 2 pareto fronts P and Q, set coverage (C-metric) $C(P,Q)$ is defined by the percentage of the solutions in P which dominated by at least one solution in Q. Using $|P|$ to represent the size of pareto front P, the definition of $C(P,Q)$ is shown in Eq. (34)

$$C(P,Q) = \frac{|\{q \in Q | \exists p \in P : p \text{ dominates } q\}|}{|Q|} \tag{34}$$

In this paper, we consider the true Pareto front P^* obtained by all the algorithms and an approximate Pareto front P obtained by one algorithm, using $C(P^*,P)$ to represent the set coverage of the corresponding algorithm. Smaller $C(P^*,P)$ is better.

Spacing (SP): The value of SP measures the distribution of the non-dominated solutions along with the approximate Pareto front obtained by an algorithm. SP can be calculated as follows (Schott, 1995):

$$SP = \sqrt{\frac{1}{(n'-1)\bar{d}} \sum_{i=1}^{n'} \left(d_i - \bar{d}\right)^2} \tag{35}$$

where n' is the number of non-dominated solutions obtained by the algorithms, d_i denotes the Euclidean distance between 2 nearest non-dominated solutions of each front, and $\bar{d}$ is the average of that distance. The algorithm with larger SP is better.

Inverted Generational Distance (IGD): IGD is a measure of both the convergence and diversity of the solutions. It represents the average distance of the given set of non-dominated solutions to the true Pareto front. IGD can be calculated as follows (Zhang & Li, 2007):

$$\text{IGD} = \frac{\sum_{i \in P} D\left(i, P^*\right)}{|P^*|} \tag{36}$$

where $D\left(i, P^*\right)$ represents the minimum Euclidean distance between point i (which belongs to the approximate Pareto front P) and the points in true Pareto front P^*. The algorithm with smaller IGD is better.

Hypervolume (HV)(Zitzler & Thiele, 1998): Another name of hypervolume is known as S-metric. It is a metric that measures both closeness and diversity (Audet et al., 2018). The hypervolume is defined as the volume of the space in the objective space dominated by the approximate Pareto front P and delimited from above by a reference point $r \in \mathbb{R}^n$, the way to calculate hypervolume is shown below :

$$HV(P, r) = \lambda_m(\bigcup_{x \in P} [x; r]) \tag{37}$$

where x_i is an individual in approximate Pareto front P, λ_m is the m-dimensional Lebesgue measure. In this paper, the reconnaissance scheduling problem has 3 objectives, so we set $m = 3$. The reference point adopted here is $(\max f_1, \max f_2, \max f_3)$. The Pareto front P with higher value of HV is better.

5.3. Test instance

As the reconnaissance mission scheduling problem is modeled as multi-mode multi-skill RCPSP, some common parameters usually used in traditional project scheduling problems are introduced to describe the difference between test instances: (1) $maxSkill$: the maximum types of skill a resource can master. (2) $nStart$: the maximum number of tasks that are the direct successor of the start task. (3) $nFinish$: the maximum number of tasks that are the direct predecessor of the end task. (4) $MaxPred$: maximum number of predecessors for each task. (5) $MaxSucc$: maximum number of successors for each task. (6) NC: the network complexity, which is defined by the average number of successors of each task; (7) $nAct$: the number of tasks in a project. (8) K: The number of agents (resource). (9) $|L|$: The number of skills.

Some new parameters are proposed to describe the feature of multi-skill agents. MSU: The maximum skill unit of each skill l that an agent can master. TS_l: The total skill supply of skill l.

Some parameters are used in other papers, but they are modified to fit the reconnaissance mission scheduling problem. Skill factor (SF) is used to describe the relationship between the number of skill types required to perform a task $|L_j|$ and total number of skill types $|L|$, $|L_j| = \lceil |L| \times SF \rceil$. By introducing parameter Resource Strength $RSS_l = \frac{TS_l}{\sum_{j \in V} SR_{jl}^{min}}$ for each skill $l \in L$, we can control the total skill requirement of the reconnaissance mission.

As there is no benchmark for reconnaissance mission scheduling problem, based on the statistic data of reconnaissance mission in the real world, we make some extensions of the instance generator

proposed by Almeida et al. (2015) to generate the test instances. Firstly, use the resource generation algorithm in Almeida et al. (2015) to generate reconnaissance resources. Since the resource in the reconnaissance mission is the agent, to fit the character of the reconnaissance agent, the skill level of each agent is re-sampled by a Gaussian Distribution. In each skill re-sample process, the mean equals the value generated by the resource generation algorithm in Almeida et al. (2015), and the standard deviation is set to 0.9. Then by executing "activity generation" algorithm in Almeida et al. (2015), the precedence network $G = (V, E)$, the longest processing time p_j^{max} for each task and the minimum skill requirement SR_{jl}^{min} are obtained. According to statistics, the reconnaissance task's setup time is usually between 0.12 to 0.18 times the total execution time. So the setup time of a task is generate by $\beta \times p_j^{max}$, β is randomly selected from $[0.1, 0.2]$. The task area A_j^l can be calculated using Eq. (22). An interval defined by $[NCR, MCR]$ is introduced to specify the maximum number of resources working in task area A_j^l, α is randomly selected from $[NCR, MCR]$, $MR_j = \frac{A_j^l}{max_{k \in R} u_{kl}} \times \alpha$. Ten test instances are generated to test the algorithms' performance. The value of $nStart$, $nFinish$, $maxSucc$, $maxPred$ for instance 1-5 are set to 4, and for instance 6-10 are set to 6. The detailed information about other parameters for the instances is shown in Table 4. To verify the performance of the algorithm comprehensively, the complexity of instance 1 to 10 increases gradually.

Table 4: Instance Parameters.

Name	$nAct$	K	L	NC	SF	RSS	TS_l	NCR	MCR
instance-1	20	20	3	2.0	0.75	[0.4, 0.5, 0.5]	[1500, 2000, 1600]	0.2	0.5
instance-2	25	20	3	2.0	0.75	[0.5, 0.4, 0.4]	[1600, 2100, 1500]	0.2	0.5
instance-3	30	25	3	2.0	0.75	[0.4, 0.4, 0.4]	[1850, 2500, 1900]	0.2	0.5
instance-4	35	30	3	2.5	0.75	[0.3, 0.4, 0.4]	[2250, 3100, 2400]	0.2	0.5
instance-5	40	30	3	2.5	0.75	[0.3, 0.3, 0.4]	[2300, 2900, 2500]	0.2	0.5
instance-6	45	35	3	2.5	0.75	[0.3, 0.4, 0.3]	[2650, 3600, 2800]	0.25	0.55
instance-7	55	40	3	5.0	0.75	[0.3, 0.3, 0.3]	[2900, 3600, 3200]	0.25	0.55
instance-8	60	40	3	3.0	0.75	[0.4, 0.4, 0.4]	[3000, 4100, 3100]	0.25	0.55
instance-9	65	45	3	3.5	0.75	[0.3, 0.4, 0.4]	[3250, 4050, 3300]	0.25	0.55
instance-10	70	45	3	3.5	0.75	[0.3, 0.3, 0.4]	[3350, 4600, 3200]	0.25	0.55

5.4. Parameters tuning

In order to properly set the parameters, Taguchi method (Taguchi, 1986) is applied in this paper. Taguchi method uses orthogonal arrays to test a group of factors. The factors that can influence an algorithm's performance can be divided into 2 categories: (1) signal factors; (2) noise factors. Taguchi method attempts to find the best level combination of signal factors to minimize the noise factors' impact in response. The smaller-the-better type of signal to noise ratio (S/N) is used in this paper:

$$S/N = -10 \times \log\left(\frac{S\left(Y^2\right)}{n}\right) \tag{38}$$

where Y denotes the response, and n denotes the number of orthogonal arrays. $S(Y^2)$ represents the sum of responses under a specific combination of signal factors. To consider the convergence and the diversity at the same time, we adopt a metric called combinatorial ratio (C.R.) proposed in Ding et al. (2018) as the response variable. C.R. uses the IGD and Hypervolume, which are defined in Section 5.2. C.R. can be calculated as:

$$C.R. = \frac{IGD}{HV} \tag{39}$$

Six levels of each factor are considered. The levels of the parameters for each algorithm are shown in Table 5. For comparison algorithms, PS denotes the population size, NOG denotes the maximum number of generations, CP denotes the crossover probability, MP denotes the mutation probability, TS denotes the tournament size, NE denotes the neighborhood size, other parameters are of the same meaning as in its corresponding paper. For the proposed algorithm, the definitions of the parameters are the same as presented in Section 4.1. To make a fair comparison, the Function Evaluation Number (FEN) is set to 120000 for each level of parameters. For the comparison algorithm, the population size remains the same from the beginning to the end, so the number of generations can be calculated as $NOG = \frac{FEN}{PS}$. For the proposed algorithm, the initial population size P_0 will increase to the maximum population size P_{max} in a few generations. Given a specific FEN, it is hard to compute the exact value of NOG of the proposed method, so we use the maximum population size ($NOG = \frac{FEN}{P_{max}}$) to make sure that the actual function evaluation number is always no larger than the comparison algorithms. The L_{48} design is selected for NSGA-II(IM), NGTA, and MOEA/D. The L_{56} design is selected for MOPSO. The L_{62} design is selected for PS-MOIWO. Test each algorithm with instance-5. The parameter level with the maximum mean of S/N is chosen as the optimal level. The optimal values of the parameters for each algorithm are presented in Table 6.

Table 5: Algorithm parameter ranges along with their levels.

Algorithm	Parameter	Parameter level					
		Level1	Level2	Level3	Level4	Level5	Level6
PS-MOIWO	NOG	1200	1000	705	600	521	480
	P_0	70	90	120	150	180	200
	P_{max}	100	120	170	200	230	250
	$\sigma_{initial}$	0.1	0.15	0.2	0.25	0.3	0.35
	σ_{final}	0.01	0.02	0.03	0.04	0.05	0.06
	S_{min}	1	2	3	4	5	6
	S_{max}	2	4	6	8	10	12
	n	1	2	3	4	5	6
NTGA	NOG	1200	1000	705	600	521	480
	PS	100	120	170	200	230	250
	CP	0.4	0.5	0.6	0.7	0.8	0.9
	MP	0.05	0.1	0.15	0.2	0.25	0.30
	TS	4	5	6	7	8	9
NSGA-II(IM)	NOG	1200	1000	705	600	521	480
	PS	100	120	170	200	230	250
	CP	0.4	0.5	0.6	0.7	0.8	0.9
	MP	0.05	0.1	0.15	0.2	0.25	0.30
	TS	4	5	6	7	8	9
MOPSO	NOG	1200	1000	705	600	521	480
	PS	100	120	170	200	230	250
	$c1$	1.0	1.25	1.50	1.75	2.0	2.25
	$c2$	1.0	1.5	2.0	2.25	2.5	2.75
	w	0.4	0.5	0.6	0.7	0.8	0.9
	REP	50	70	80	90	110	130
	G	6	7	8	9	10	11
MOEA/D	NOG	1200	1000	705	600	521	480
	PS	100	120	170	200	230	250
	CP	0.4	0.5	0.6	0.7	0.8	0.9
	MP	0.05	0.1	0.15	0.2	0.25	0.30
	NE	3	4	5	6	7	8

Table 6: Optimal value of the parameters.

Algorithm	Parameter	Optimal value
PS-MOIWO	NOG	600
	P_0	150
	P_{max}	200
	$\sigma_{initial}$	0.2
	σ_{final}	0.04
	S_{min}	2
	S_{max}	7
	n	4
NTGA	NOG	705
	PS	170
	CP	0.6
	MP	0.1
	TS	8
NSGA-II(IM)	NOG	1000
	PS	120
	CP	0.6
	MP	0.25
	TS	7
MOPSO	NOG	521
	PS	230
	$c1$	1.75
	$c2$	2.25
	w	0.5
	REP	70
	G	8
MOEA/D	NOG	600
	PS	200
	CP	0.7
	MP	0.15
	NE	7

5.5. Performance evaluation

In this section, the performance comparison of the algorithms is presented. Firstly, we use the metrics defined in Section 5.2 to compare the algorithms. All the results presented are obtained by executing 20 independent runs of each algorithm on each instance. From the numeric point of view, the average and standard deviation values of C-metric, SP, IGD and HV are presented in Tables 7 to 10. Wilcoxon's rank sum test at a 5% significance level is conducted to test the significance of the difference between the mean metric values yielded by the proposed algorithm and the comparison algorithms. The symbols †, §, and ≈ indicate that the performance of the proposed PS-MOIWO algorithm is better than, worse than, and similar to that of the comparison algorithm according to Wilcoxon's rank sum test. In each table, the number in parentheses is the standard deviation; the bold data represent the best value in a row. Figure 12 shows the Pareto fronts of a single run of each algorithm on 3 instances.

Concerning the C-metric values in Table 7, the proposed algorithm PS-MOIWO has the best performance. It obtains the best value among all the 10 instances. To show the relative relations between the algorithms better, the mean value of the C-metric of all algorithms for each test instance is plotted in Figure 8. The solutions obtained by PS-MOIWO generally dominate above 80% of those obtained by the competitors, which indicates a significant advantage over the other competitors on all the test instances. NGTA performs better than NSGA-II(IM), MOPSP, and MOEA/D. MOPSO and MOEA/D have similar performance in C-metric. Neither of them performs significantly better than the other.

Secondly, we use Spacing (SP) to compare the algorithms. The value of SP measures the distribution and diversity of the solutions. Table 8 presents the details of SP value on each instance. Figure 9 provides a comprehensive view of the mean value of SP for each instance. In general, the proposed PS-MOIWO obtains 2 best values among the 10 instances. MOEA/D obtains 3 best solutions. NSGA-II(IM) and MOPSO obtain 2 best solutions, respectively. NGTA obtains only 1 best solution. As we can see from Table 8 and Figure 9, there is no significant difference in SP between all the algorithms. Generally, the algorithm with higher quality Pareto front trends to get a lower value of SP.

About the values of Inverted Generational Distance (IGD) in Table 9, the proposed algorithm PS-MOIWO obtains all the best solutions among all the 10 instances. In other words, PS-MOIWO obtains better results than the other competitors. Figure 11 presents the mean value of IGD for each test instance. For PS-MOIWO, a clear advantage can be seen from Figure 11 over the other three algorithms. As illustrated in Section 5.3, the complexity of instance 1 to 10 increases gradually. According to Figure 11, we can conclude that: with the increase of the complexity of the instance, the advantage of PS-MOIWO increases gradually in general. This is mainly because the PS-MOIWO makes better use of the problem information. NGTA has the second-best performance on IGD.

Finally, we compare the solutions obtained by each algorithm by using the Hypervolume (HV). HV reflects both the convergence and diversity of the solutions at the same time. Table 8 shows the details of the HV value for each instance. Figure 9 gives a direct view of the mean value of HV for each instance. The proposed algorithm PS-MOIWO obtains 6 of the best solutions among all 10 instances, which shows that it has the best performance. NGTA and NSGA-II(IM) obtain 2 best solutions, respectively. NGTA obtains 4 solutions that are worse than PS-MOIWO but close to it, and NSGA-II(IM) obtains 3 solutions close to PS-MOIWO. Generally, NGTA has the second-best performance on HV. MOPSO and MOEA/D do not obtain any best solution.

Moreover, in order to show the superiority of PS-MOIWO more clearly, we plot the non-

dominated solutions of a single run for all the algorithms on instance-3,instance-6, and instance-8. In each instance, two cube regions represented by two different colors are enlarged to show more information. The graphic representations of their obtained non-dominated solutions are shown in Figure 12. It is apparent in Figure 12 that PS-MOIWO obtains higher quality non-dominated solutions than the other competitors.

Furthermore, the result of convergence analysis on a random single run of all algorithms is presented in Figure 13. In Figure 13, we show the convergence curve of HV for the 10 instances. Obviously, we can see that all five algorithms show good convergence in terms of HV but fall into slight degradation occasionally. This is most probably because it is affected by the failure to maintain the diversity of the solutions. PS-MOIWO has the best HV value in the beginning of the iteration in most instances. This is mainly because the proposed population initialization algorithm can make better use of the problem information to generate a higher quality initial population. Nevertheless, PS-MOIWO obtains high-quality Pareto fronts and good convergence.

In summary, we have evaluated the performance of the proposed algorithm PS-MOIWO with the other four multi-objective optimization algorithms. Four performance metrics have been introduced for the evaluation of the solution quality. Regarding the Set Coverage and Inverted General Distance, PS-MOIWO outperforms the competitors, which shows that the solutions obtained by PS-MOIWO have the highest quality. Considering that the PS-MOIWO has the highest quality Pareto front, it also shows good performance on SP and HV. The convergence analysis clearly shows that PS-MOIWO attains a fast convergence with high-quality solutions. The reasons why PS-MOIWO is effective can be concluded below:

(1) The framework of the PS-MOIWO algorithm is designed specifically to fit the nature of the reconnaissance mission scheduling problem while other algorithms are not. The appropriate solution representation scheme and the specially designed population initialization algorithm effectively obtain a high-quality population.

(2) The proposed constraint handling technique can tackle the objective function and the constraints simultaneously, which helps to get better solutions in the same number of iterations. The proper design of the local search method improves the exploration ability of the proposed algorithm.

Table 7: Performance comparison of the algorithms from the view point of Set Coverage (C-metric).

	PS-MOIWO	NTGA	NSGA-II(IM)	MOPSO	MOEA/D
instance-1	**0.275(0.019)**	0.864(0.103)†	0.842(0.086)†	0.865(0.095)†	0.906(0.071)†
instance-2	**0.180(0.012)**	0.850(0.053)†	0.918(0.092)†	0.862(0.068)†	0.929(0.094)†
instance-3	**0.214(0.021)**	0.841(0.067)†	0.915(0.089)†	0.826(0.067)†	0.917(0.086)†
instance-4	**0.041(0.003)**	0.884(0.047)†	0.892(0.049)†	0.886(0.061)†	0.934(0.079)†
instance-5	**0.120(0.011)**	0.891(0.054)†	0.906(0.053)†	0.901(0.077)†	0.902(0.061)†
instance-6	**0.117(0.013)**	0.853(0.070)†	0.872(0.102)†	0.918(0.072)†	0.898(0.086)†
instance-7	**0.105(0.012)**	0.867(0.063)†	0.878(0.058)†	0.930(0.086)†	0.885(0.084)†
instance-8	**0.091(0.011)**	0.934(0.054)†	0.874(0.102)†	0.934(0.057)†	0.937(0.106)†
instance-9	**0.055(0.005)**	0.917(0.087)†	0.867(0.078)†	0.939(0.051)†	0.949(0.074)†
instance-10	**0.130(0.008)**	0.929(0.072)†	0.875(0.062)†	0.914(0.061)†	0.924(0.103)†
†/§/≈	–	10/0/0	10/0/0	10/0/0	10/0/0

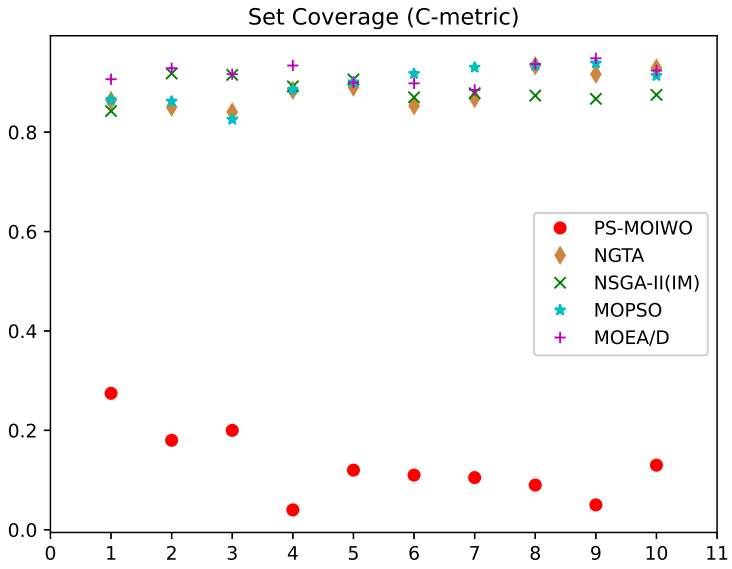


Figure 8: Mean value of C-metric with respect to test instance.

Table 8: Performance comparison of the algorithms from the view point of Spacing (SP).

	PS-MOIWO	NTGA	NSGA-II(IM)	MOPSO	MOEA/D
instance-1	4.399(0.457)	4.549(0.491)≈	5.107(0.480)§	4.604(0.525)≈	**5.155(0.284)**§
instance-2	**5.131(0.590)**	4.983(0.398)≈	5.061(0.516)≈	4.739(0.554)†	5.110(0.255)≈
instance-3	5.367(0.558)	5.376(0.349)≈	5.446(0.654)≈	5.380(0.581)≈	**5.761(0.622)**§
instance-4	6.132(0.515)	6.236(0.524)≈	6.152(0.455)≈	**7.109(0.691)**§	5.697(0.581)†
instance-5	8.136(0.928)	8.021(0.786)≈	**8.875(0.862)**§	8.851(0.858)§	8.595(1.014)≈
instance-6	8.051(0.749)	8.081(0.743)≈	**9.306(0.568)**§	8.582(0.712)§	8.993(1.074)§
instance-7	**8.659(0.892)**	8.464(0.863)≈	8.151(0.562)≈	8.388(0.596)≈	8.598(0.464)≈
instance-8	10.750(0.582)	**11.375(0.660)**≈	10.385(0.769)≈	10.620(1.126)≈	11.255(0.642)≈
instance-9	11.208(0.572)	9.553(0.764)†	11.666(0.898)≈	10.525(1.158)†	**12.449(0.946)**§
instance-10	9.125(0.547)	9.654(0.927)≈	11.252(0.776)§	**11.594(0.638)**§	10.797(0.723)§
†/§/≈	–	1/0/9	0/4/6	2/4/4	1/5/4

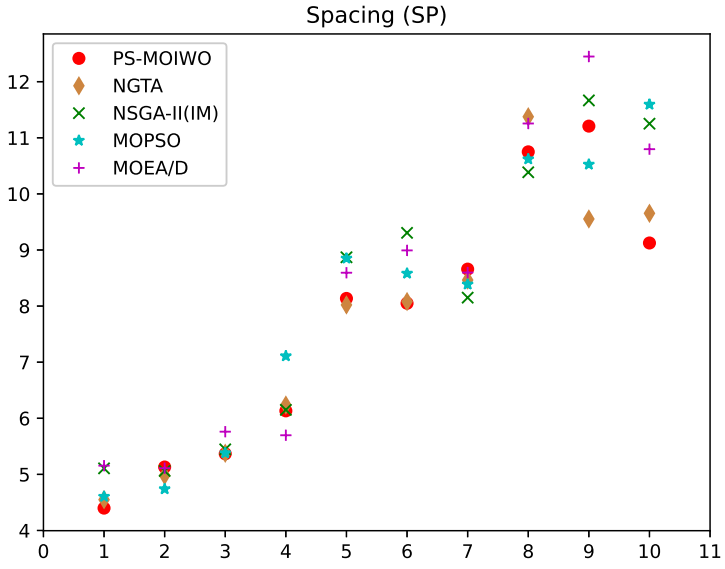


Figure 9: Mean value of SP with respect to test instance.

Table 9: Performance comparison of the algorithms from the view point of Inverted Generational Distance (IGD).

	PS-MOIWO	NTGA	NSGA-II(IM)	MOPSO	MOEA/D
instance-1	**52.986(4.292)**	62.947(4.973)†	55.179(3.531)≈	60.059(7.027)†	69.570(5.566)†
instance-2	**54.445(6.207)**	71.347(7.420)†	99.146(9.221)†	61.594(7.330)†	95.344(5.722)†
instance-3	**45.899(3.718)**	64.440(7.024)†	59.355(6.826)†	81.292(7.235)†	87.066(6.791)†
instance-4	**73.038(4.528)**	105.566(6.017)†	118.002(10.148)†	113.988(12.197)†	129.886(11.694)†
instance-5	**102.661(8.213)**	145.025(10.877)†	155.786(13.865)†	176.434(10.057)†	171.966(8.942)†
instance-6	**100.542(9.853)**	158.664(9.837)†	169.124(19.957)†	198.304(11.105)†	185.431(16.503)†
instance-7	**130.992(10.741)**	183.558(11.197)†	211.029(20.259)†	235.779(23.578)†	224.224(13.678)†
instance-8	**171.284(11.647)**	324.828(25.012)†	336.060(18.483)†	387.595(37.209)†	368.696(20.647)†
instance-9	**168.985(17.574)**	357.954(37.227)†	353.282(20.490)†	395.941(44.741)†	389.503(30.771)†
instance-10	**209.609(21.170)**	411.913(20.596)†	403.767(25.034)†	495.923(56.535)†	517.387(38.804)†
†/§/≈	–	10/0/0	9/0/1	10/0/0	10/0/0

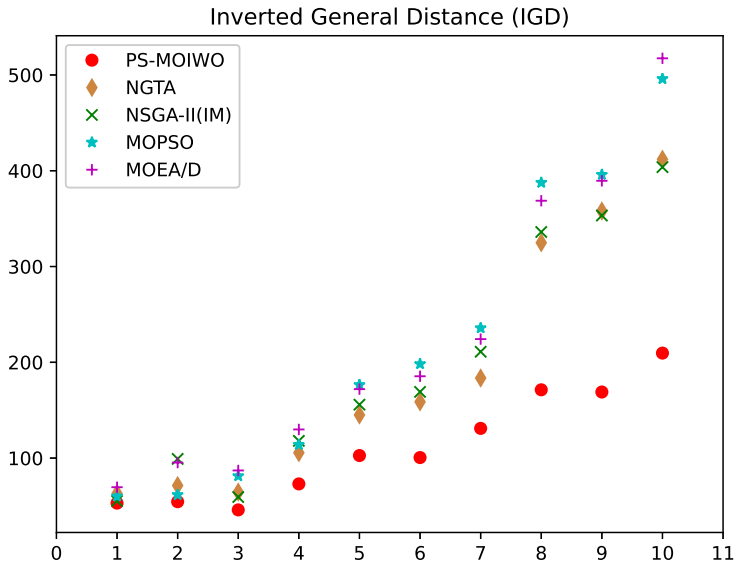


Figure 10: Mean value of IGD with respect to test instance.

Table 10: Performance comparison of the algorithms from the view point of Hypervolume (HV).

	PS-MOIWO	NTGA	NSGA-II(IM)	MOPSO	MOEA/D
instance-1	**3.83e+13(3.14e+12)**	3.75e+13(4.39e+12)≈	3.67e+13(3.37e+12)≈	2.90e+13(2.84e+12)†	3.03e+13(2.58e+12)†
instance-2	4.41e+13(2.56e+12)	**4.57e+13(2.97e+12)**≈	4.04e+13(3.59e+12)†	3.10e+13(2.61e+12)†	2.70e+13(1.87e+12)†
instance-3	**3.09e+13(2.56e+12)**	2.41e+13(1.23e+12)†	2.79e+13(2.20e+12)†	2.15e+13(1.59e+12)†	2.29e+13(1.33e+12)†
instance-4	**5.82e+13(6.41e+12)**	5.47e+13(2.96e+12)†	3.99e+13(4.35e+12)†	5.68e+13(6.08e+12)≈	4.33e+13(2.73e+12)†
instance-5	**1.49e+14(1.45e+13)**	1.23e+14(6.52e+12)†	1.29e+14(9.39e+12)†	1.16e+14(6.84e+12)†	1.27e+14(1.13e+13)†
instance-6	2.24e+14(1.86e+13)	1.77e+14(1.73e+13)†	**2.36e+14(2.17e+13)**≈	2.13e+14(1.70e+13)≈	1.84e+14(1.60e+13)†
instance-7	3.74e+14(2.88e+13)	3.64e+14(3.35e+13)≈	**4.01e+14(4.69e+13)**§	3.44e+14(3.96e+13)†	3.69e+14(1.96e+13)≈
instance-8	**1.42e+15(1.25e+14)**	1.23e+15(8.09e+13)†	1.17e+15(1.39e+14)†	1.07e+15(1.24e+14)†	1.40e+15(1.02e+14)≈
instance-9	3.56e+15(3.24e+14)	**3.66e+15(4.17e+14)**≈	3.57e+15(4.03e+14)≈	3.21e+15(3.34e+14)†	3.55e+15(2.17e+14)≈
instance-10	**2.34e+15(1.94e+14)**	1.79e+15(9.13e+13)†	2.00e+15(1.78e+14)†	1.38e+15(9.37e+13)†	2.03e+15(1.68e+14)†
†/§/≈	–	6/0/4	6/1/3	8/0/2	7/0/3

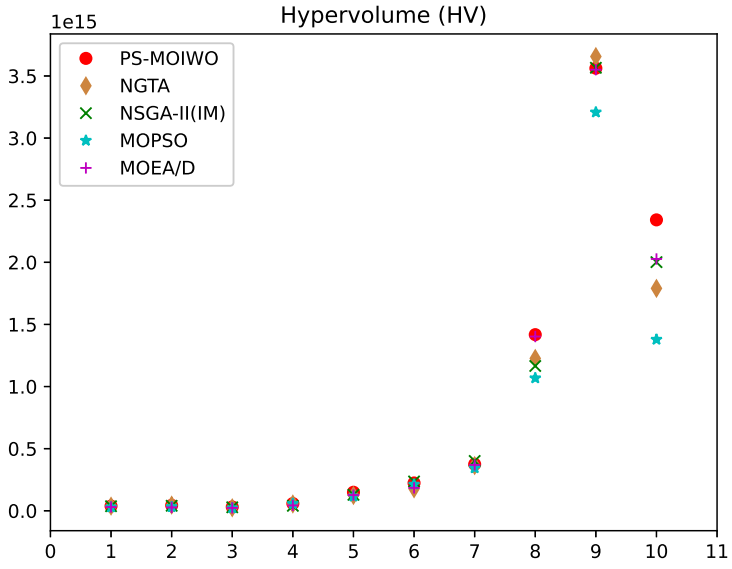
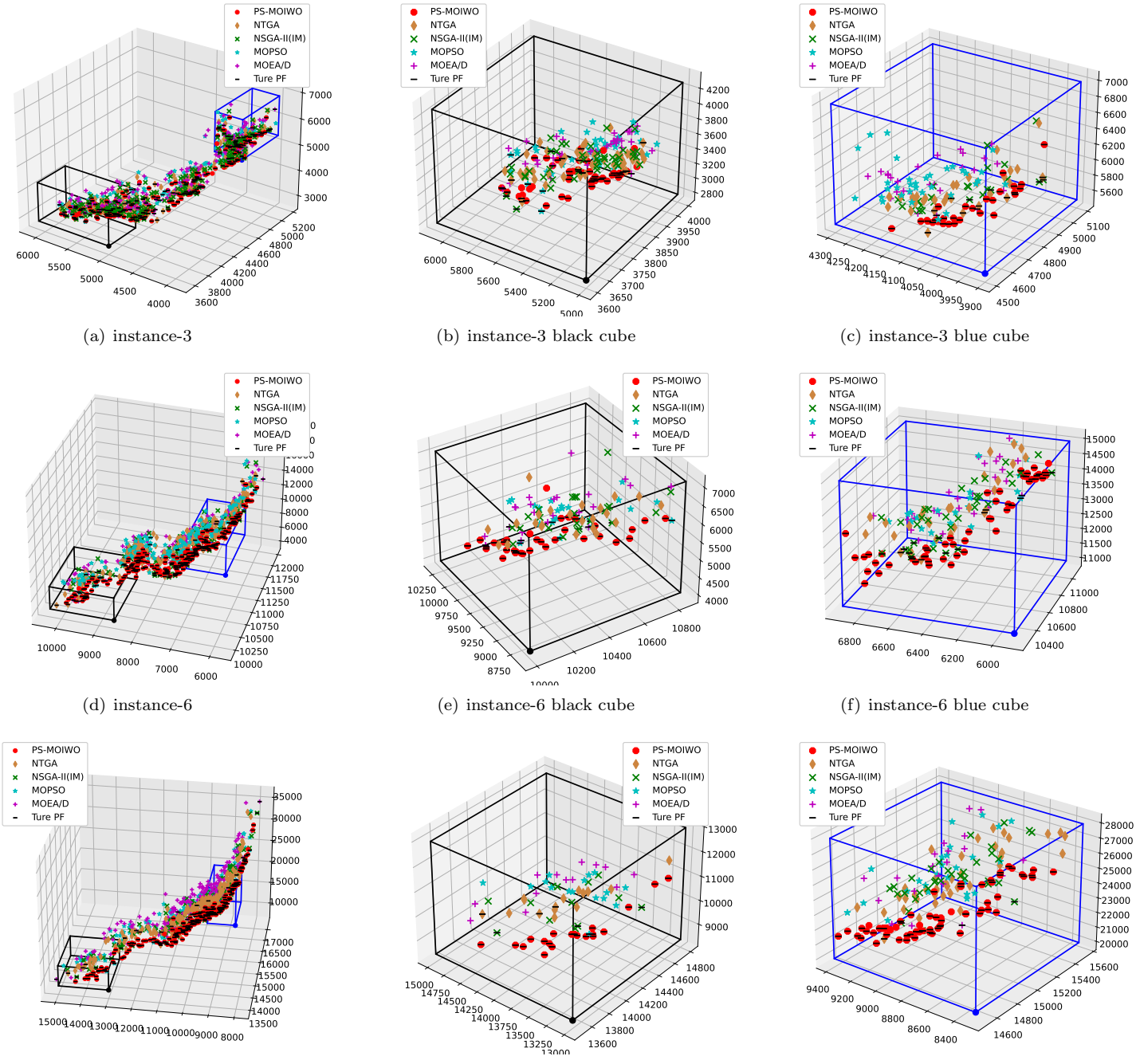


Figure 11: Mean value of HV with respect to test instance.

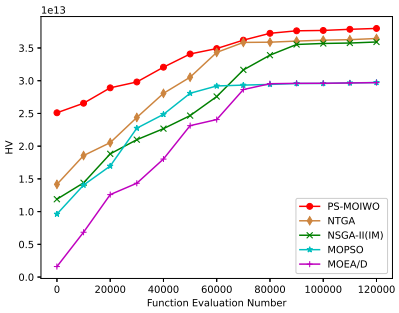


(a) instance-3 (b) instance-3 black cube (c) instance-3 blue cube

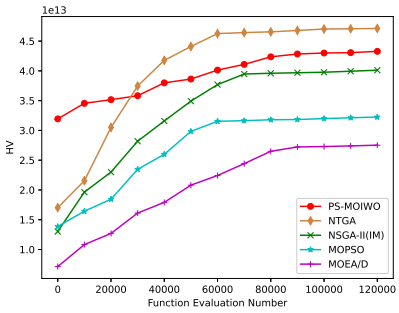
(d) instance-6 (e) instance-6 black cube (f) instance-6 blue cube

(g) instance-8 (h) instance-8 black cube (i) instance-8 blue cube

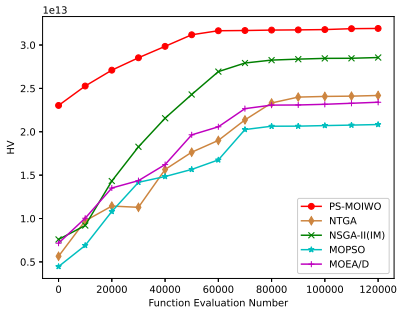
Figure 12: Non-dominated solutions of a single run of all the algorithms for each instance.



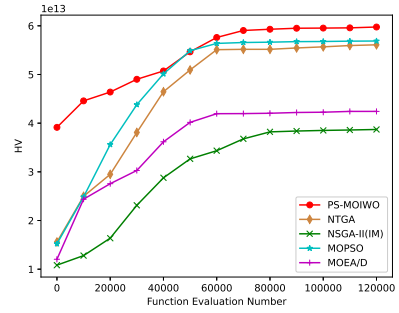
(a) instance-1



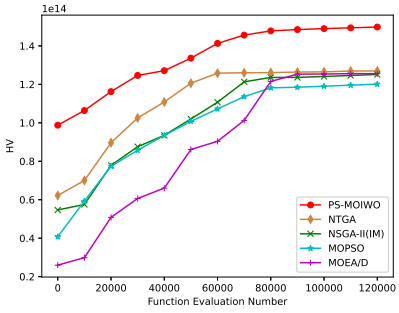
(b) instance-2



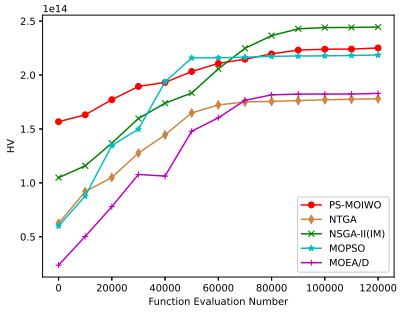
(c) instance-3



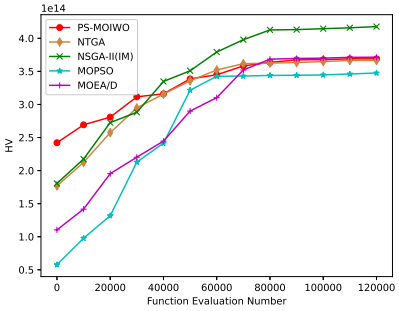
(d) instance-4



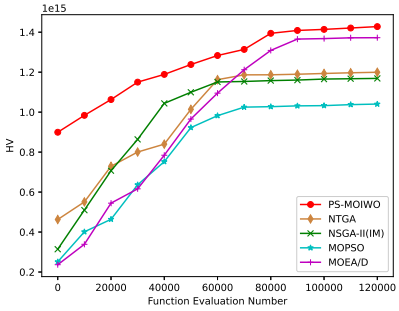
(e) instance-5



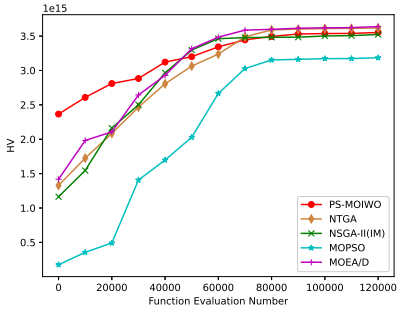
(f) instance-6



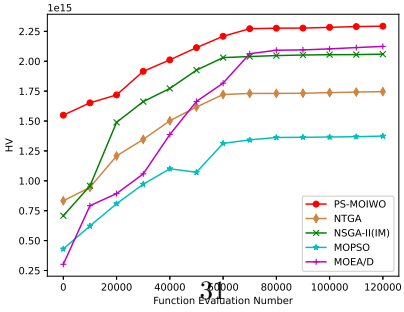
(g) instance-7



(h) instance-8



(i) instance-9



(j) instance-10

Figure 13: Convergence analysis with Hypervolume of a single run of all the algorithms for each instance.

6. Conclusion

The multi-agent system reconnaissance mission scheduling problem is investigated in this paper. The problem is modeled as an extension of Multi-Mode Multi-Skill Resource-Constrained Project Scheduling Problem. The mode information of the reconnaissance mission is not known in advance. The existing algorithms need the exact mode information to work, which can not fit the reconnaissance mission planning problem's characteristics. Even though we know the exact mode information in advance, each agent masters different skill levels, using the traditional method is ineffective. To deal with the reconnaissance mission planning problem, the PS-MOIWO is proposed in this paper. A new solution representation scheme and the corresponding initialization method are designed. A local search procedure and a self-adaptive penalty-based constraint handling method are proposed to improve the population's quality using the problem-specific knowledge. We perform diverse comparisons to validate the proposed algorithm. The results show that PS-MOIWO has competitive performance in dealing with reconnaissance mission scheduling problems.

Our future research plan includes the following two aspects: (1) As multi-objective optimization is always time-consuming, we will try to accelerate the proposed algorithm using the parallel computing method. (2) Accidents may occur in the process of performing a reconnaissance mission. The reactive planning method for reconnaissance missions needs to be studied. The method that can obtain a robust baseline schedule is another way to deal with such accidents or disruptions in reconnaissance missions.

References

Abdolshah, M. (2014). A review of resource-constrained project scheduling problems (RCPSP) approaches and solutions. *International Transaction Journal of Engineering, Management, & Applied Sciences & Technologies*, *5*, 253–286.

Alcaraz, J., Maroto, C., & Ruiz, R. (2003). Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. *Journal of the Operational Research Society*, *54*, 614–626.

Almeida, B., Correia, I., & Saldanha-da Gama, F. (2015). An instance generator for the multi-skill resource-constrained project scheduling problem. *Faculdade de Ciências da Universidade de Lisboa-Centro de Matemática, Aplicações Fundamentais e Investigação Operacional*, .

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.

Almeida, B. F., Correia, I., & Saldanha-da Gama, F. (2019). Modeling frameworks for the multi-skill resource-constrained project scheduling problem: a theoretical and empirical comparison. *International Transactions in Operational Research*, *26*, 946–967.

Audet, C., Bigeon, J., Cartier, D., Le Digabel, S., & Salomon, L. (2018). Performance indicators in multiobjective optimization. *Optimization Online*, .

Bellenguez-Morineau, O., & Néron, E. (2007). A branch-and-bound method for solving multi-skill project scheduling problem. *RAIRO-Operations Research*, *41*, 155–170.

Blank, J., Deb, K., & Mostaghim, S. (2017). Solving the bi-objective traveling thief problem with multi-objective evolutionary algorithms. In *International Conference on Evolutionary Multi-Criterion Optimization* (pp. 46–60). Springer.

Blazewicz, J., Lenstra, J. K., & Kan, A. R. (1983). Scheduling subject to resource constraints: classification and complexity. *Discrete applied mathematics*, *5*, 11–24.

Bloss, R. (2014). Robot innovation brings to agriculture efficiency, safety, labor savings and accuracy by plowing, milking, harvesting, crop tending/picking and monitoring. *Industrial Robot: An International Journal*, *4*, 182–195.

Chakrabortty, R. K., Abbasi, A., & Ryan, M. J. (2019). Multi-mode resource-constrained project scheduling using modified variable neighborhood search heuristic. *International Transactions in Operational Research*, *20*, 102–111.

Chakrabortty, R. K., Sarker, R. A., & Essam, D. L. (2016). Multi-mode resource constrained project scheduling under resource disruptions. *Computers & Chemical Engineering*, *88*, 13–29.

Chen, H., Zhou, Y., He, S., Ouyang, X., & Guo, P. (2013). Invasive weed optimization algorithm for solving permutation flow-shop scheduling problem. *Journal of Computational and Theoretical Nanoscience*, *10*, 708–713.

Chen, S., Lan, Y., Zhou, Z., Li, J., Ouyang, F., Xu, X., Yao, W. et al. (2019). Test and evaluation for flight quality of aerial spraying of plant protection UAV. *Journal of South China Agricultural University*, *40*, 89–96.

Chen, W.-N., & Zhang, J. (2012). Ant colony optimization for software project scheduling and staffing with an event-based scheduler. *IEEE Transactions on Software Engineering*, *39*, 1–17.

Cheng, J., Fowler, J., Kempf, K., & Mason, S. (2015). Multi-mode resource-constrained project scheduling problems with non-preemptive activity splitting. *Computers & Operations Research*, *53*, 275–287.

Choi, U., & Ahn, J. (2020). Imitation learning-based unmanned aerial vehicle planning for multi-target reconnaissance under uncertainty. *Journal of Aerospace Information Systems*, *17*, 36–50.

Chu, X., & Yu, X. (2018). Improved Crowding Distance for NSGA-II. *arXiv preprint arXiv:1811.12667*, .

Coello, C. C., & Lechuga, M. S. (2002). MOPSO: A proposal for multiple objective particle swarm optimization. In *Proceedings of the Congress on Evolutionary Computation* (pp. 1051–1056). IEEE volume 2.

Das, P. P., & Acharyya, S. (2011). Simulated annealing variants for solving resource constrained project scheduling problem: a comparative study. In *14th International Conference on Computer and Information Technology (ICCIT)* (pp. 469–474). IEEE.

Ding, S., Chen, C., Xin, B., & Pardalos, P. M. (2018). A bi-objective load balancing model in a distributed simulation system using NSGA-II and MOPSO approaches. *Applied Soft Computing*, *63*, 249–267.

Ghalenoei, M. R., Hajimirsadeghi, H., & Lucas, C. (2009). Discrete invasive weed optimization algorithm: application to cooperative multiple task assignment of UAVs. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 28th Chinese Control Conference* (pp. 1665–1670). IEEE.

Gustavsson, P., & Syberfeldt, A. (2018). A new algorithm using the non-dominated tree to improve non-dominated sorting. *Evolutionary computation*, *26*, 89–116.

Heidari, A. A., Abbaspour, R. A., & Jordehi, A. R. (2017). Gaussian bare-bones water cycle algorithm for optimal reactive power dispatch in electrical power systems. *Applied Soft Computing*, *57*, 657–671.

Hill, A., Lalla-Ruiz, E., Voß, S., & Goycoolea, M. (2019). A multi-mode resource-constrained project scheduling reformulation for the waterway ship scheduling problem. *Journal of Scheduling*, *22*, 173–182.

Hosseinian, A. H., Baradaran, V., & Bashiri, M. (2019). Modeling of the time-dependent multi-skilled RCPSP considering learning effect: An evolutionary solution approach. *Journal of Modelling in Management*, *14*, 521–558.

Hsu, C.-C., & Kim, D. S. (2005). A new heuristic for the multi-mode resource investment problem. *Journal of the Operational Research Society*, *56*, 406–413.

John A. Richards, A. T., Anirudh Patel (2019). *Autonomous Multi-Platform Sensor Scheduling for Intelligence Surveillance and Reconnaissance*. Technical Report Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).

Kadrou, Y., & Najid, N. M. (2006). A new heuristic to solve RCPSP with multiple execution modes and Multi-Skilled Labor. In *The Proceedings of the Multiconference on Computational Engineering in Systems Applications* (pp. 1302–1309). IEEE volume 2.

Kazemipoor, H., Tavakkoli-Moghaddam, R., & Shahnazari-Shahrezaei, P. (2002). Solving a mixed-integer linear programming model for a multi-skilled project scheduling problem by simulated annealing. *Management Science Letters*, *2*, 681–688.

Kim, D., Xue, L., Li, D., Zhu, Y., Wang, W., & Tokuta, A. O. (2017). On theoretical trajectory planning of multiple drones to minimize latency in search-and-reconnaissance operations. *IEEE Transactions on Mobile Computing*, *16*, 3156–3166.

Kundu, D., Suresh, K., Ghosh, S., Das, S., Panigrahi, B. K., & Das, S. (2011). Multi-objective optimization with artificial weed colonies. *Information Sciences*, *181*, 2441–2454.

Laszczyk, M., & Myszkowski, P. B. (2019). Improved selection in evolutionary multi–objective optimization of multi–skill resource–constrained project scheduling problem. *Information Sciences*, *481*, 412–431.

Laszlo, B., Agoston, R., & Xu, Q. (2018). Conceptual approach of measuring the professional and economic effectiveness of drone applications supporting forest fire management. *Procedia Engineering*, *211*, 8–17.

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 & Chemical Engineering*, *88*, 157–169.

Mehrabian, A. R., & Lucas, C. (2006). A novel numerical optimization algorithm inspired from weed colonization. *Ecological Informatics*, *1*, 355–366.

Myszkowski, P. B., Laszczyk, M., & Kalinowski, D. (2017). Co-evolutionary algorithm solving multi-skill resource-constrained project scheduling problem. In *Federated Conference on Computer Science and Information Systems (FedCSIS)* (pp. 75–82). IEEE.

Myszkowski, P. B., Olech, Ł. P., Laszczyk, M., & Skowroński, M. E. (2018). Hybrid differential evolution and greedy algorithm (DEGR) for solving multi-skill resource-constrained project scheduling problem. *Applied Soft Computing*, *62*, 1–14.

Myszkowski, P. B., & Siemieński, J. J. (2016). GRASP Applied to Multi–Skill Resource–Constrained Project Scheduling Problem. In *International Conference on Computational Collective Intelligence* (pp. 402–411). Springer.

Myszkowski, P. B., Skowroński, M. E., Olech, Ł. P., & Oślizło, K. (2015). Hybrid ant colony optimization in solving multi-skill resource-constrained project scheduling problem. *Soft Computing*, *19*, 3599–3619.

Néron, E. (2002). Lower bounds for the multi-skill project scheduling problem. In *Proceeding of the Eighth International Workshop on Project Management and Scheduling* (pp. 274–277).

Santos, M. A., & Tereso, A. P. (2011). On the Multi-mode, Multi-skill Resource Constrained Project Scheduling Problem–A Software Application. In *Soft Computing in Industrial Applications* (pp. 239–248). Springer.

Schott, J. R. (1995). Fault tolerant design using single and multicriteria genetic algorithm optimization. *Air Force Inst of Tech Wright-Patterson*, *20*, 510–524.

Skowroński, M. E., Myszkowski, P. B., Adamski, M., & Kwiatek, P. (2013). Tabu search approach for multi-skill resource-constrained project scheduling problem. In *Federated Conference on Computer Science and Information Systems* (pp. 153–158). IEEE.

Su, S.-B., Wang, J.-W., Zhang, L., Fang, J., & Li, F.-P. (2009). An invasive weed optimization algorithm for constrained engineering design problems. *Journal of University of Science and Technology of China*, *39*, 885–893.

Taguchi, G. (1986). Introduction to quality engineering: designing quality into products and processes, .

Tao, S., & Dong, Z. S. (2018). Multi-mode resource-constrained project scheduling problem with alternative project structures. *Computers & Industrial Engineering*, *125*, 333–347.

Van Peteghem, V., & Vanhoucke, M. (2010). A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, *201*, 409–418.

Yang, X.-S., Deb, S., Zhao, Y.-X., Fong, S., & He, X. (2018). Swarm intelligence: past, present and future. *Soft Computing*, *22*, 5923–5933.

Zhang, Q., & Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, *11*, 712–731.

Zhao, H., Wang, P.-h., Peng, X., Qian, J., & Wang, Q. (2009). Constrained optimization of combustion at a coal-fired utility boiler using hybrid particle swarm optimization with invasive weed. In *International Conference on Energy and Environment Technology* (pp. 564–567). IEEE volume 1.

Zheng, H.-y., Wang, L., & Zheng, X.-l. (2017). Teaching–learning-based optimization algorithm for multi-skill resource constrained project scheduling problem. *Soft Computing*, *21*, 1537–1548.

Zitzler, E., & Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms—a comparative case study. In *International Conference on Parallel Problem Solving from Nature* (pp. 292–301). Springer.

Zoraghi, N., Shahsavar, A., Abbasi, B., & Van Peteghem, V. (2017). Multi-mode resource-constrained project scheduling problem with material ordering under bonus–penalty policies. *Top*, *25*, 49–79.