Complex & Intelligent Systems
https://doi.org/10.1007/s40747-023-01087-3

ORIGINAL ARTICLE

A problem-specific parallel pareto local search for the reactive decision support of a special RCPSP extension

Junqi Cai[1,2] · Zhihong Peng[1,2] · Shuxin Ding[3] · Zhiguo Wang[1,2] · Yue Wei[4]

Received: 25 April 2022 / Accepted: 10 April 2023

Abstract
The disaster information collection mission should be executed after the disaster occurs to provide details for the decision-makers. During the execution of the information collection mission, some disruptions may occur and prevent the resource used for information collection from completing the mission as planned. It is difficult for decision-makers to make reactive resource scheduling plan that optimize the mission's execution time, quality, and cost at the same time under such circumstances. This article focuses on designing the reactive decision support algorithm for the disaster information collection resource scheduling, which aims to provide multi high-quality scheduling plans for decision-makers to choose. The problem studied in this article is modeled as an extension of Resource-Constrained Project Scheduling Problem (RCPSP). First, the basic problem formulation for a normal schedule and two disruption recovery models are presented. Second, a novel framework of a parallel pareto local search based on decomposition is designed to repair the schedule within the time limit. Third, two solution acceptance criteria based on constraint handling and negative correlation are specially designed to maintain high-quality population with diversity. The experiments show that the proposed method outperforms the other competitors with respect to Inverted Generational Distance, Spacing, and Hypervolume, which means that the proposed method can help decision-makers to make better decisions.

Keywords Reactive decision support · Parallel · Pareto local search · Information collection · RCPSP

Introduction

Disasters have caused huge losses to human beings. Many scholars have studied different disaster relief issues [1–4] to reduce losses and save lives. At present, decision-makers can make better decisions through decision support technology in many fields; such as ship trajectory cleansing and prediction [5], maintenance strategies making [6], bank telemarketing sales prediction [7], vehicle routing [8], and smart grid management [9]. The disaster information collection resource scheduling decision support module plays a vital role in modern disaster relief command and control systems, which aims to present high-quality resource allocation plans to decision-makers. A disaster information collection mission has several tasks with precedence relations. The precedence relations restrict that one task cannot be started if at least one of its predecessor tasks is not finished. This is because there are many hidden dangers, such as fire and chemical leakage after the disaster, so the precedence relations are built based on the geographic accessibility of each task to ensure the safety of the information collection agents. Each information collection task needs some skill provided by the information collection agent. The agent represents a team with fewer than 5 people, which is the smallest unit for resource scheduling. The team can own vehicles, laser radars, UAVs, ranging instruments, and other equipment. The information collection in this article includes three types of tasks: urban area information collection, woodland area information collection, and terrain information collection. Because each type of task corresponds to different working modes of agents and uses different sensor combinations, this article models

✉ Zhihong Peng
peng@bit.edu.cn

1 School of Automation, Beijing Institute of Technology, Beijing 100081, China

2 State Key Laboratory of Intelligent Control and Decision of Complex System, Beijing 100081, China

3 Signal and Communication Research Institute, China Academy of Railway Sciences Corporation Limited, Beijing 100081, China

4 Pengcheng Laboratory, Shenzhen 518055, China

Published online: 14 June 2023

the ability of agents to perform tasks as skill 1–3. Each information collection agent has a pre-defined quality value and a cost value that correspond to each task. An example of an information collection mission in disaster relief scenario is presented in Fig. 1. During the execution of the information collection mission, some unpredictable events may occur and prevent the information collection mission to be processed with the original schedule. Since the information collection agents are performing tasks in the area of interest when the disruption occurs, according to the requirements of the disaster relief decision support systems, the reactive scheduling time needs to be controlled within 60 s. The reactive scheduling of disaster information collection repairs the original schedule, which helps the decision-maker to answer the following questions: (1) each task's start time; (2) the information collection agents which are assigned to the task; (3) which kind of skill that each agent should use in each task. Three objectives are optimized at the same time, minimize the makespan, minimize the cost, and maximize the quality of the information collection mission.

The disaster information collection resource scheduling problem is modeled as an extension of resource-constrained project scheduling problem (RCPSP). RCPSP focus on building a resource allocation result for a commercial project of activities, which are constrained by a limited resource supply and the precedence relation network [10]. The information collection mission is just as the "project" in RCPSP, the "task" corresponds to the "activity" in RCPSP, and the agent is very similar to the "multi-skill resource" in RCPSP. The processing time of each activity is assumed to be fixed in RCPSP, but the information collection task's processing time is not fixed. Actually, it is a non-linear function of the resource allocated to that task. For example, both one agent or two agents can finish an information collection task, but the task completion time for two agents is shorter. In RCPSP, the resource transfer time between activities is usually set to 0. However, the task's location is different in our problem, so the resource transfer time must be considered. These new features increase the difficulty of solving the problem in this article. Using the traditional RCPSP algorithm to solve the problem in this article is time-consuming and ineffective.

Some related works are addressed in this part. Tradition algorithms for solving RCPSP can be divided into three categories: exact algorithm, heuristic algorithm, and meta-heuristic algorithm. Some articles [11–13] used the exact method to solve RCPSP and get the best solution. However, the exact algorithm is time-consuming and does not fit the reactive scheduling problem in this article.

Some articles [14–16] developed heuristic methods to solve RCPSP. Although the heuristic algorithm is really fast, it cannot deal the multi-objective optimization cases.

Since the problem in this article is a multi-objective reactive robust optimization problem, the related research progress of meta-heuristic algorithm for solving RCPSP and its extensions are introduced below. Lambrechts et al. [17] focused on the uncertainty in resource availabilities subject to unforeseen breakdowns, a robust schedule model was built that meets the project deadline and minimizes the schedule instability, and proactive strategies are proposed to solve the problem. Chen and Zhang [18] aimed to develop a two-stage model to obtain a proactive and reactive schedule in resource-constrained project scheduling problems under uncertainty, a modified tabu search was employed to ensure scheduling process execution in reactive phase. Davari and Demeulemeester [19] proposed to use the selection-based reactions and the class of buffer-based reactions to deal with the uncertainty and disruptions in resource-constrained project scheduling problem. Chakrabortty et al. [20] focused on finding a robust initial schedule that can protect itself from any possible future disruptions or resource breakdowns, and a variable neighborhood search-based heuristic algorithm was proposed to solve the problem. The above articles focused on finding a robust initial schedule, but they are not targeted to deal with reactive scheduling scenarios.

Some articles focused on the single-objective RCPSP and its extensions. Deblaere et al. [21] addressed the reactive multi-mode RCPSP, and they proposed and evaluated several dedicated exact reactive scheduling procedures as well as a tabu search heuristic for repairing a disrupted schedule under the assumption that no activity can be started before its baseline starting time. Ning et al. [22] constructed the schedule adjustment cost determined by project reactive scheduling to manage disruptions caused by the randomness of activity duration, a tabu simulated annealing, and a variable neighborhood tabu search were developed to solve the problem. Adamu et al. [23] proposed a model called hybrid-RCPSP to solve reactive project scheduling problem. Davari and Demeulemeester [24] studied the reactive resource-constrained project scheduling problem, in which the approach to get a solution to the problem was a proactive and reactive policy that is a combination of a baseline schedule and a set of required reactions. Davari and Demeulemeester [25] addressed the proactive and reactive project scheduling with stochastic duration, and a dynamic programming method was proposed to solve the problem over different classes of proactive and reactive policies. Wang et al. [26] studied the reactive strategies in the multi-project scheduling problem, and a dual population genetic algorithm was designed to solve this problem. The above articles can deal with the reactive scheduling cases, but they are all single-objective optimization problem and assume that the task processing time is fixed. Zheng et al. [27] addressed the proactive and reactive of resource-constrained project problem in which activity durations are stochastic variables, and two reactive scheduling models were proposed to repair the

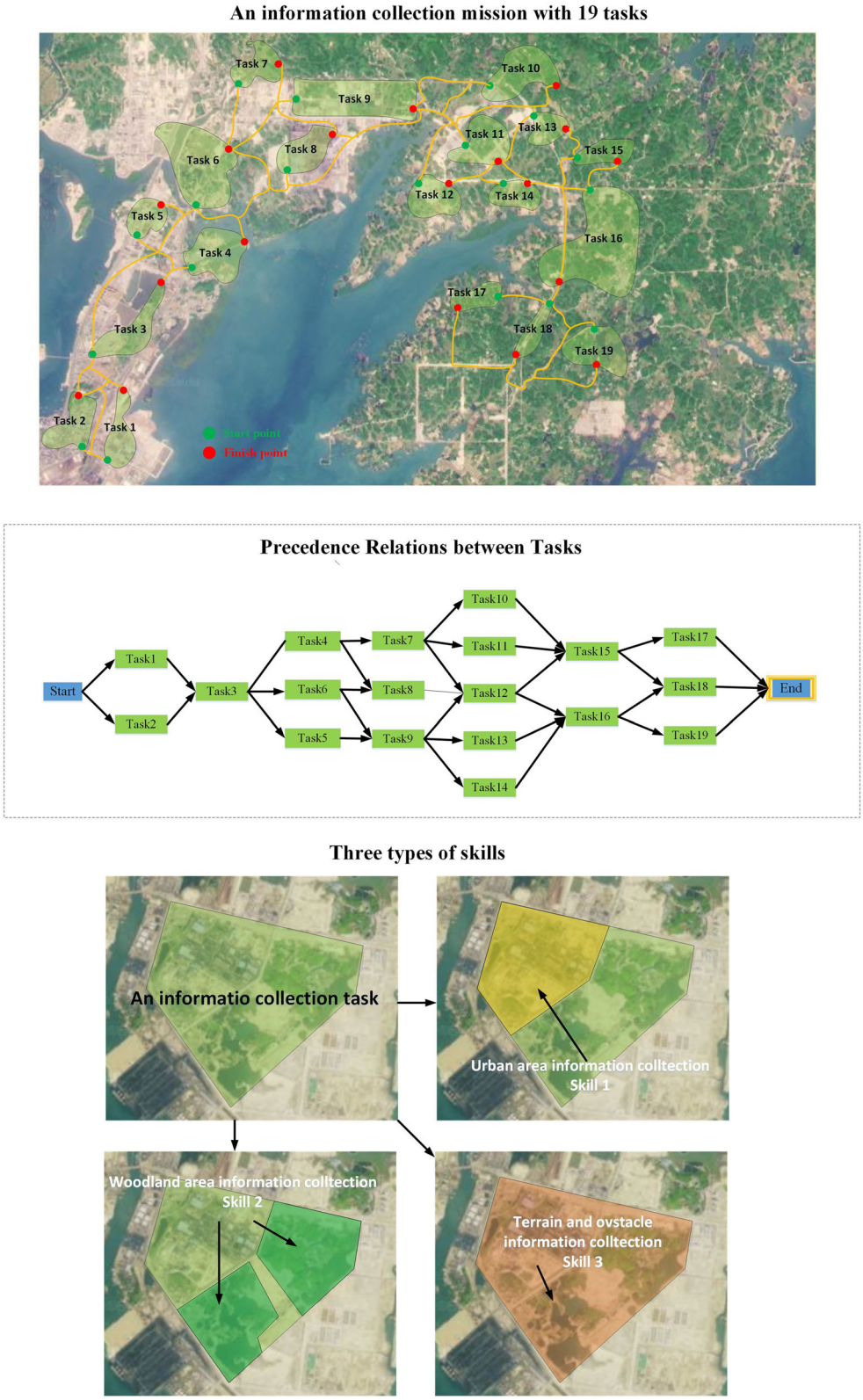


Fig. 1 Example of an information collection mission

Table 1 Differences between the existing research and this article

Article	Multi-objective	Proactive scheduling	Reactive scheduling	Varying task processing time	Transfer time
[11]	×	×	×	×	×
[12]	×	×	×	×	×
[13]	×	×	×	×	×
[14]	×	×	×	✓	✓
[15]	×	×	×	×	✓
[16]	×	×	×	×	✓
[17]	×	✓	×	×	×
[18]	×	✓	×	×	×
[19]	×	✓	×	×	×
[20]	×	✓	×	×	×
[21]	×	×	✓	×	✓
[22]	×	×	✓	✓	×
[23]	×	×	✓	×	×
[24]	×	×	✓	×	×
[25]	×	✓	✓	×	×
[26]	×	×	✓	×	×
[27]	×	✓	✓	✓	×
[28]	✓	×	×	✓	×
[29]	✓	×	×	✓	×
[30]	✓	×	×	×	×
[31]	✓	×	×	×	×
[32]	✓	×	×	×	×
[33]	×	×	×	×	✓
[34]	×	✓	×	×	×
This article	✓	×	✓	✓	✓

baseline schedules after disruptions. However, it can only deal with single-objective optimization problems.

Some articles focused on the multi-objective RCPSP and its extensions. Bagherinejad et al. [28] focused on the multi-mode multi-objective RCPSP and proposed a hybrid ant colony and genetic algorithm to solve the problem. Yeganeh and Zegordi [29] presented a multi-objective optimization approach for constructing resilient project schedules under resource constraints to cope with uncertain activity durations. Li et al. [30] proposed a multi-objective discrete Jaya algorithm to solve the multi-skill multi-objective RCPSP. Zhu et al. [31] presented an efficient decomposition-based multi-objective genetic programming hyper-heuristic algorithm to solve the multi-skill RCPSP with the objectives of minimizing the project's makespan and the total resource assignment cost at the same time. Hosseinian and Baradaran [32] considered the transfer time in multi-skill RCPSP, and built a model to optimize the project's makespan and cost simultaneously, and then, a multi-objective multi-agent optimization algorithm is proposed to get feasible schedules. The above articles dealt with multi-objective RCPSP, but the algorithms are not optimized for reactive scheduling, and the calculation time is relatively long.

Some literature focused on designing algorithm strategies. Chand et al. [33] focused on the resource-constrained project scheduling problem with resource unavailability and disruptions, and a genetic programming hyper-heuristic that can automatically evolve the priority heuristic was proposed to solve the problem. Chakrabortty et al. [34] addressed the event-based reactive approach to deal with reactive resource-constrained project scheduling problem, and an enhanced iterated greedy approach was also proposed to solve the large-scale problem. RCPSP and its extension can also be applied in command and control system [35], nuclear laboratory research planning [36], new production development [37], etc. Table 1 shows the differences and gaps between the existing research and the research in this article.

Overall, to solve the problem addressed in this article, the following characteristics should be considered: (1) multi-objectives for reactive scheduling; (2) the precedence relations between tasks; (2) using multi-skill resource (agent); (4) the transfer time is considered; (5) the processing time of a task is a non-linear function of the resource allocation

Table 2 The symbols

Symbols	Description
i, j	Index of tasks, $i, j = 0, 1, 2, \ldots, N, N+1$
l	Index of skill type, $l = 1, 2, \ldots, LN$
k	Index of information collection agent, $k = 1, 2, \ldots, K$
t	Index of the time step
N	The number of non-dummy tasks
m	The number of objectives
$V = \{0, \ldots, i, \ldots, j, \ldots, N+1\}$	Task set, task 0 and N+1 is start and end task, respectively
V^*	Incomplete tasks set after the disruption occurs
$R = \{1, \ldots, k, \ldots K\}$	Agent or resource set
K	The number of agents which can be used
R^*	Agent or resource set after the disruption occurs
$L = \{1, \ldots, l, \ldots LN\}$	Set of information collection skills
P_j, P_j^I	The indirect and direct predecessor of task j
S_j, S_j^I	The indirect and direct predecessor of task j
F_j, F_j^*	The finish time of task j in the initial and repaired schedule
ST_j, ST_j^*	The start time of task j in the initial and repaired schedule
L_j	The skills required by task j
L^k	Agent k's skill capacity
V_k	The set of tasks that can use resource k
A_j^l	The area size in task j that needs skill l to perform
A_j^{*l}	The area size in task j which needs skill l to perform
R_j	The set of agents that can be used in task j
RA_j^l	The set of agents that are allocated to task j to perform skill l
$r_l^\rho(t)$	The total skill consumption of skill l in a given time t within the initial schedule
$r_l^{*\rho}(t)$	The total skill consumption of skill l in a given time t within the repaired schedule
MR_j	The maximum number of agents that can be used in task j
Dt	The time point when the disruption happens
Δ_{ij}	The time cost for transfer agents from task i to task j
UB	The maximum makespan for the information collection mission
$T = \{0, \ldots, t, \ldots, UB\}$	Set of time steps
ES_j, LS_j	Task j's earliest and latest time to start
p_j	Task j's processing time
$p_j^{\max}$	Task j's maximum processing time
Γ_j	The preparation time for the agents which are allocated to task j
u_{kl}	The number of skill l that the agent k can provide
c_{kl}	The cost for agent k to use skill l per time step
q_{kl}	The quality contribution for agent k to use skill l per time step
s_j'	The actual start time of task j
s_{jt} (decision variable)	Equals 1 if task j is started at time t, 0 otherwise
x_{jklt} (decision variable)	Equals 1 if agent k is allocated to task j to perform skill l at time t, 0 otherwise
z_{ijk} (decision variable)	Equals 1 if agent k is transferred from task i to task j, 0 otherwise

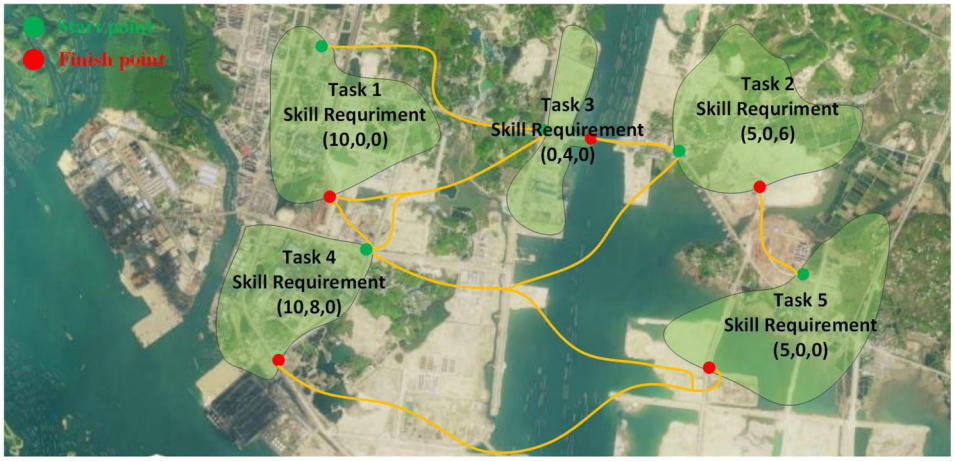
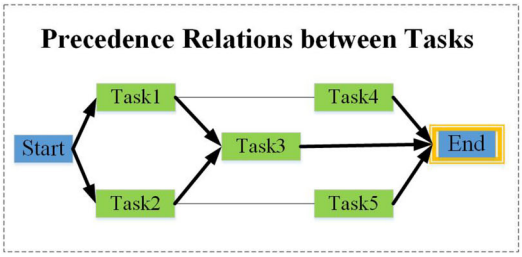


Fig. 2 Example of a small information collection mission scheduling problem



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

result. The existing RCPSP related literature did not focus on the problem addressed in this article and failed to consider the problem with the above characteristics simultaneously. The existing algorithms are ineffective and time-consuming in solving the problems in this article, and cannot be applied in the decision support system.

To solve the problem better, the above features are all considered in this article, and a parallel pareto local search is proposed to solve the multi-objective reactive information collection mission scheduling problem using the problem-specific information.

The contribution of this article is threefold: (1) the mathematical model of multi-objective information collection mission reactive scheduling problem under preempt-repeat and preempt-resume condition is established; (2) a parallel pareto local search framework is designed, in which the Tchebycheff scalar objective function and Nadir point are combined to decompose the optimization objectives into different parallel search process to speed up the reactive scheduling. (3) Two acceptance criterion based on constraint handling and negative correlation are proposed, the first acceptance criterion uses problem-specific information to guide the search process to better solutions, and the second acceptance criterion significantly increases the diversity of the Pareto fronts without compromising solution quality.

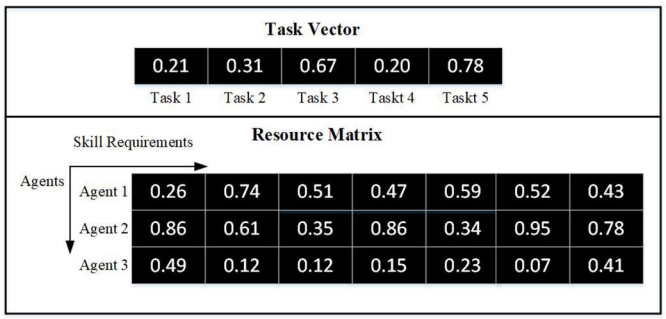
Problem formulation

Three objectives are optimized at the same time in this article: (1) minimizing the information collection mission's makespan; (2) minimizing the cost caused by the agent performing information collection tasks; (3) maximizing the mission's quality. A task on node graph $G = (V, E)$ is adopted to represent the precedence relations, in which V denotes a set of information collection tasks, and E denotes the precedence between tasks. Some assumptions [4, 35] are made in this article:

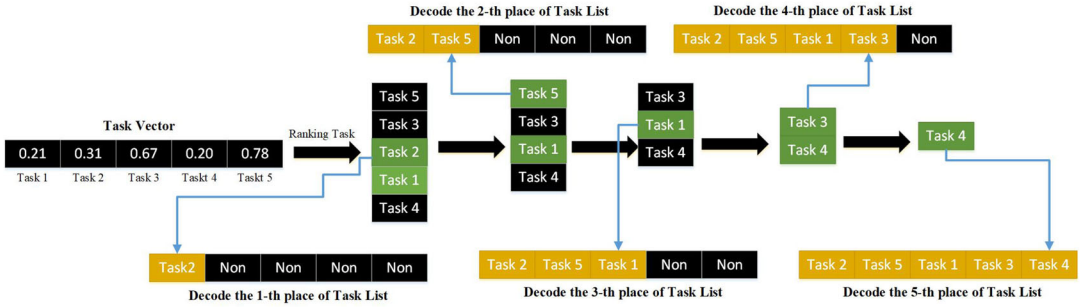
- Task 0 denotes the dummy start task, and task $N + 1$ denotes the dummy end task.
- Preemption is allowed when disruption occurs.
- Each agent can only contribute one type of the skills it masters in a task.
- The cost and quality are pre-defined for each agent corresponding to each task.
- Only after all the allocated resources are transferred to the starting point of the task, the task can be started.

Notations

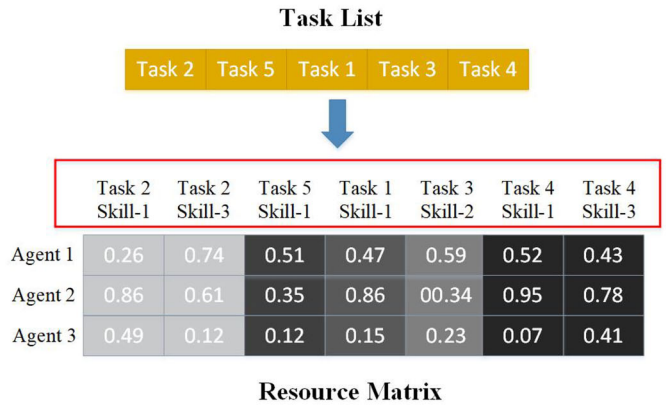
The notations are shown in Table 2.



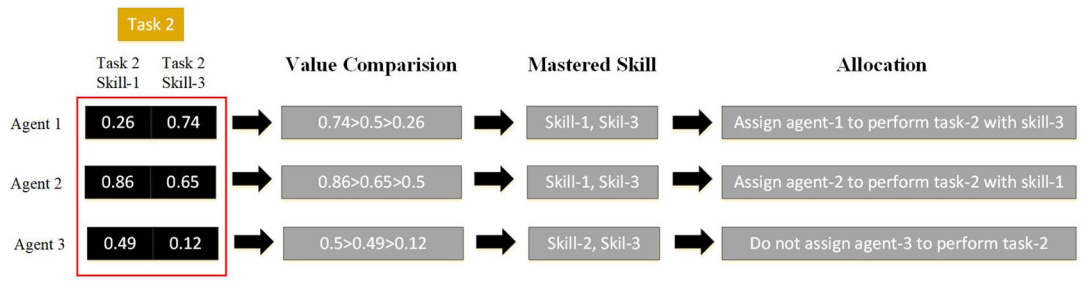
(a) Solution representation



(b) Decoding task vector to task list



(c) Column definition of resource matrix



(d) Decoding resource matrix to resource allocation result

Fig. 3 Illustration the solution representation and decoding process

Table 3 Instance parameters

Name	nAct	K	L	NC	SF	RSS	MD
1	30	25	3	2.0	0.75	[0.4, 0.5, 0.5]	3
2	35	30	3	2.0	0.75	[0.5, 0.4, 0.4]	3
3	40	30	3	2.0	0.75	[0.4, 0.4, 0.4]	3
4	45	30	3	2.5	0.75	[0.3, 0.4, 0.4]	4
5	50	30	3	2.5	0.75	[0.3, 0.3, 0.4]	4
6	55	35	3	2.5	0.75	[0.3, 0.4, 0.3]	4
7	60	40	3	3.0	1.0	[0.3, 0.3, 0.3]	5
8	65	40	3	3.0	1.0	[0.4, 0.4, 0.4]	5
9	70	45	3	3.5	1.0	[0.3, 0.4, 0.4]	5
10	75	45	3	3.5	1.0	[0.3, 0.3, 0.4]	6

Table 4 Parameter levels

Algorithm	Parameter	Parameter level		
		Level 1	Level 2	Level 3
PPLS	W	14	**16**	18
	β	0.95	**0.97**	0.99
	ϵ	0.30	**0.35**	0.40
	Mn	15	**16**	17
	σ_i	0.04	**0.06**	0.08
MOTLA	SPo	150	**200**	250
	PCr	0.15	**0.25**	0.35
	PMu	0.02	**0.04**	0.06
EMOIS	SPo	150	**200**	250
	PCr	0.1	**0.2**	0.3
	PMu	0.01	**0.03**	0.05
MOIWO	PS	150	200	**250**
	Initial sigma	0.1	**0.2**	0.3
	Final sigma	0.01	**0.03**	0.05
	S-min	2	**3**	4
	S-max	6	**8**	10

The bold numbers are represent the optimal value of each experiment

Basic formulation for normal schedules

Based on the model presented in [2, 3, 38], the formulation for normal information collection scheduling problem without disruptions is given as follows:

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

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

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

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

$$\Phi_{jklt} = \max\{p_j\cdot x_{jklt}\cdot q_{kl}, 10^{-4}\},\quad j\in V, l\in L, k\in R, t\in T, \tag{5}$$

$$\sum_{l\in L}\sum_{k\in R}\sum_{t\in T} x_{jklt} \le MR_j\ ,\quad j\in V, \tag{6}$$

$$\sum_{t=ES_j}^{LS_j} s_{jt} = 1\ ,\quad j\in V, \tag{7}$$

$$\sum_{l\in L} x_{jklt} \le s_{jt}\ ,\quad k\in R, ES_j\le t\le LS_j, j\in V, \tag{8}$$

$$\sum_{t=ES_j}^{LS_j} ts_{jt} - \sum_{t=ES_i}^{LS_i}(t+p_i)s_{it} - \Delta_{ij}z_{ijk}\ge 0,\quad i\in V\backslash\{n+1\}, j\in V\backslash P_i^I, k\in R, \tag{9}$$

$$\sum_{j\in V}\sum_{\tau=\max\{ES_j,t-p_j+1\}}^{\min\{LS_j,t\}}\sum_{l\in L} x_{jkl\tau}\le 1\ ,\quad t\in T, k\in R, \tag{10}$$

$$\sum_{j\in V} z_{ijk}\le 1\ ,\quad i\in V, k\in R_i\cap R_j, \tag{11}$$

$$\sum_{i\in V\backslash S_j^I} z_{ijk}\ge \sum_{e\in V\backslash P_j^I} z_{jek}\ ,\quad j\in V\backslash\{0\},\quad k\in R_j, \tag{12}$$

$$\sum_{t=ES_j}^{LS_j}\sum_{k\in R} x_{jklt}\cdot u_{kl}\ge \frac{A_j^l}{p_j^{\max}-\Gamma_j}\ ,\quad j\in V\backslash\{0,n+1\}, l\in L, \tag{13}$$

$$\sum_{l\in L}\sum_{t=ES_j}^{LS_j} x_{jklt} = \sum_{i\in V\backslash\{n+1\}} z_{ijk}\ ,\quad j\in V\backslash\{0,n+1\}, k\in R_i\cap R_j, \tag{14}$$

$$s_{jt}\in\{0,1\}\ ,\quad j\in V, t\in T, \tag{15}$$

$$x_{jklt}\in\{0,1\}\ ,\quad j\in V, k\in R, l\in L, t\in T, \tag{16}$$

$$z_{ijk}\in\{0,1\}\ ,\quad i\in V\backslash\{0\}, j\backslash\{S_j^I\}, k\in R_i\cap R_j. \tag{17}$$

Equation (1) is to minimize the makespan of the information collection mission. Equation (2) aims to minimize the cost. Equation (3) is to maximize the mission's quality, and to make the optimization direction consistent, this article minimizes the reciprocal of the mission's quality. Constraints (4) show the way to calculate the processing time of an information collection task when given a specific resource allocation result. Constraints (5) ensure that the denominator of Eq. (3) is not equal to zero. The constraints (6) limit the maximum number of agents allocated to a given task. Constraints (7) restrict that the task in information collection mission can be only started to be processed only once. Constraints (8) make sure that once the agent is allocated to a task, it can only perform one type of skill in that task. Constraints (9) make sure that the transfer time and precedence relations must be respected in resource allocation. Constraints (10)–(11) restrict that the agent can only perform one task at the same time. Constraints (12) ensure that if an agent is to be assigned to another task, the current position of that agent

should be the direct or indirect predecessor of the corresponding task. Constraints (13) make sure that each task's processing is less than its maximum processing according to the resource allocation result. Constraints (14) show the relationship between decision variables z and x, which avoid that the agent is assigned from a predecessor task to a successor task. Constraints (15)–(17) define the domain for each decision variable.

Disruption recovery model

Disruptions can be caused by the breakdown of agents, the incorrect estimations of environment parameters, and the dangerous situation discovered during the mission. The disruptions can cause the deterioration of the optimization objectives. Two types of disruptions recovery conditions are considered. The symbols are defined in Table 2.

Preempt-repeat condition

In the preempt-repeat condition, the affected tasks should be processed from their very beginning. The reactive scheduling will be executed just after the disruption occurs.

Precedence relations: The precedence relation is the same as Eq. (9).

Start time constraints: Assume the rescheduling is started immediately after the disruption. The incomplete or affected tasks must be finished after the disruption

$$\sum_{t=Dt}^{LS_j} s_{jt} = 1\,, \quad j \in V^*. \tag{18}$$

Skill requirement: The size of the task area and the agent's availability may change after the disruption. The skill requirement must be satisfied in the repaired schedule

$$\sum_{t=Dt}^{LS_j} \sum_{k \in R^*} x_{jklt} \cdot u_{kl} \geq \frac{A_j^{*l}}{p_j^{\max} - \Gamma_j}, \quad j \in V^*, l \in L. \tag{19}$$

Preempt-resume condition

In preempt-resume conditions, the affected task starts from the portion of work it left before the disruption.

Precedence relations: The precedence relation is the same as Eq. (9).

Start time constraints: For the task whose start time is earlier than the disruption, it follows constraints (20). For the task whose start time and finish time are later than the disruption, it should be started twice, as shown in constraints (21). s'_j is the actual start time of task j

$$\sum_{t=Dt}^{LS_j} s_{jt} = 1\,, \quad j \in V^*, s'_j > Dt \tag{20}$$

$$\sum_{t=Dt}^{LS_j} s_{jt} = 1\,, \quad j \in V^*, s'_j < Dt, F_j > Dt. \tag{21}$$

Skill requirement: Constraints are the same as Eq. (19) for the task whose start time is earlier than the disruption. For the task whose start time is earlier than the disruption and the finish time is later than the disruption, if the skill requirement increases, more agents should be allocated to the task as constraints (22)

$$\sum_{t=Dt}^{LS_j} \sum_{k \in R^*} x_{jklt} \cdot u_{kl} \geq \max\left\{0, \frac{A_j^{*l} - A_j^l}{p_j^{\max} - \Gamma_j}\right\}, \quad j \in V*, l \in L, s'_j < Dt, F_j > Dt. \tag{22}$$

Parallel pareto local search algorithm

The reactive scheduling of the information collection mission is time-critical (less than 60 s). Although the multi-objective meta-heuristic algorithm has stronger global search ability, given a high-quality initial population in advance and a time limit, it does not always obtain a better solution than the local search method. The approximated pareto front before disruption is the initial population in this article. A parallel pareto local search framework is designed to deal with the information collection mission reactive scheduling problem under two types of disruptions.

Solution representation

A task vector and a resource matrix are combined to represent the solutions. The task vector is denoted as $\boldsymbol{\pi} = \{\pi_1, \pi_2, \ldots, \pi_N\}$; each element in the task vector takes a priority value from 0 to 1, which indicates the priority of the corresponding task. Then, the way to decode a task vector to a task list that meets the precedence constraints is presented: all the elements in the task list are rearranged in ascending order, while the precedence constraints are met, and the position of each element represents the index of task, and then, a feasible task list is obtained. The resource matrix is denoted as $\boldsymbol{M}$, which decides the resource allocation result. $\boldsymbol{M}$ is a $K \times \sum_{j \in V} |L_j|$ matrix; each value in $\boldsymbol{M}$ takes value from 0 to 1. The index of row in $\boldsymbol{M}$ denotes the resource index, and the index of column in $\boldsymbol{M}$ denotes the types of skill required for each task corresponding to the task list. Assume task i's index in task list is ip, then the $\sum_{j \in \{\pi_1 \ldots \pi_{ip-1}\}} |L_j|$ column in matrix $\boldsymbol{M}$ represents the agents assigned to task-i to perform the first type of skill in $|L_j|$. Given an element in $\boldsymbol{M}$, if

Table 5 Performance comparison of IGD in preempt-repeat condition

	PPLS	EMOIS	MOIWO	MOTLA
Instance-1	337.289 (19.225)	**254.545 (17.818)**≀	392.454 (45.132)†	290.945 (33.75)≀
Instance-2	335.945 (31.579)	347.225 (19.445)≈	388.568 (41.577)†	**305.936 (26.31)**≀
Instance-3	**236.876 (17.055)**	253.771 (14.972)†	313.026 (36.624)†	244.106 (22.458)≈
Instance-4	614.829 (70.705)	**602.858 (39.789)**≈	640.366 (49.949)≈	624.096 (52.424)≈
Instance-5	**676.131 (66.261)**	704.918 (64.852)≈	818.136 (90.813)†	705.26 (54.305)≈
Instance-6	**728.668 (82.34)**	841.264 (72.349)†	841.156 (58.04)†	855.684 (48.774)†
Instance-7	**790.837 (61.685)**	999.644 (109.961)†	945.279 (72.786)†	1010.093 (54.545)†
Instance-8	**1442.284 (128.363)**	1737.017 (166.754)†	1797.055 (122.2)†	1838.744 (187.552)†
Instance-9	**1605.757 (176.633)**	1913.732 (130.134)†	2011.649 (120.699)†	1970.588 (100.5)†
Instance-10	**1615.725 (93.712)**	1878.023 (161.51)†	1850.601 (142.496)†	2084.211 (106.295)†
†/≀/≈	–	6/1/3	9/0/1	5/2/3

The bold numbers are represent the optimal value of each experiment

Table 6 Performance comparison of IGD in preempt-resume condition

	PPLS	EMOIS	MOIWO	MOTLA
Instance-1	296.889 (29.689)	**230.612 (20.294)**≀	355.844 (40.922)†	271.226 (27.394)≀
Instance-2	294.533 (21.206)	297.767 (21.141)≈	354.779 (21.287)†	**281.94 (26.502)**≈
Instance-3	368.881 (35.413)	374.959 (34.496)≈	472.723 (31.2)†	**347.678 (26.771)**≈
Instance-4	**656.057 (76.759)**	666.31 (44.643)≈	736.49 (58.919)†	688.686 (42.699)≈
Instance-5	1356.051 (115.264)	1291.477 (78.78)≈	1398.77 (116.098)≈	**1258.062 (124.548)**≀
Instance-6	**1216.13 (125.261)**	1376.378 (94.97)†	1507.744 (117.604)†	1462.239 (99.432)†
Instance-7	**1626.66 (190.319)**	1779.951 (177.995)†	1832.006 (122.744)†	1865.787 (106.35)†
Instance-8	**1432.015 (104.537)**	1683.063 (102.667)†	1853.527 (174.232)†	1760.33 (191.876)†
Instance-9	**1764.954 (125.312)**	2174.526 (213.104)†	2177.77 (132.844)†	2048.478 (219.187)†
Instance-10	**1665.306 (156.539)**	2004.089 (140.286)†	2067.712 (215.042)†	2146.815 (251.177)†
†/≀/≈	–	5/1/4	9/0/1	5/2/3

The bold numbers are represent the optimal value of each experiment

Table 7 Performance comparison of SP in preempt-repeat condition

	PPLS	EMOIS	MOIWO	MOTLA
Instance-1	7.270 (0.785)	7.377 (0.73)≈	8.019 (0.473)≀	**8.498 (0.833)**≀
Instance-2	7.366 (0.759)	7.811 (0.43)≀	8.141 (0.464)≀	**8.843 (0.964)**≀
Instance-3	7.981 (0.495)	7.273 (0.662)†	**8.12 (0.438)**≈	6.629 (0.703)†
Instance-4	9.850 (0.561)	9.107 (0.61)†	**11.56 (0.728)**≀	10.767 (0.775)≀
Instance-5	9.593 (0.835)	10.269 (0.657)≀	9.005 (1.009)†	**10.916 (0.72)**≀
Instance-6	15.97 (1.182)	**18.588 (1.487)**≀	15.869 (1.063)≈	13.202 (0.858)†
Instance-7	13.163 (1.369)	13.708 (0.836)≈	11.846 (0.746)†	**14.147 (1.075)**≀
Instance-8	**14.001 (1.442)**	13.809 (1.16)≈	13.026 (0.808)†	13.875 (0.971)≈
Instance-9	12.776 (0.779)	12.953 (1.282)≈	**13.422 (1.248)**≈	12.626 (1.503)≈
Instance-10	19.016 (1.864)	20.507 (1.128)≀	**21.545 (1.163)**≀	15.781 (1.294)†
†/≀/≈	–	2/4/4	3/4/3	3/5/2

The bold numbers are represent the optimal value of each experiment

Table 8 Performance comparison of SP in preempt-resume condition

	PPLS	EMOIS	MOIWO	MOTLA
Instance-1	6.893 (0.503)	**9.479 (0.891)**≀	5.735 (0.539)†	7.187 (0.446)≈
Instance-2	6.977 (0.733)	**8.059 (0.467)**≀	5.695 (0.513)†	7.478 (0.733)≀
Instance-3	10.261 (0.759)	11.148 (0.557)≀	9.851 (0.847)≈	**11.91 (1.167)**≀
Instance-4	10.347 (1.159)	11.478 (0.872)≀	11.248 (0.686)≀	**11.834 (1.325)**≀
Instance-5	8.143 (0.708)	7.998 (0.904)≈	**8.713 (0.889)**≀	7.432 (0.81)†
Instance-6	12.797 (1.472)	**13.581 (1.589)**≀	11.711 (0.843)†	12.353 (1.136)≈
Instance-7	13.346 (0.867)	11.29 (0.948)†	**15.218 (0.791)**≀	13.778 (1.571)≈
Instance-8	15.856 (1.823)	12.449 (1.108)†	**20.719 (2.445)**≀	13.278 (1.407)†
Instance-9	13.683 (0.93)	13.523 (1.339)≈	14.317 (0.802)≈	**15.479 (1.099)**≀
Instance-10	16.81 (1.328)	17.773 (1.102)≈	13.782 (1.502)†	**17.922 (0.914)**≀
†/≀/≈	–	2/5/3	4/4/2	2/5/3

The bold numbers are represent the optimal value of each experiment

Table 9 Performance comparison of HV in preempt-repeat condition

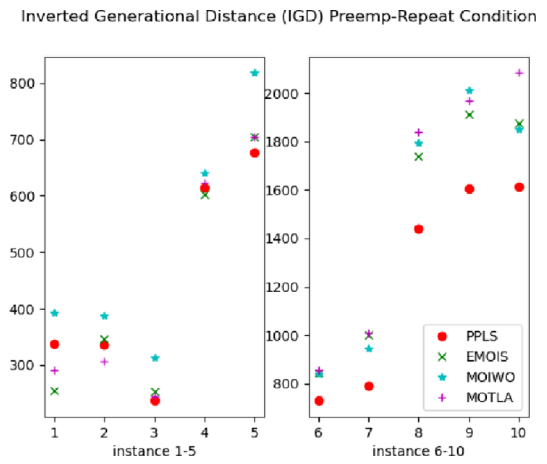
	PPLS	EMOIS	MOIWO	MOTLA
Instance-1	**8.36e+12 (9.78e+11)**	8.20e+12 (7.46e+11)≈	7.46e+12 (5.37e+11)†	5.90e+12 (5.55e+11)†
Instance-2	**7.66e+12 (8.81e+11)**	5.99e+12 (3.47e+11)†	7.38e+12 (4.14e+11)≈	5.92e+12 (4.14e+11)†
Instance-3	**8.52e+12 (7.84e+11)**	6.26e+12 (7.51e+11)†	8.16e+12 (8.16e+11)≈	4.96e+12 (3.87e+11)†
Instance-4	**1.65e+14 (1.72e+13)**	1.03e+14 (1.05e+13)†	1.46e+14 (1.27e+13)†	1.13e+14 (1.26e+13)†
Instance-5	4.41e+13 (3.00e+12)	1.93e+13 (9.83e+11)†	**4.71e+13 (2.59e+12)**≀	4.36e+13 (3.05e+12)≈
Instance-6	1.69e+14 (2.01e+13)	**1.90e+14 (1.65e+13)**≀	1.48e+14 (1.69e+13)†	1.34e+14 (1.50e+13)†
Instance-7	9.14e+14 (1.02e+14)	7.96e+14 (4.93e+13)†	4.39e+14 (2.37e+13)†	**9.23e+14 (6.83e+13)**≈
Instance-8	2.43e+14 (2.21e+13)	2.08e+14 (1.08e+13)†	1.93e+14 (9.64e+12)†	**2.98e+14 (2.92e+13)**≀
Instance-9	**5.60e+14 (3.14e+13)**	5.23e+14 (4.71e+13)†	3.65e+14 (3.03e+13)†	4.49e+14 (5.38e+13)†
Instance-10	8.33e+14 (6.41e+13)	8.25e+14 (8.00e+13)≈	6.76e+14 (5.68e+13)†	**8.82e+14 (4.59e+13)**≈
†/≀/≈	–	7/1/2	7/1/2	6/1/3

The bold numbers are represent the optimal value of each experiment

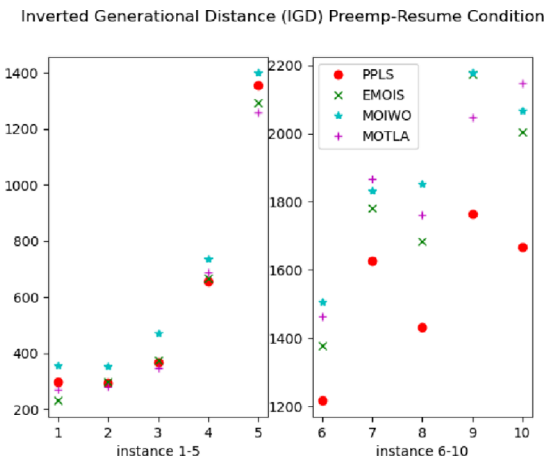
Table 10 Performance comparison of HV in preempt-resume condition

	PPLS	EMOIS	MOIWO	MOTLA
Instance-1	**9.04e+12 (7.50e+11)**	7.59e+12 (7.74e+11)†	8.86e+12 (1.01e+12)≈	7.37e+12 (8.40e+11)†
Instance-2	**9.12e+12 (5.84e+11)**	7.87e+12 (6.14e+11)†	8.84e+12 (9.28e+11)≈	7.51e+12 (5.86e+11)†
Instance-3	**1.51e+13 (1.65e+12)**	1.39e+13 (1.36e+12)†	1.41e+13 (7.45e+11)†	1.20e+13 (1.20e+12)†
Instance-4	1.85e+14 (1.83e+13)	**1.87e+14 (1.19e+13)**≈	1.38e+14 (1.31e+13)†	1.54e+14 (1.65e+13)†
Instance-5	**3.46e+13 (3.15e+12)**	3.00e+13 (3.42e+12)†	3.33e+13 (2.33e+12)≈	2.43e+13 (2.58e+12)†
Instance-6	1.65e+14 (1.15e+13)	**1.80e+14 (1.91e+13)**≀	1.59e+14 (1.37e+13)≈	1.67e+14 (1.59e+13)≈
Instance-7	**9.58e+14 (9.58e+13)**	8.49e+14 (5.60e+13)†	7.75e+14 (5.97e+13)†	8.26e+14 (8.67e+13)†
Instance-8	**1.78e+14 (1.80e+13)**	1.09e+14 (9.58e+12)†	1.11e+14 (1.19e+13)†	1.46e+14 (1.74e+13)†
Instance-9	3.29e+14 (3.25e+13)	**3.44e+14 (3.68e+13)**≈	2.08e+14 (1.97e+13)†	2.88e+14 (2.94e+13)†
Instance-10	**1.36e+15 (1.00e+14)**	1.28e+15 (6.65e+13)≈	1.12e+15 (5.60e+13)†	1.12e+15 (1.06e+14)†
†/≀/≈	–	6/1/3	6/0/4	9/0/1

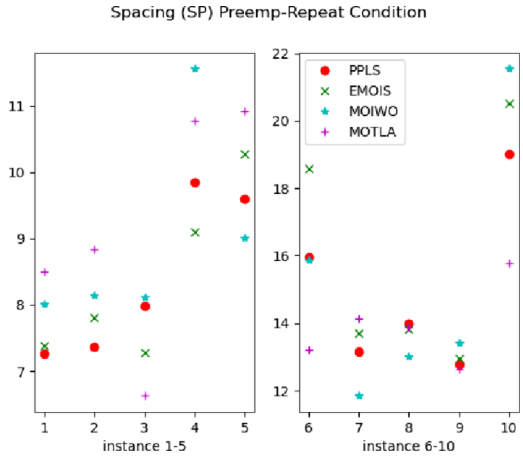
The bold numbers are represent the optimal value of each experiment



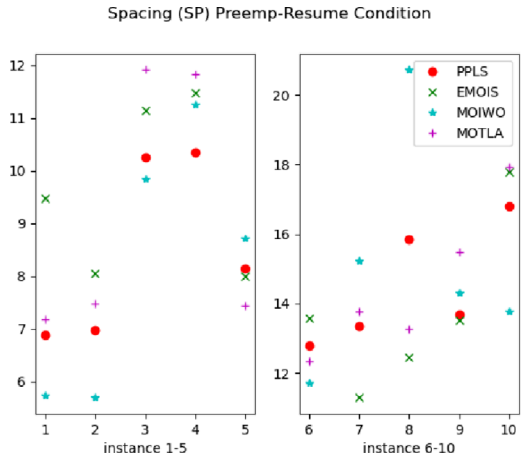
(a) IGD (preempt repeat condition)



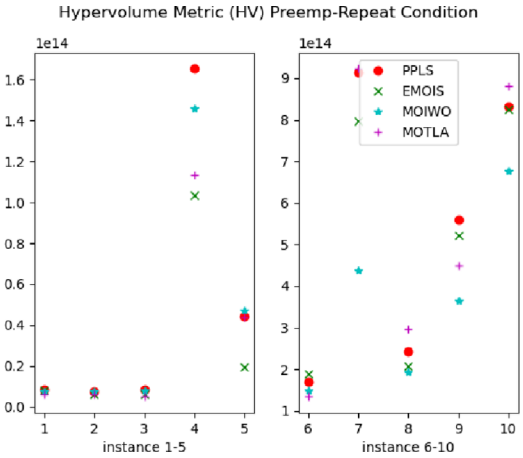
(b) IGD (preempt resume condition)



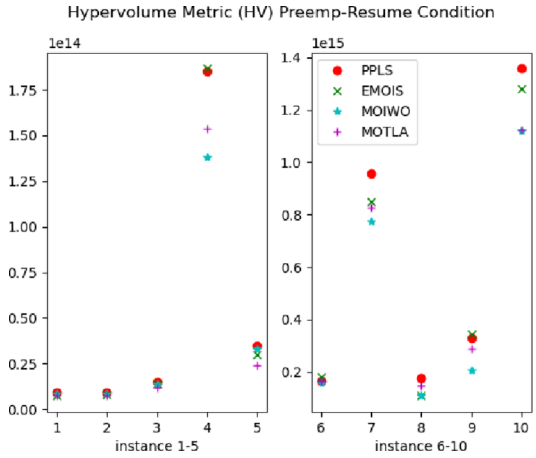
(c) SP (preempt repeat condition)



(d) SP (preempt resume condition)



(e) HV (preempt repeat condition)



(f) HV (preempt resume condition)

Fig. 4 Performance comparison

Table 11 The effectiveness of the acceptance criterion 2 with respect to SP

	Preempt-repeat condition		Preempt-resume condition	
	PPLS	PPLS-WN	PPLS	PPLS-WN
Instance-1	**7.270 (0.785)**	7.119 (0.736)	6.893 (0.503)	**7.366 (0.496)**
Instance-2	7.366 (0.759)	**7.541 (0.736)**	**6.977 (0.733)**	6.549 (0.780)
Instance-3	**7.981 (0.495)**	7.645 (0.477)	**10.261 (0.759)**	9.649 (0.806)
Instance-4	9.850 (0.561)	**10.296 (0.525)**	**10.347 (1.159)**	9.669 (1.239)
Instance-5	**9.593 (0.835)**	9.219 (0.869)	8.143 (0.708)	**8.395 (0.753)**
Instance-6	**15.970 (1.182)**	14.922 (1.222)	**12.797 (1.472)**	11.937 (1.455)
Instance-7	**13.163 (1.369)**	12.479 (1.452)	13.346 (0.867)	**14.085 (0.915)**
Instance-8	14.001 (1.442)	**14.472 (1.471)**	**15.856 (1.823)**	14.851 (1.932)
Instance-9	**12.776 (0.779)**	12.496 (0.796)	**13.683 (0.930)**	12.750 (0.929)
Instance-10	**19.016 (1.864)**	18.116 (1.741)	**16.810 (1.328)**	15.794 (1.318)

The bold numbers are represent the optimal value of each experiment

Table 12 The effectiveness of the acceptance criterion 2 with respect to HV

	Preempt-repeat condition		Preempt-resume condition	
	PPLS	PPLS-WN	PPLS	PPLS-WN
Instance-1	8.36e+12 (9.78e+11)	**8.86e+12 (9.76e+11)**	9.04e+12 (7.50e+11)	**9.54e+12 (7.75e+11)**
Instance-2	**7.66e+12 (8.81e+11)**	7.41e+12 (8.48e+11)	**9.12e+12 (5.84e+11)**	8.48e+12 (6.05e+11)
Instance-3	**8.52e+12 (7.84e+11)**	8.05e+12 (7.32e+11)	**1.51e+13 (1.65e+12)**	1.47e+13 (1.72e+12)
Instance-4	**1.65e+14 (1.72e+13)**	1.61e+14 (1.67e+13)	1.85e+14 (1.83e+13)	**1.94e+14 (1.79e+13)**
Instance-5	4.41e+13 (3.00e+12)	**4.55e+13 (3.01e+12)**	3.46e+13 (3.15e+12)	**3.65e+13 (3.23e+12)**
Instance-6	**1.69e+14 (2.01e+13)**	1.62e+14 (1.90e+13)	**1.65e+14 (1.15e+13)**	1.63e+14 (1.23e+13)
Instance-7	**9.14e+14 (1.02e+14)**	8.85e+14 (1.09e+14)	9.58e+14 (9.58e+13)	**1.02e+15 (9.51e+13)**
Instance-8	**2.43e+14 (2.21e+13)**	2.27e+14 (2.08e+13)	**1.78e+14 (1.80e+13)**	1.73e+14 (1.89e+13)
Instance-9	5.60e+14 (3.14e+13)	**5.81e+14 (3.24e+13)**	3.29e+14 (3.25e+13)	**3.38e+14 (3.40e+13)**
Instance-10	8.33e+14 (6.41e+13)	**8.91e+14 (6.47e+13)**	**1.36e+15 (1.00e+14)**	1.27e+15 (1.03e+14)

The bold numbers are represent the optimal value of each experiment

the value of that element is greater than 0.5, the agent corresponding to the row index is assigned to perform the skill and the tasks that are represented by the column index. Following the above procedure, a feasible resource allocation result could be obtained. To show the above decoding process more intuitively, an information collection mission is presented in Fig. 2 and its decoding process is shown in Fig. 3.

Parallel pareto local search

Decomposition method

The Tchebycheff scalar objective function is selected, because it can handle the case in which the shape of pareto front is not convex. W weight vectors $\lambda^1, \ldots, \lambda^W$ are defined, and each vector corresponds to a parallel process. The sum of the elements in each weight vector $\lambda^w = (\lambda_1^w, \ldots, \lambda_m^w)$ is equal to 1, and $\lambda_m^w \geq 0$, $w \in W$. Given a positive integer H, the way to calculate the value of each weight vector and the subregion definition for each parallel process w is the same as [39]. Define $z^* = (z_1, \ldots, z_m)$ as the Nadir point. Using the Tchebycheff approach, the objective function for process w is defined as

$$\max f^{te}(x|\lambda^w, z^*) = \min_{1\leq i\leq m}\left\{\frac{1}{\lambda_i^w}(f_i(x) - z_i^*)\right\}. \tag{23}$$

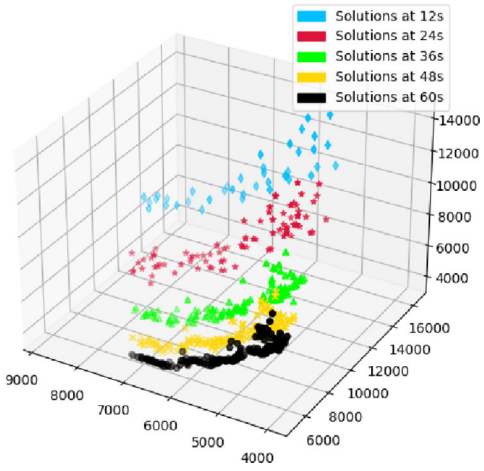
Framework of the proposed method

The framework of the proposed parallel pareto local search (PPLS) is shown in Algorithm 1. W processes are running in parallel. In each process, A_w is initialized to the solutions both in the pareto front before disruption (A_0) and the subregion λ^w. Mn is number of individuals each parallel process maintains.

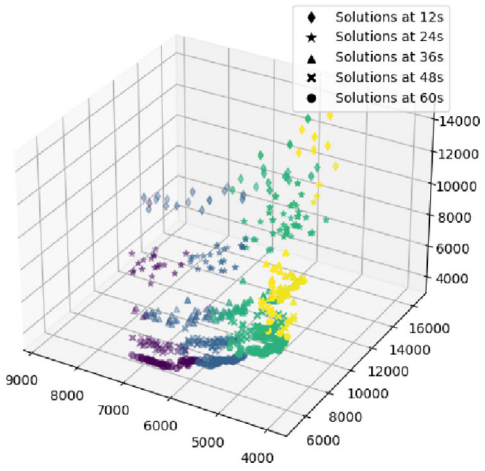
Individual local search

Given an existing solution x, the Gaussian mutation operator is adopted to conduct local search

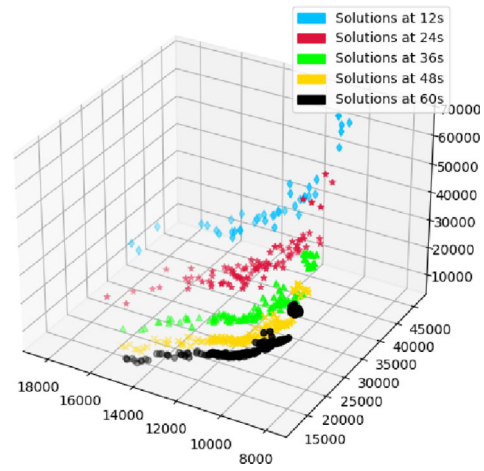
$$x_i' = x_i + N(0, \sigma_i), \tag{24}$$



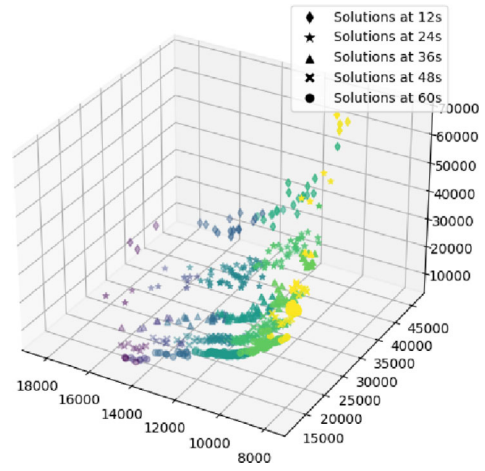
(a) Instance-3 (color deontes parallel process)



(b) Instance-3 (color deontes time)



(c) Instance-7 (color deontes parallel process)



(d) Instance-7 (color deontes time)

Fig. 5 The search process of PPLS

where x_i denotes the ith element of x and $N(0, \sigma_i)$ denotes a Gaussian random variable with zero mean and standard deviation σ_i. If $x' > 1$, set x' to 1. If $x' < 0$, set x' to 0. In the beginning, σ_i is initialized to the same value. The value of σ_i is adapted in each based on the $\frac{1}{5}$ successful rule proposed in [40]

$$\sigma_i = \begin{cases} \sigma_i/\beta & \text{if } nc/Mn > 0.2 \\ \sigma_i & \text{if } nc/Mn = 0.2 \\ \sigma_i \times \beta & \text{if } nc/Mn < 0.2, \end{cases} \tag{25}$$

where nc is the number of changed individual in a process, and β is a fixed parameter.

Acceptance criterion based on constraint handling

A problem-specific constraint handling method is proposed in this part. For preempt-repeat condition, given an individual x and using the decoding process designed in Sect. 3.1, the violation of constraints can be computed as

$$C(x) = \sum_{j \in J*} \sum_{l \in L} \left(\frac{A_j^{*l}}{p_j^{\max} - \Gamma_j} - \sum_{t=Dt}^{LS_j} \sum_{k \in R^*} x_{jklt} \cdot u_{kl} \right) \tag{26}$$

For preempt-resume condition

$$C(x) = \sum_{j \in J*} \sum_{l \in L} \left(\frac{A_j^{*l} - A_j^{l}}{p_j^{\max} - \Gamma_j} - \sum_{t=Dt}^{LS_j} \sum_{k \in R^*} x_{jklt} \cdot u_{kl} \right). \tag{27}$$

In each parallel process, three different conditions are considered:

(1) There is no feasible solution in the current population Pop. Considering the constraint violation $C(x)$ as the

Algorithm 1 parallel pareto local search (PPLS)

$PS = \{1, \ldots, W\}$;
$Mi = Mn$;
foreach $w \in PS$ **do**
  $A_w \leftarrow \{x \in A_0 | x \in subregion\ \lambda^w\}$;
end
while *Time Limit is not reached* **do**
  foreach $w \in PS$ **do**
    Do independently in parallel:
    $fw = 0$
    foreach $x \in A_w$ **do**
      if x *is a feasible solution* **then**
        $fw += 1$
      end
    end
    foreach $x \in A_w$ **do**
      Generate x' using individual local search (Eq. (24));
      AcceptFlag = False;
      if $|A_w| < Mn$ **then**
        Add x' to A_w;
        AcceptFalg = True
      end
      else
        Conduct Algorithm 2 to determine whether to accept x';
        Conduct Algorithm 3 to determine whether to accept x';
      end
    end
    if $fw == 0$ *and* $t > \frac{Time\ Limit}{4}$ **then**
      Remove w from PS
    end
    else
      Update σ_w;
    end
  end
  $Mn = \lfloor Mi \times W / |PS| \rfloor$;
end

only objective, optimize $C(x)$ until at least one feasible solution is obtained.

(2) There are both feasible and infeasible solutions in the current population Pop. We divide the current population into feasible group Pop_1 and infeasible group Pop_2. The objective functions are transformed to Eqs. (28)–(29) to penalize the infeasible solutions

If $e \in Pop_1$:

$$f_i'(x) = f_i(x). \tag{28}$$

If $x \in Pop_2$:

$$f_i'(x) = \max\left\{\varphi f_i^{\max} + (1-\varphi) f_i^{min},\ f_i(x)\right\}. \tag{29}$$

Then, the transformed objective functions and the constraint violations are normalized

$$\tilde{f}_i(x) = \frac{f_i'(x) - \min_{\bar{x} \in Pop} f_i'(\bar{x})}{\max_{\bar{x} \in Pop} f_i'(\bar{x}) - \min_{\bar{x} \in Pop} f_i'(\bar{x})}, i \in \{1, \ldots, m\} \tag{30}$$

$$\tilde{C}(x) = \begin{cases} 0, & \text{if } x \in Pop_1 \\ \frac{C(x) - \min_{\bar{x} \in Pop_2} C(\mathbf{x})}{\max_{\bar{x} \in Pop_2} C(\mathbf{x}) - \min_{\bar{x} \in Pop_2} C(\mathbf{x})}, & \text{if } x \in Pop_2. \end{cases} \tag{31}$$

F_i denotes the fitness value for i-th objective, we use F_i to replace f_i in Eq. (23), F_i can be calculated as

$$F_i(x) = \tilde{f}_i(x) + \tilde{C}(x), i \in \{1, \ldots, m\}. \tag{32}$$

(3) All the solutions in a population are feasible. The fitness value equals the corresponding objective value.

The detail of acceptance criterion 1 is presented in Algorithm 2.

Algorithm 2 Acceptance criterion 1

if *AcceptFalg* == *True* **then**
  break;
end
else if $F^{te}(x'|\lambda^w, z^*) > \min_{x \in A_w} F^{te}(x|\lambda^w, z^*)$ **then**
  Replace x with x';
  AccepFlag = True;
end
else
  Discard x';
end

Acceptance criterion based on negative correlation

For ith parallel process, the average of the solutions it maintains is denoted as $\tilde{x}_i$. The Bhattacharyya distance is used to present the negative correlation between different process. The Bhattacharyya distance between parallel process p_i and p_j is defined as

$$D_B(p_i, p_j) = \frac{1}{8}\left(\tilde{x}_i - \tilde{x}_j\right)^T \mathbf{\Sigma}^{-1}\left(\tilde{x}_i - \tilde{x}_j\right) + \frac{1}{2}\ln\left(\frac{\det \mathbf{\Sigma}}{\sqrt{\det \mathbf{\Sigma}_i \det \mathbf{\Sigma}_j}}\right), \tag{33}$$

where $\mathbf{\Sigma_i}$ is $\sigma_i^2 \boldsymbol{I}$, $\boldsymbol{I}$ is the identity matrix, and $\mathbf{\Sigma} = (\mathbf{\Sigma}_i + \mathbf{\Sigma}_j)/2$. The main idea behind the criterion based on negative correlation is that, for a parallel process, the selected solution should be with high quality and should lead to a distribution that is distant from the other parallel processes. The former can be represented by Eq. (32). The latter can be represented as

$$\text{Corr}(p_i) = \min_j \left\{D_B(p_i, p_j) \mid j \neq i\right\}. \tag{34}$$

Then, normalization is conducted by requiring $\text{Corr}(x) + \text{Corr}(x') = 1$. Using the above definitions, the detail of acceptance criterion 2 is presented in Algorithm 3.

Algorithm 3 Acceptance criterion 2

```
if AcceptFalg == True then
  | break;
end
else if F(x')/Corr(p'_w) < ϵ then
  | Replace x with x';
  | AccepFlag = True;
end
else
  | Discard x';
end
```

Experiment

A computer with Intel E5 3.6 GHz CPU and 32 GB of RAM is used to conduct the experiment in this article. The algorithms discussed in this part are coded in C++. Three effective multi-objective algorithms for RCPSP are selected as the comparison algorithms.

- EMOIS [41], which uses an improved NSGA-II to solve multi-objective multi-skill RCPSP.
- MOIWO [42], in which a modified multi-objective invasive weeds optimization algorithm is proposed to solve multi-mode multi-skill RCPSP.
- MOTLA [43], which uses teaching–learning-based optimization algorithm to solve multi-objective multi-skill RCPSP.

Performance metric

Five metrics are considered. Since the true pareto front is unknown, the P^* is obtained by merging all the approximation pareto fronts found by all the algorithms. The metrics that describe the quality of the pareto front:

Inverted Generational Distance (IGD) [44]: The value of IGD shows the convergence and the diversity of the obtained pareto front at the same time. IGD can be calculated as

$$\mathrm{IGD} = \frac{\sum_{i \in P} D\,(i,\, P^*)}{|P^*|}, \tag{35}$$

where $D\,(i,\, P^*)$ is the minimum Euclidean distance between i and the point in P^*. Smaller IGD is better.

Spacing (SP) [44]: SP shows the diversity of the obtained pareto front, which is defined as

$$SP = \sqrt{\frac{1}{(n'-1)\bar{d}} \sum_{i=1}^{n'} \left(d_i - \bar{d}\right)^2}, \tag{36}$$

where n' denotes the number of non-dominated solutions, and d_i is the Euclidean distance between the two nearest non-dominated solutions of the obtained pareto front and the true pareto front, the average of which is denoted as $\bar{d}$. The pareto front with a larger SP value shows better performance on solutions diversity.

Hypervolume Metric (HV)[44]: The value of HV shows the convergence and the diversity of the obtained pareto front simultaneously. HV represents the objective space volume which is dominated by the approximation pareto front P and delimited from above by a reference point $r \in \mathbb{R}^n$. HV is defined as

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

where x_i denotes a point in the pareto front P, and λ_m represents the m-dimensional Lebesgue measure. (max f_1, max f_2, max f_3) is adopted as the reference point. The pareto front with a larger HV value has better performance.

Test instance

The MS-RCPSP test instance generator proposed in [45] is modified to generate new instances for our problem. Some parameters which can be used to describe the difference between the test instances are presented: (1) $nAct$: the number of non-dummy information collection tasks in a mission. (2) K: the number of resource(agent) which can be used in the information collection mission. (3) $|L|$: the number of skill types. (4) NC: the average number of successors of each task, which is used to represent the network complexity. (5) MD, the number of tasks that are influenced by the disruption. (6) Skill factor SF: skill factor, which represents the relationship between the number of skill types required to in an information collection task. (7) RSS_l: resource strength, which shows the relative relationship between the task's skill requirement and the corresponding resource's supply. Ten test instances are generated, and detailed information of the test instance is shown in Table 3. Each instance is tested under preempt-repeat and preempt-resume conditions.

Parameters' tuning

Taguchi method is applied to tune the parameters as [44]. Three levels of each parameter are considered (Table 4), and the best value is in bold. The time limit for all algorithms is 60 s.

Performance evaluation

All rendered results are obtained by performing ten independent runs of each algorithm. The time limit is set to 60 s. The average value and standard deviation are calculated and

are shown in Tables 5, 6, 7, 8, 9, and 10 and Fig. 4. The Wilcoxon's rank sum test at 5% significance level is used to present the difference between the comparison algorithm and the algorithm designed in this article. Using T' to denote the rank sum of the comparison algorithm result. According to the rank sum table, $P(82 < T' < 128) = 0.05$, that is, $T' > 128$ means that the comparison algorithm is worse than the proposed algorithm, $T' < 82$ represents the comparison algorithm is better than the proposed algorithm, and $82 < T' < 128$ means the comparison algorithm is similar to the proposed algorithm. The symbols †, ≀, and ≈ denote that the performance of the proposed algorithm in this article is better, worse, and similar than the comparison algorithm. The standard deviation is shown in parentheses, and the data in bold are the best value found in the corresponding test instance.

First, the performance on IGD is discussed. The PPLS outperforms the other algorithms significantly. The PPLS obtains 7 best solutions in preempt-repeat condition and 6 in preempt-resume condition. It is clear that PPLS obtains the best pareto front. MOTLA has the second-best performance generally.

Second, SP of each algorithm is compared. PPLS obtains only 1 best solution in preempt-repeat condition and 0 in preempt-resume condition. The algorithm with higher quality pareto front usually does not has competitive performance on SP, so the performance of PPLS is acceptable. Besides, none of the comparison algorithms has a clear advantage over the others.

Third, with respect to HV, PPLS obtains 5 best solutions in preempt-repeat condition and 7 in preempt-resume condition. MOTLA finds the 3 best solutions in preempt-repeat condition, which is the closest to PPLS. EMOIS finds 3 best solutions in preempt-resume conditions. MOIWO has the worst performance and PPLS has the best performance.

Fourth, the experiment is conducted to show the effectiveness of the proposed acceptance criterion based on negative correlation. PPLS-WN represents the proposed PPLS without the acceptance criterion based on negative correlation. HV and SP are calculated under preempt-repeat and preempt-resume condition, the results are presented in Tables 11 and 12. The results clearly show that the acceptance criterion based on negative correlation can increase the diversity of the obtained pareto front without reducing its quality.

Furthermore, to show the PPLS's search process better, 5 moments are presented on instance-3 and instance-7, as shown in Fig. 5.

Summarizing, the proposed method outperforms the comparison algorithms generally. The reasons why the proposed method outperforms other algorithms can be concluded as follows:

- The framework of PPLS is designed specifically to fit the information collection mission reactive scheduling problem's characteristics.
- The proper design of the solution representation and decoding scheme contributes to reducing the search space.
- For the information collection mission reactive scheduling problem within a short time limit, the algorithm with stronger local search ability tends to have better performance, and the importance of global search ability is relatively weak.

Conclusion

This article focuses on providing decision support for the decision-makers when disruption prevents the disaster information collection mission from completing the work as planned. When disruption occurs, it is vary difficult for the decision-makers to make high-quality reactive decisions. The disaster information collection resource scheduling problem is modeled as an extension of resource-constrained project scheduling problem (RCPSP). The mathematical model of the disaster reactive decision support problem with two recovery models is given. A novel framework of a parallel pareto local search based on decomposition is specially designed to provide reactive decisions for the decision-makers within the time limit. Two solution acceptance criteria based on constraint handling and negative correlation are also proposed to maintain high-quality population with diversity. The experiments have been conducted and the results show the proposed method outperforms the other competitors. As a part of our future research plan, we aim to develop new algorithms for solving RCPSPs involving complex practical issues, such as the prediction of dynamic disruptions and resource uncertainty. Although the proposed algorithm is efficient, it is not a real-time algorithm. The user experience of real-time decision support systems is significantly better than that of non-real-time systems. Using the deep reinforcement learning method to train a policy network which represents the reactive policy when disruption happens is a possible way to realize real-time decision support. Although the training process might be time-consuming, it can be viewed as a preparation before the use of the decision support system. Once the policy network is obtained, the time cost of the reactive scheduling process will be in milliseconds. Game theory can also be introduced in the deep reinforcement learning methods to explore the interesting interaction between decision-making and the environment's feedback.

Funding This article was funded by Key Programme (Grant no. U2013602).

References

1. Ahmadi M, Seifi A, Tootooni B (2015) A humanitarian logistics model for disaster relief operation considering network failure and standard relief time: a case study on san francisco district. Transport Res Part E Logist Transport Rev 75:145–163
2. Wei X, Qiu H, Wang D, Duan J, Wang Y, Cheng T (2020) An integrated location-routing problem with post-disaster relief distribution. Comput Ind Eng 147:106632
3. Feng Y, Liu T, Hu Z, Wang D, Cheng T, Yin Y (2021) Casualty transport scheduling considering survival probability and injury classification. Comput Ind Eng 161:107655
4. Jiao L, Peng Z, Xi L, Guo M, Ding S, Wei Y (2023) A multistage heuristic algorithm based on task grouping for vehicle routing problem with energy constraint in disasters. Expert Syst Appl 212:118740
5. Sun Y, Chen X, Jun L, Zhao J, Hu Q, Fang X, Yan Y (2021) Ship trajectory cleansing and prediction with historical ais data using an ensemble ann framework. Int J Innov Comput Inf Control 17:443–459
6. Arena S, Florian E, Zennaro I, Orrù P, Sgarbossa F (2022) A novel decision support system for managing predictive maintenance strategies based on machine learning approaches. Saf Sci 146:105529
7. Feng Y, Yin Y, Wang D, Dhamotharan L (2022) A dynamic ensemble selection method for bank telemarketing sales prediction. J Bus Res 139:368–382
8. Wang D, Zhu J, Wei X, Cheng T, Yin Y, Wang Y (2019) Integrated production and multiple trips vehicle routing with time windows and uncertain travel times. Comput Oper Res 103:1–12
9. Mufana MW, Ibrahim A (2022) Implementation of smart grid decision support systems. IDOSR J Sci Res 7(1):50–57
10. Xiao J, Wu Z, Hong X-X, Tang J-C, Tang Y (2016) Integration of electromagnetism with multi-objective evolutionary algorithms for rcpsp. Eur J Oper Res 251(1):22–35
11. Chaleshtarti AS, Shadrokh S, Khakifirooz M, Fathi M, Pardalos PM (2020) A hybrid genetic and lagrangian relaxation algorithm for resource-constrained project scheduling under nonrenewable resources. Appl Soft Comput 94:106482
12. Davari M, Demeulemeester E (2019) A novel branch-and-bound algorithm for the chance-constrained resource-constrained project scheduling problem. Int J Prod Res 57(4):1265–1282
13. Fouilhoux P, Mahjoub AR, Quilliot A, Toussaint H (2018) Branch-and-cut-and-price algorithms for the preemptive rcpsp. RAIRO-Oper Res 52(2):513–528
14. Chu Z, Xu Z, Li H (2019) New heuristics for the rcpsp with multiple overlapping modes. Comput Ind Eng 131:146–156
15. Kadrou Y, Najid NM (2006) A new heuristic to solve rcpsp with multiple execution modes and multi-skilled labor. In: Proceedings of the multiconference on "computational engineering in systems applications", vol 2, pp 1302–1309 . IEEE
16. Bhaskar T, Pal MN, Pal AK (2011) A heuristic method for rcpsp with fuzzy activity times. Eur J Oper Res 208(1):57–66
17. Lambrechts O, Demeulemeester E, Herroelen W (2008) Proactive and reactive strategies for resource-constrained project scheduling with uncertain resource availabilities. J Sched 11(2):121–136
18. Chen L, Zhang Z (2014) A two-stage resource-constrained project scheduling model with proactive and reactive strategies under uncertainty. In: Proceedings of the eighth international conference on management science and engineering management. Springer, pp 1397–1407
19. Davari M, Demeulemeester E (2017) The proactive and reactive resource-constrained project scheduling problem: the crucial role of buffer-based reactions. KU Leuven, Faculty of Economics and Business, KBI_1707
20. Chakrabortty RK, Abbasi A, Ryan MJ (2018) Robust project scheduling with unreliable resources: a variable neighbourhood search based heuristic approach. In: 2018 IEEE international conference on industrial engineering and engineering management (IEEM), pp 656–660. IEEE
21. Deblaere F, Demeulemeester E, Herroelen W (2011) Reactive scheduling in the multi-mode rcpsp. Comput Oper Res 38(1):63–74
22. Ning M, He Z, Wang N, Liu R (2018) Metaheuristic algorithms for proactive and reactive project scheduling to minimize contractor's cash flow gap under random activity duration. IEEE Access 6:30547–30558
23. Adamu PI, Akinwumi II, Okagbue HI (2019) Reactive project scheduling: minimizing delays in the completion times of projects. Asian J Civ Eng 20(8):1189–1202
24. Davari M, Demeulemeester E (2019) Important classes of reactions for the proactive and reactive resource-constrained project scheduling problem. Ann Oper Res 274(1–2):187–210
25. Davari M, Demeulemeester E (2019) The proactive and reactive resource-constrained project scheduling problem. J Sched 22(2):211–237
26. Wang W, Su J, Xu J, Ge X (2020) Reactive strategies in the multiproject scheduling with multifactor disruptions. Math Probl Eng 20:20
27. Zheng W, He Z, Wang N, Jia T (2018) Proactive and reactive resource-constrained max-npv project scheduling with random activity duration. J Oper Res Soc 69:1115126
28. Bagherinejad J, Jolai F, Abdollahnejad R, Shoeib M (2020) A hybrid algorithm based on non-dominated sorting ant colony and genetic algorithms for solving multi-objective multi-mode project scheduling problems under resource constraints. Manage Prod Eng Rev 11:25
29. Yeganeh FT, Zegordi SH (2020) A multi-objective optimization approach to project scheduling with resiliency criteria under uncertain activity duration. Ann Oper Res 285(1):161–196
30. Li Y-Y, Lin J, Wang Z-J (2022) Multi-skill resource constrained project scheduling using a multi-objective discrete jaya algorithm. Appl Intell 52:557185738
31. Zhu L, Lin J, Li Y-Y, Wang Z-J (2021) A decomposition-based multi-objective genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem. Knowl-Based Syst 225:107099
32. Hosseinian AH, Baradaran V (2021) A multi-objective multi-agent optimization algorithm for the multi-skill resource-constrained project scheduling problem with transfer times. RAIRO-Oper Res 55(4):2093–2128

33. Chand S, Singh H, Ray T (2019) Evolving heuristics for the resource constrained project scheduling problem with dynamic resource disruptions. Swarm Evol Comput 44:897–912
34. Chakrabortty RK, Rahman HF, Haque KM, Paul SK, Ryan MJ (2020) An event-based reactive scheduling approach for the resource constrained project scheduling problem with unreliable resources. Comput Ind Eng 20:106981
35. Cai J, Peng Z, Liao S, Ding S (2022) A multi-mode multi-skill project scheduling reformulation for reconnaissance mission planning. Sci China Inf Sci 65(6):169201
36. Mejia OP, Anselmet M-C, Artigues C, Lopez P (2017) A new rcpsp variant for scheduling research activities in a nuclear laboratory. In: 47th international conference on computers and industrial engineering (CIE47), p 8
37. Ortiz-Pimiento NR, Diaz-Serna FJ (2020) An optimization model to solve the resource constrained project scheduling problem rcpsp in new product development projects. Dyna 87(212):179–188
38. Almeida BF, Correia I, Saldanha-da-Gama F (2019) Modeling frameworks for the multi-skill resource-constrained project scheduling problem: a theoretical and empirical comparison. Int Trans Oper Res 26(3):946–967
39. Shi J, Zhang Q, Sun J (2020) PPLS/D: parallel pareto local search based on decomposition. IEEE Trans Cybern 20:50–3
40. Beyer H-G, Schwefel H-P (2002) Evolution strategies—a comprehensive introduction. Nat Comput 1(1):3–52
41. Laszczyk M, Myszkowski PB (2019) Improved selection in evolutionary multi-objective optimization of multi-skill resource-constrained project scheduling problem. Inf Sci 481:412–431
42. Maghsoudlou H, Afshar-Nadjafi B, Niaki STA (2016) A multi-objective invasive weeds optimization algorithm for solving multi-skill multi-mode resource constrained project scheduling problem. Comput Chem Eng 88:157–169
43. Zabihi S, Kahag MR, Maghsoudlou H, Afshar-Nadjafi B (2019) Multi-objective teaching-learning-based meta-heuristic algorithms to solve multi-skilled project scheduling problem. Comput Ind Eng 136:195–211
44. Ding S, Chen C, Xin B, Pardalos PM (2018) A bi-objective load balancing model in a distributed simulation system using NSGA-II and MOPSO approaches. Appl Soft Comput 63:249–267
45. 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