Information Sciences 636 (2023) 118924

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Frequent pattern-based parallel search approach for time-dependent agile earth observation satellite scheduling

Jian Wu [a,1], Feng Yao [a,1], Yanjie Song [a], Lei He [a], Fang Lu [c], Yonghao Du [a], Jungang Yan [a], Yuning Chen [a,*], Lining Xing [b,*], Junwei Ou [a]

[a] College of Systems Engineering, National University of Defense Technology, Changsha 410073, China
[b] School of Electronic Engineering, Xidian University, Xian 710000, China
[c] Business School, Xiangtan University, Xiangtan 411100, China

ARTICLE INFO

Keywords:
Time-dependent
Agile earth observation satellite scheduling
Parallel local search
Algorithm and operator adaptive selection
Frequent pattern mining

ABSTRACT

To address the time-dependent agile earth observation satellite (AEOS) scheduling problem more effectively, the frequent pattern-based parallel search (FPBPS) algorithm is proposed, which is composed of a parallel local search procedure, a competition-based algorithm and operator adaptive selection procedure and the new solutions construction method based on frequent pattern. First, the algorithm and the operator are selected from the algorithm pool and operator pool and run in parallel. As a result, some high-quality and diverse solutions are obtained in a short time. Second, we update the probability of each algorithm and operator to strengthen the self-adaptive ability of the algorithm. Third, frequent pattern mining method is used to extract knowledge with respect to AEOS scheduling to construct new solutions, and parallel local search is applied to further improve these solutions. Finally, extensive experiments prove that the FPBPS algorithm has a better performance than other comparison algorithms in the quality of solution, computation time, and robustness.

1. Introduction

The earth observation satellite (EOS) has the advantages of wide coverage, long imaging time, and no border restrictions, which play an increasingly important role in economic development, disaster relief and emergency monitoring [1]. With the continuous improvement of the satellite hardware level, the number of requirements specified by users also increases. To meet diversified and large-scale observation requirements, efficient satellite task scheduling technology is particularly crucial [2].

In Fig. 1, the EOS only has one degree of rolling, which leads the execution duration of tasks is equal to the length of visible time window (VTW). The satellite can only observe the target in the VTW. Different from EOS, the AEOS has three degrees, which are rolling, pitching, and yawing. As a result, the execution duration of tasks is shorter than the length of the VTW, the start time of tasks need to be determined, which increases the difficulty in solving the AEOS scheduling problem [23].

It has been proved that the AEOS task scheduling problem is NP-hard [4]. The rule is a common method for solving this problem in the industrial circle, which has low computational complexity and strong interpretability [5–6]. The rule includes the length of VTW, imaging quantity, remaining observation opportunities and other properties [7–10]. Generally, the quality of solutions generated by

* Corresponding authors.
E-mail addresses: 451480978@qq.com (Y. Chen), xing2999@qq.com (L. Xing).
[1] These authors contributed equally to this work and should be considered co-first authors.

https://doi.org/10.1016/j.ins.2023.04.003
Received 26 July 2022; Received in revised form 24 March 2023; Accepted 2 April 2023
Available online 6 April 2023

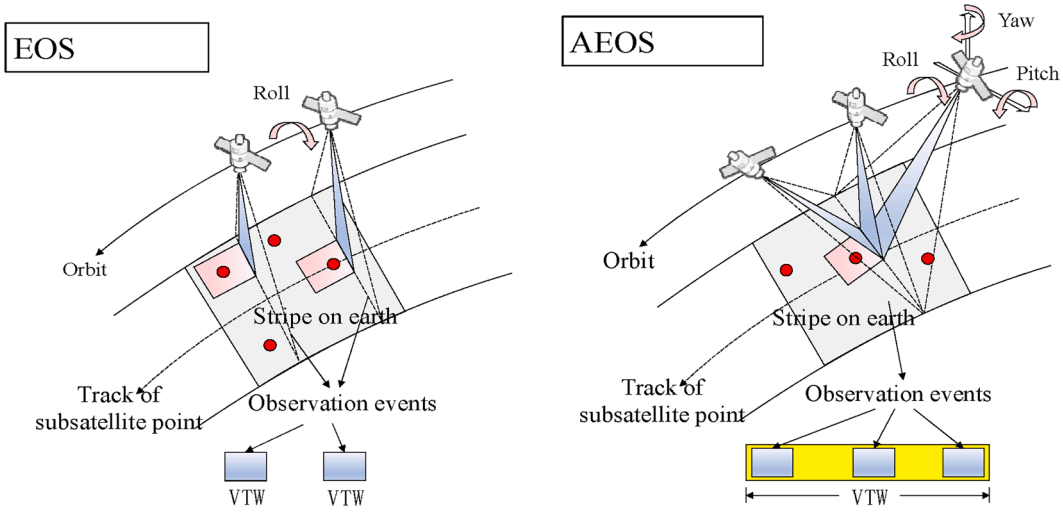

Fig. 1. Observation mode of EOS and AEOS [3].

rule is not high.

In the academic circle, there are three main methods to solve the AEOS task scheduling problem, which are exact algorithm, *meta*-heuristic algorithm and reinforcement learning (RL) [22]. The difficulty of exact algorithm is how to build linear programming model for problem. the column generation [11], dynamic programming [12] and some other exact algorithms [13] are successfully used to solve it. In theory, it can obtain the optimal solution in small-scale instances. However, as the scale increases, it will require considerable time and cannot even solve the problem, which is unacceptable in the project. the core idea of *meta*-heuristic algorithm is to find a better solution through neighbourhood operators or evolutionary operators in the process of search. Liu et al. designed the new neighbourhoods combining some problem-specific destroy operators and repair operators to solve the AEOS scheduling problem [14]. On this basis, He et al. added some assignment operators to solve the multi-AEOS collaborative scheduling problem [15]. Du et al. used the historical scheduling plan to reduce the search space of algorithm, and sped up the convergence of the algorithm [3]. A greedy randomized iterated local search algorithm was proposed by Peng et al., meanwhile, he designed a quick method for checking the insertion positions [16]. In addition, the AEOS task scheduling under uncertain environment is also a focus of research [1718]. The objective of the above algorithms is to maximize observation profit. The load balance degree is usually optimized as another objective [19–21]. The last is the RL, which has been applied successfully to solve classical combinatorial optimization problems [23–25]. Therefore, some scholars tried to solve the AEOS task scheduling problem with RL. To address the satellites resource allocation, an actor–critic algorithm was designed by Usaha et al. [26]. On this basis, Haijiao et al. used the same method to solve the online satellite task scheduling problem [27]. He et al. built the markov decision process model for the task execution process, and used the Q-network to solve it [1]. However, these works simplified the problem, and did not consider the complex constraints.

The *meta*-heuristic algorithm is selected in this paper. However, the low intelligence leads to a blind search and cannot guarantee the quality of final solution. Recently, more and more scholars use artificial intelligence (AI) to improve the efficiency of algorithm [28–30]. These improvements are mainly in the aspects of initial solution construction [31], neighbourhood selection [32], operator design [33] and estimation of distribution algorithms [34]. To overcome the shortcoming of meta -heuristic in solving the AEOS task scheduling problem, a frequent pattern-based parallel search algorithm (FPBPS) is proposed, which is composed of a parallel local search procedure, competition-based algorithm and operator adaptive selection (Competition) procedure and the new solutions construction method based on frequent pattern. First, we adopt the parallel local search to obtain diversified solutions with high-quality in a short time. Then, the competition procedure is proposed to update the probability of each algorithm and operator in FPBPS. Finally, to make full use of the high-quality solutions, we use a frequent pattern mining technique to extract problem-specific knowledge and construct new solutions to guide the search of algorithm.

The paper has two contributions. One is that the paper proposes a new algorithm framework, which provides a new idea to combine data mining with *meta*-heuristic algorithm. Meanwhile, the FPBPS can be modified to solve other time/order-dependent problems. The other is that the paper defines the knowledge of the AEOS task scheduling problem. Extensive research tried to improve the quality of solution from the perspective of algorithm capabilities, and ignored the feature the problem. This paper extracts the problem-specific knowledge and designs the effective algorithm based on the knowledge to solve this problem.

In structure, this paper consists of five parts. In part 2, we describe the problem and related assumptions in detail, meanwhile, the mathematical model is built. In part 3, we describe the principle the FPBPS algorithm and introduce each component in detail. In part 4, we design fifteen simulation scenarios and analyse the experimental results. In part 5, we give some conclusions and further work.

2. Mathematical model

In the AEOS task scheduling, there are multiple *meta*-tasks. The purpose is to generate a task execution plan, which cannot violate the constraints including maximum memory and maximum electricity and transition time. Meanwhile, the start time of task needs to be determined. The objective of problem is to maximize the profit of scheduled tasks. The following assumptions are first are given firstly.

1) Once a task is executed, it cannot be abandoned.
2) Satellites cannot execute multiple tasks simultaneously.
3) The task is *meta*-task in this paper, which can be executed at one time.

The parameters are divided into two parts: task properties and satellite properties.

1) Task properties.

$T = \{t_i | 1 \le i \le N\}$ denotes the task set;
N: the number of tasks;
p_i: the priority of task t_i;
l_i: The execution duration of task t_i;
m_i: The memory consumed by t_i every second;
e_i: The electricity consumed by t_i every second;

2) Satellite properties.

E: The maximum electricity of the satellite;
M: The maximum memory of the satellite;
$\gamma^t, \pi^t, \varphi^t$: The rolling angles, pitching angles and yawing angles at time t, respectively;
$w = \{w_{ij} | 1 \le i \le N, 1 \le j \le |w_i|\}$ denotes the VTW set;
$|w_i|$: The number of VTW for task t_i;
w_{ij}: The jth time window of task t_i;
st_{ij}, et_{ij}, d_{ij}: The start time, end time and length of w_{ij}, respectively.
e^f : The fixed electricity consumed in each attitude transition;
e^s: The electricity consumed per angle of attitude transition;

The AEOS task scheduling problem has two decision variables, one is the binary variable x_{ij}, it can be expressed by Formula (1). Another is the start time of task u_{ij}.

$$x_{ij} = \begin{cases} 1 if t_i is successfully scheduled in w_{ij} \\ 0 otherwise \end{cases} \quad \forall t_i \in T, w_{ij} \in w_i \tag{1}$$

The model of AEOS task scheduling problem is built as follows.

$$\textbf{Maximize} \sum_{i=1}^{N} \sum_{j=1}^{|w_i|} x_{ij} p_i \tag{2}$$

Subject to

$$\forall t_i \in T, \sum_{j=1}^{|w_i|} x_{ij} \le 1 \tag{3}$$

$$\forall t_i \in T, st_{ij} \le u_{ij} \le u_{ij} + l_i \le et_{ij}, \text{if} x_{ij} = 1 \tag{4}$$

$$\rho_{w_{ij} w_{i^* j^*}} = \begin{cases} 1 t_i is the predecessor of t_{i^*} \\ 0 otherwise \end{cases} \quad \forall t_i, t_{i^*} \in T \tag{5}$$

$$\forall t_i, t_{i^*} \in T, \Delta g = |\gamma^{u_{ij}} - \gamma^{u_{i^* j^*}}| + |\pi^{u_{ij}} - \pi^{u_{i^* j^*}}| + |\varphi^{u_{ij}} \tag{6}$$

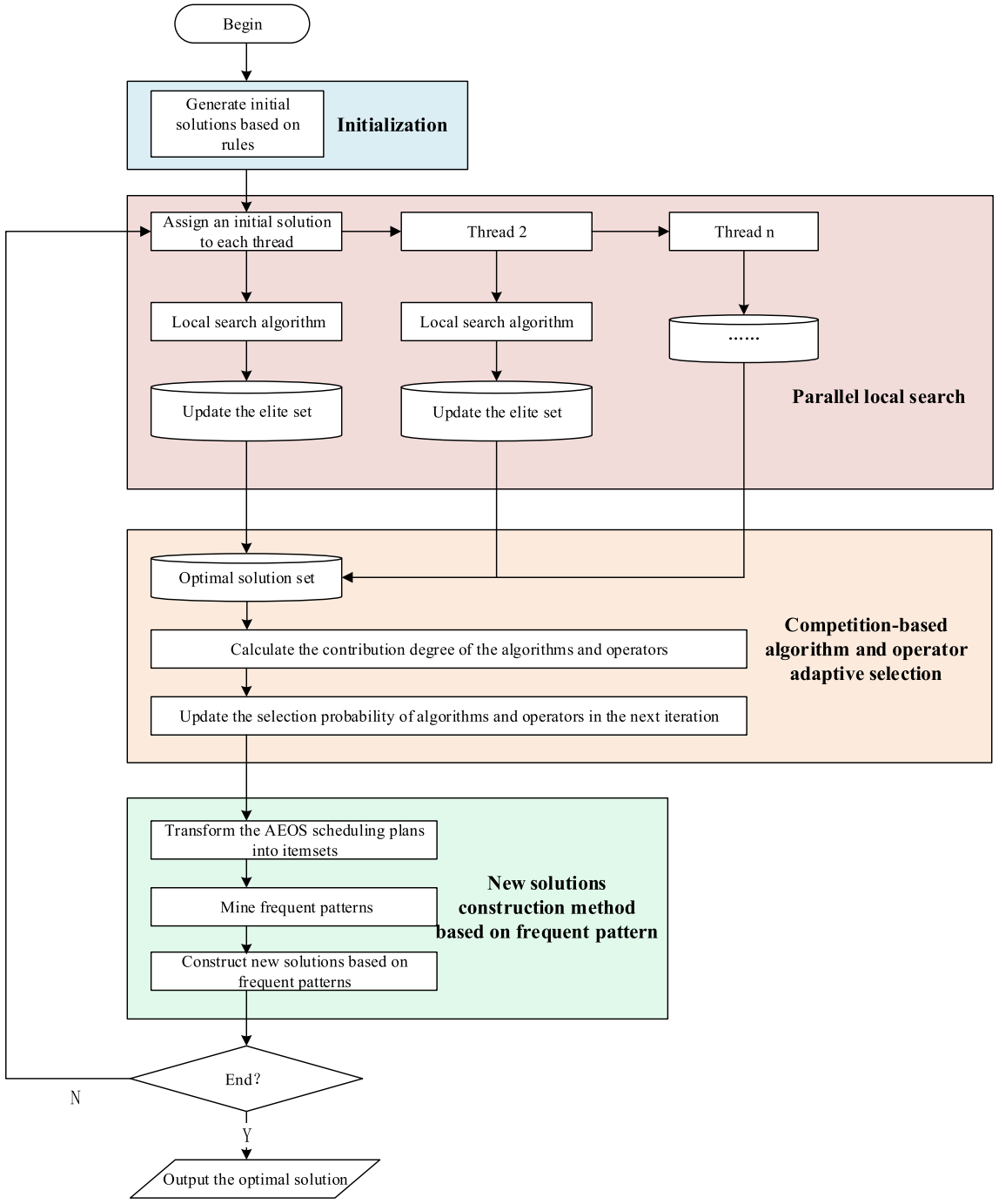

Fig. 2. Flowchart of the FPBPS algorithm.

$$\Delta(t_i, t_{i^*}) = \begin{cases} 11.66\Delta g \le 10 \\ a_1 + \dfrac{\Delta g}{v_1} 10 < \Delta g \le 30 \\ a_2 + \dfrac{\Delta g}{v_2} 30 < \Delta g \le 60 \\ a_3 + \dfrac{\Delta g}{v_3} 60 < \Delta g \le 90 \\ a_4 + \dfrac{\Delta g}{v_4} \Delta g > 90 \end{cases} \tag{7}$$

$$\forall t_i, t_{i^*} \in T, i \ne i^*, w_{ij} \in w_i, w_{i^*j^*} \in w_{i^*}, u_{ij} + l_i + \Delta(t_i, t_{i^*}) \le u_{i^*j^*}, \text{if} \rho_{w_{ij} w_{i^*j^*}} = 1 \tag{8}$$

$$\sum_{i=1}^{N} \sum_{j=1}^{|w_i|} x_{ij} l_i m_i \le M \tag{9}$$

$$\forall t_{i^*} \in T, w_{i^*j^*} \in w_{i^*} \sum_{i=1}^{N} \sum_{j=1}^{|w_i|} \left(x_{ij} l_i e_i + \rho w_{ij} w_{i^*j^*} e^f + \rho w_{ij} w_{i^*j^*} \Delta g e^s \right) \lesssim E \tag{10}$$

1) Formula (3) represents that each task cannot be executed multiple times.
2) Formula (4) represents that the task must be executed within the VTW.
3) Formula (7) is piecewise function to calculate the transition time between tasks.$v_1 = 1.5°/s$, a_1=5, v_2= 2°/s, a_2=10, v_3= 2.5°/s, a_3=16, v_4= 3°/s, and$a_4 = 22$ [16].
4) Formula (8) represents that any two tasks cannot violate the transition time constraint.
5) Formula (9) represents that the used memory cannot exceed maximum memory M.
6) Formula (10) represents that the used electricity cannot exceed maximum electricity E.

3. Related method

Fig. 2 is the flowchart of the FPBPS algorithm. The basic idea of the proposed FPBPS algorithm is to hybridize the different local search algorithms and data mining. The FPBPS algorithm is composed of four parts: 1) an initialization procedure; 2) a parallel local search procedure; 3) a competition-based algorithm and operator adaptive selection procedure; and 4) a new solutions construction method based on frequent pattern.

Algorithm 1: FPBPS

Input: The tasks set T; the algorithm pool A; the operator pool O; the elite set U_X, number of threads nt; the algorithm set U_A; the operator set U_O; the max size of elite set K; the set of probability for algorithm P_A; the set of probability for operator P_o; the set of frequent patterns U_{FP}; a termination condition.
Output: The solution X

1. $U_I \leftarrow$ Generate multiple initial solutions//Initialization (Section 3.1)
2. Initialize $U_X, U_A, U_O, P_A, P_o, U_{FP}$
3. **While** the termination condition is not satisfied **do**
4. //Parallel local search (Section 3.2)
5. Use the roulette strategy to select an initial solution from U_I
6. Each thread selects an algorithm from algorithm pool A based on P_A
7. Select a removal operator R and a sorting operator I from operator pool O based on P_o
8. $X' \leftarrow$Construct a new solution $I(R(X))$
9. **If** the condition for accepting the new solution X' is satisfied **then**
10. $X \leftarrow X'$
11. **End if**
12. $U_X \leftarrow$Update the elite set U_X with X'
13. $U_O \leftarrow$Record the operator R and I in U_O
14. //Competition-based algorithm and operator adaptive selection (Section 3.3)
15. $U_X \leftarrow$Merge the elite set from each thread
16. P_A, $P_o \leftarrow$Update the P_A and P_o
17. //New solutions construction method based on frequent pattern (Section 3.4)
18. Transform the elite set U_X to a suitable pattern for AEOS scheduling
19. $U_{FP} \leftarrow$ Frequent pattern mining with FP-growth
20. $U_I \leftarrow$ Construct new solutions with U_{FP}
21. **End while**
22. **Return** X

The FPBPS algorithm starts from multiple initial solutions, which are provided by the initialization procedure (Section 3.1) and assigns an initial solution to each thread. Lines 4–21 is the main loop of the FPBPS algorithm. Each thread selects an algorithm, a

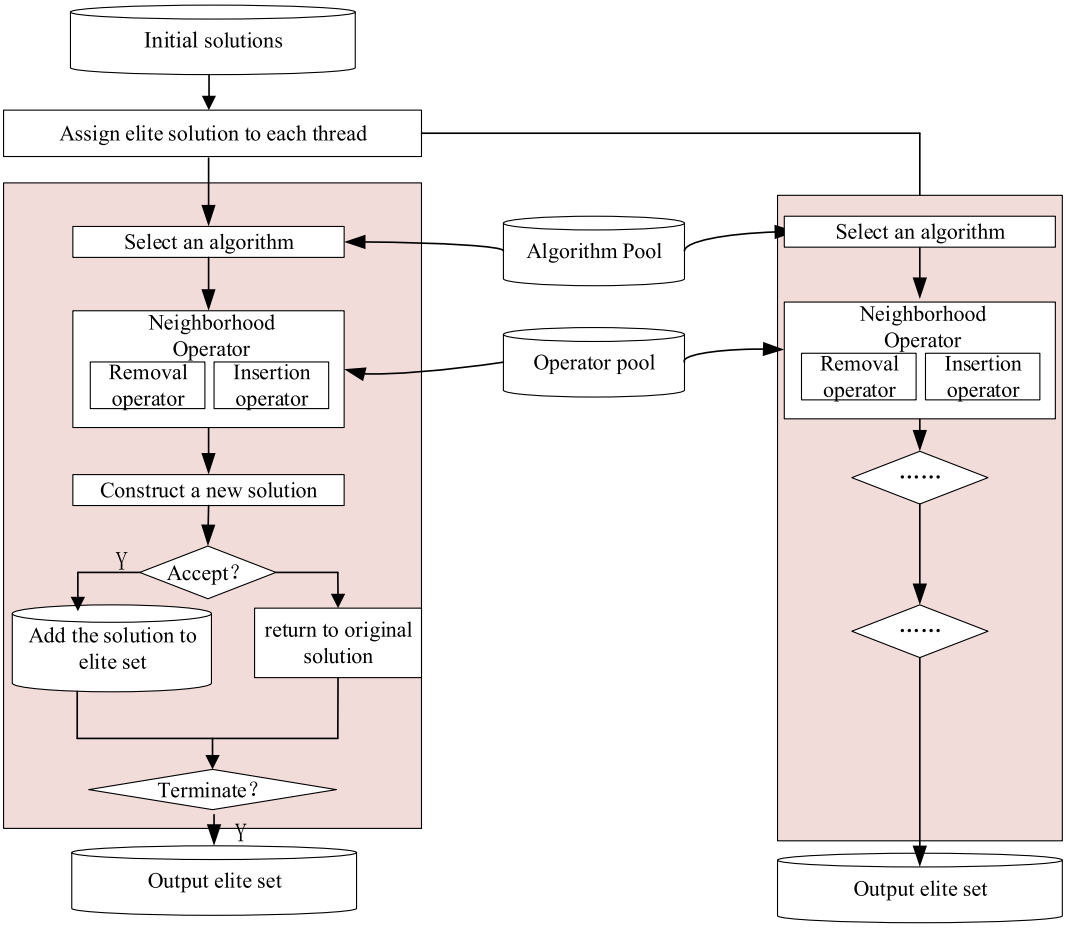

Fig. 3. Flowchart of parallel local search.

neighbourhood operator based on probability independently (Lines 6–14). Next, the main thread merges the elite set from each thread and updates the probability of each algorithm and operator (Lines 16–17). Finally, the frequent pattern mining is worked on the elite set to mine the problem-specific patterns and construct new solutions. The new solutions are assigned to different threads for next-round optimization (Lines 19–21). The process is repeated until a termination condition is satisfied. This paper gives two termination conditions, which are the maximum iterations and the consecutive number of generations without improvement.

3.1. Initialization procedure

A high-quality initial solution has the significant impact on the final solution. Three heuristic rules are adopted to sort tasks in this paper, which are earliest start time, priority and the number of VTW. the arranged tasks are inserted into the empty plan one by one, if the task can be inserted without violating constraints, the task is started as early as possible. Otherwise, it will be abandoned.

3.2. Parallel local search procedure

Fig. 3 is the flowchart of parallel local search procedure. The parallel local search procedure is the first part of the main loop. In the process of search, each thread selects an algorithm and a neighbourhood operator to construct a new solution independently. In the search process, an elite set is maintained to record the high-quality solutions continually, and operators that construct new solutions are recorded in the operator set. The algorithm pool and operator pool are described in detail as follows.

3.2.1. Algorithm pool

The algorithm tool provides different local search algorithms for each thread. At the beginning, the probability that the local search algorithms are allocated to each thread is the same, and it is updated continuously in subsequent iterations. Three improved local search algorithms are added to the algorithm pool, which are improved hill climbing (IHC), improved tabu search (ITS), and improved simulated annealing (ISA).

Algorithm 2: IHC.

Input: The initial solutions U_I, objective function $F(X)$, the max size of elite set K, operator pool O, a termination condition
Output: The current solution X, the elite set U_X, the operator set U_O

1. $U_X, U_O = \emptyset$
2. $X \leftarrow$ Choose an initial solution from the U_I
3. **Repeat**
4. $R, I \leftarrow$Select a removal operator R and a sorting operator$I$from operator pool O according to probability
5. $X' \leftarrow I(R(X))$
6. **If** X' in U_X **then**

First in first out

7. Replace the same solution in U_X with X'
8. Record the operator R and$I$in U_O
9. **Else:**
10. **If** the size of elite set $U_X < K$ **then**
11. Record the current solution X' in U_X
12. Record the operator R and I in U_O
13. **Else:**
14. Find$X^* \leftarrow argmin\{F(X) : X \in U_X\}$
15. **If** $F(X') \geq F(X^*)$
16. ReplaceX^* by X'
17. Record the operator $R$and I in U_O
18. **End if**
19. **End if**
20. **End if**
21. **If** $F(X') \geq F(X)$ **then**
22. $X \leftarrow X'$
23. **End if**
24. **Until** termination condition is satisfied
25. **Return** X, U_X, U_O

In Algorithm 2, a neighbourhood operator combining a removal operator and sorting operator is applied to construct the new solution (Lines 4–5). In the neighbourhood operator, some tasks are first deleted from an initial solution through the removal operator. Then, the unscheduled tasks and tasks deleted by a removal operator are sorted in a heuristic rule, and the insertion method is used to arrange tasks in turn. The neighbourhood operator will be discussed in Section 3.2.2. Next, the algorithm enters the procedure that updates the elite set (Lines 6–20). In this procedure, we first check whether the new solution X' is in the elite set, and if so, we use first-in-first-out (FIFO) to update the elite set and operator set (Lines 6–8). Otherwise, the new solution X' is inserted into the elite set if one of the following two conditions is satisfied: 1) if the size of elite set is less than threshold K, the new solution is added directly to the elite set (Lines 10–12); and 2) if the size of the elite set is K and $F(X') \geq F(X^*)$, where X^* is $argmin\{F(X) : X \in U_X\}$, the solution X^* is replaced by X'. Subsequent algorithms will also adopt this method to update the elite set and operator set. In Lines 21–22, the condition that it is accepted is $F(X') \geq F(X)$.

Algorithm 3: ITS

Input: The initial solutions U_I, objective function $F(X)$, the max size of elite set K, operator pool O, tabu set U_T, tabu size t, a termination condition
Output: The current solution X, the elite set U_X, the operator set U_O

1. $U_X, U_O, U_T = \emptyset$
2. $X \leftarrow$ Choose an initial solution from the U_I
3. **Repeat**
4. $R, I \leftarrow$Select a removal operator R and a sorting operator$I$from operator pool O based on probability
5. $X' \leftarrow I(R(X))$
6. **If** X' in U_X **then**

First in first out

7. Replace the same solution in U_X with X'
8. Record the operator R and$I$in U_O
9. **Else:**
10. **If** the size of elite set $U_X < K$ **then**
11. Record the current solution X' in U_X
12. Record the operator R and I in U_O
13. **Else:**
14. Find$X^* \leftarrow argmin\{F(X) : X \in U_X\}$
15. **If** $F(X') \geq F(X^*)$
16. ReplaceX^* by X'
17. Record the operator $R$and I in U_O
18. **End if**
19. **End if**
20. **End if**
21. **If** $F(X') \geq F(X)$ **then**
22. $X \leftarrow X'$
23. Update the tabu set U_T with the tasks by the sorting operator I

(*continued on next page*)

(*continued*)

```
24.     Else:
25.       Update the tabu set U_T with the tasks by the removal operator R
26.     End if
27.   Until termination condition is satisfied
28.   Return X, U_X, U_O
```

Based on Algorithm 2, an improved tabu search for AEOS scheduling is proposed. In each iteration, if the new solution is accepted, the tasks that are inserted into the new solution are forbidden to be deleted for the next t iterations (Line 23), otherwise, the tasks that are deleted from the current plan are forbidden to be deleted for the next t iterations (Line 25). Meanwhile, if the number of deleted tasks is less than the threshold, the remaining tasks are selected to be deleted randomly until the number is satisfied. The tabu strategy is also applied to Algorithm 4.

Algorithm 4: ISA

```
Input: The initial solutions U_I, objective function F(X), the max size of elite set K, operator pool O, tabu set U_T, tabu size t, initial temperature T_init, cooling
       coefficientc, a termination condition
Output: The current solution X, the elite set U_X, the operator set U_O
1.   U_X, U_O, U_T = Ø
2.   T = T_init
3.   X← Choose an initial solution from the U_I
4.   Repeat
5.     R, I ←Select a removal operator R and a sorting operatorIfrom operator pool O according to probability
6.     X'←Construct a new solution I(R(X))
7.     If X' in U_X then
     First in first out
8.       Replace the same solution in U_X with X'
9.       Record the operator R andIin U_O
10.    Else:
11.      If the size of elite set U_X < K then
12.        Record the current solution X' in U_X
13.        Record the operator R and I in U_O
14.      Else:
15.        FindX* ←argmin{F(X) : X ∈ U_X}
16.        If F(X') ≥ F(X*)
17.          ReplaceX* by X'
18.          Record the operator Rand I in U_O
19.        End if
20.      End if
21.    End if
22.    If F(X') ≥ F(X) then
23.      X ←X'
24.      Update the tabu set U_T with the tasks by the sorting operator I
25.    Else:
26.      If random(0,1)⟨p then
27.        X ←X'
28.        Update the tabu set U_T with the tasks by the sorting operator I
29.      Else:
30.        Update the tabu set U_T with the tasks by the removal operator R
31.      End if
32.    End if
33.    T = T*c
34.  Until termination condition is satisfied
35.  Return X, U_X, U_O
```

In Lines 22–31, if $F(X^{'}) \geq F(X)$, the current solution will be updated by X' in the next iteration. If $F(X^{'})\langle F(X)$, the new solution is accepted based on p. In this paper, p is set as follows:

$$p = \exp\left\{\frac{100}{T}\left(\frac{F(X^{'}) - F(X)}{F(X)}\right)\right\} \tag{11}$$

where T is the temperature, and a linear simulated annealing method $T = T^{*}c$ is used to update the temperature in each iteration. c is the annealing coefficient. The value of initial temperature T_{init} is the same as reference [15], and X_o is the initial solution.

$$T_{init} = \frac{-0.05}{\ln 0.5} * F(X_o) \tag{12}$$

3.2.2. Operator pool

The neighbourhood operators are used to construct new solution in the FPBPS, which combine different removal operators and

Table 1
The time complexity of operators.

		Time complexity
Removal operators	*Random removal operator*	$O(n)$
	Min profit removal operator	$O(n\log n)$
	Min unit profit removal operator	$O(n\log n)$
	Max transition time removal operator	$O(n\log n)$
	Max opportunity removal operator	$O(n\log n)$
	Max conflict removal operator	$O(n\log n)$
	Worst route removal operator	$O(n)$
	Worst route removal operator	$O(n)$
Sorting operators	*Random sorting operator*	$O(n)$
	Max profit sorting operator	$O(n\log n)$
	Max unit profit sorting operator	$O(n\log n)$
	Min transition time sorting operator	$O(n\log n)$
	Min opportunity sorting operator	$O(n\log n)$
	Min conflict sorting operator	$O(n\log n)$
	Min distance sorting operator	$O(n\log n)$

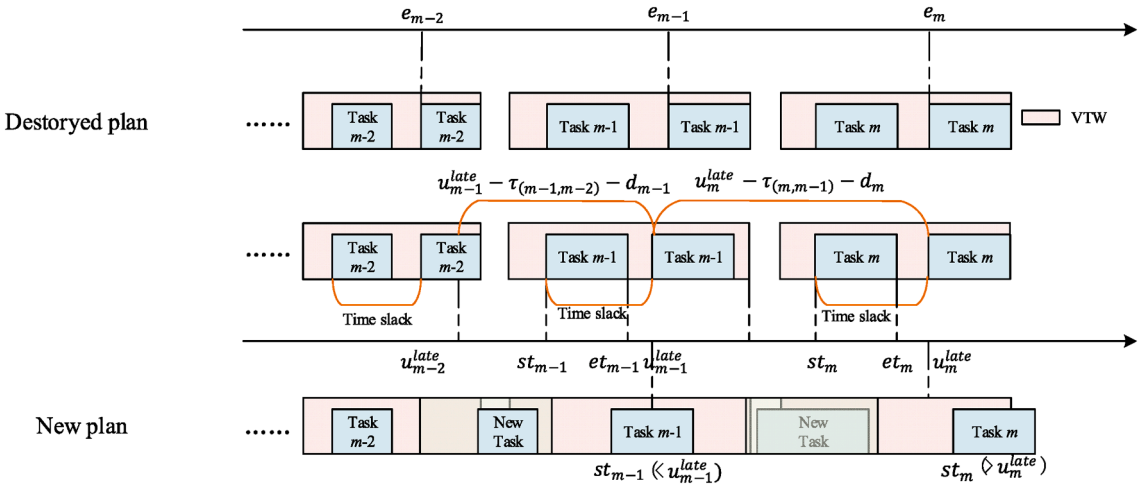

Fig. 4. The FI method.

sorting operators. The method is as follows. First, the removal operator is selected to remove some tasks from the current solution. Next, the rest tasks are sorted by sorting operators. Finally, the fast insertion (FI) method is applied to insert the rest tasks into destroyed solution one by one. We briefly introduce the complexity of these operators and the FI method in this paper, more details can be found in the reference [15]. Table 1 shows the time complexity of removal operators and sorting operators.

In the FI method, the time slack strategy is used to judge the feasibility of inserting a task into each position of the destroyed solution. Time slack is the latest time that a task can be postponed. In AEOS scheduling problem, the latest start time of a task depends on its successor tasks, so we adopt a backwards propagation mode to calculate the time slack. The destroyed plan $dp = \{t_1, t_2 \cdots t_m\}$ has m tasks, and the latest start time of task t_i is calculated by Formula 13.

$$u_i^{late} = \begin{cases} min\{(u_{i+1}^{late} - TransitionTime(t_i, t_{i-1}) - d_{i-1}), e_{i-1}\}, 1 \le i < m \\ e_i, i = m \end{cases} \tag{13}$$

In Fig. 4, a task is inserted into the position between tasks t_{m-2} and t_{m-1}, and the start time of t_{m-1} is postponed to st_{m-1}. This position is feasible if st_{m-1} is earlier than u_{m-1}^{late}. However, when inserted into the position between tasks t_{m-1} and t_m, st_m is later than u_m^{late}, and the position is abandoned. The transition time increment when task t_i is inserted at each position in p_{t_i} is calculated, and the best position is selected to minimize the transition time increment.

3.3. Competition-based algorithm and operator adaptive selection

The competition procedure is the third part in FPBPS, which is used to evaluate the contribution of algorithms, removal operators and sorting operators with the purpose of updating the probability in the next iteration. The process of the competition strategy is

	1	2	3	4	5	6	7
Plan 1	5	4	7	2	1	6	3
Plan 2	7	4	5	3	1	6	2
Plan 3	7	4	5	2	3	6	1

⇒

	1	2	3	4	5	6
Plan 1	5,4	4,7	7,2	2,1	1,6	6,3
Plan 2	7,4	4,5	5,3	3,1	1,6	6,2
Plan 3	7,4	4,5	5,2	2,3	3,6	6,1

Fig. 5. Transformation procedure.

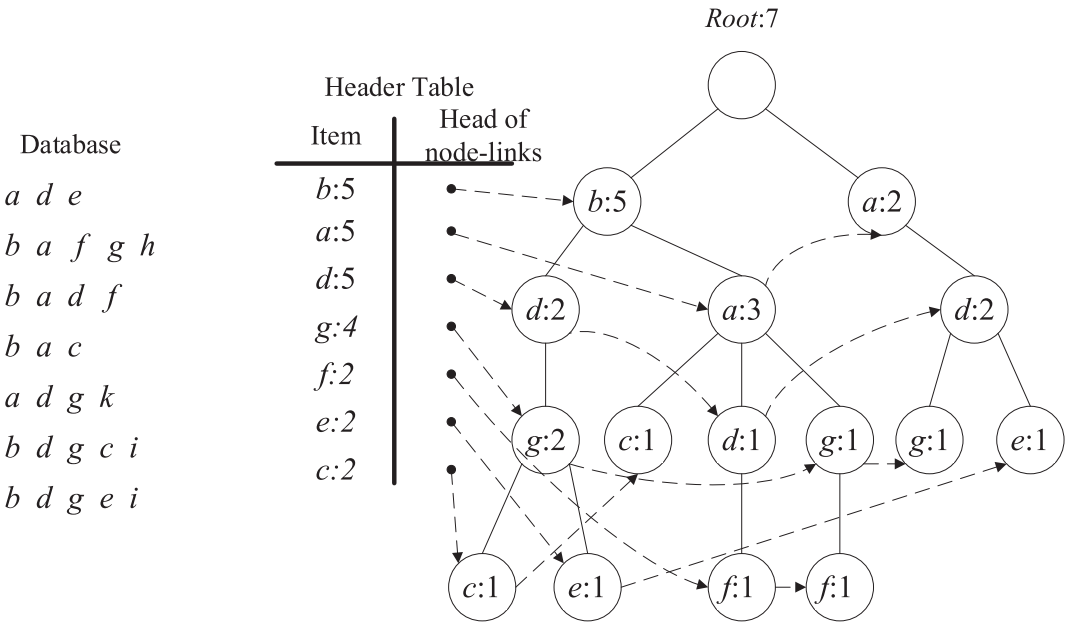

Fig. 6. Construction of FP-tree.

described in detail as follows.

Step 1: Generate the historical optimal solution set. The elite sets obtained by n_T threads are represented by U_x^1, U_x^2…, $U_x^{n_T}$, and the optimal solution set is $U_X = \bigcup_{i=1}^{n_T} U_x^i$.

Step 2: Sort the historical optimal solution set U_X in descending order of objective value.

Step 3: Let the size of the current optimal solution set P_X be n_P, and P_X can be generated according to the formula:

$$P_X = \begin{cases} \{X_i | X_i \in U_X, i \le n_P\}, |U_X| \ge n_P \\ U_X, |U_X| < n_P \end{cases} \tag{14}$$

Step 4: Update the probability of algorithms, removal operators and sorting operators in the next iteration. We record the algorithm, removal operator and sorting operator that create each solution in P_X. Suppose the i-th algorithm in the algorithm pool has contributed n_A^i solutions to P_X, the i-th removal operator has contributed n_R^i solutions to P_X, and the i-th sorting operator has contributed n_I^i solutions to P_X. Then, the contributions of the i-th algorithm c_A^i, removal operator c_R^i, and sorting operator c_I^i can be expressed by the following formulas:

$$c_A^i = \frac{n_A^i}{n_P}, i \in \text{Algorithm pool} \tag{15}$$

$$c_R^i = \frac{n_R^i}{n_P}, i \in \text{Removal operator pool} \tag{16}$$

$$c_I^i = \frac{n_I^i}{n_P}, i \in \text{Sorting operator pool} \tag{17}$$

The probability of the ith algorithm P_A^i, removal operator P_R^i, and sorting operator P_I^i can be expressed by the following formulas:

$$P_A^i = P_A^i * (1 - \partial) + \frac{n_A^i}{n_P} * \partial, i \in \text{Algorithm pool} \tag{18}$$

Table 2
Algorithm parameters.

Parameter	Assignment		
	FPBPS	ALNS/TPF	ALNS/I
Percentage of destroyed tasks	10%	10%	10%
Annealing coefficient	1000^(1/1000)	1000^(1/1000)	1000^(1/1000)
Tabu list length	0.1*N	0.1*N	0.1*N
Maximum number of elites	50	–	–
Maximum number of generations	100	500	500
Maximum consecutive number of generations without improvement	50	50	50
Rate of historical probability	0.5	0.5	0.5
Number of solutions for frequent pattern mining	10	–	–
Minimum support	0.2	–	–
Number of threads	5	–	–

Table 3
Experimental results with different algorithms.

ID	Number of tasks	Profit rate (%)								
		FPBPS			ALNS/TPF			ALNS/I		
		Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
Scenario 1	50	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
Scenario 2	75	**100%**	**100%**	**100%**	97.30%	99.46%	**100%**	98.11%	99.76%	**100%**
Scenario 3	100	**100%**	**100%**	**100%**	89.52%	95.56%	98.40%	91.47%	95.58%	98.22%
Scenario 4	125	**97.39%**	**99.00%**	**99.85%**	77.45%	89.68%	98.01%	83.13%	90.83%	95.40%
Scenario 5	150	**93.68%**	**94.89%**	**96.07%**	83.19%	89.32%	91.78%	82.96%	87.60%	90.46%
Scenario 6	175	**91.37%**	**92.24%**	**93.08%**	83.55%	86.39%	88.97%	83.35%	84.57%	88.87%
Scenario 7	200	**83.33%**	**84.46%**	**85.86%**	64.89%	77.90%	83.15%	71.35%	74.66%	80.24%
Scenario 8	225	**75.77%**	**79%**	**81.43%**	67.32%	72.54%	74.94%	65.21%	69.87%	72.53%
Scenario 9	250	**74.01%**	**75.78%**	**78.03%**	60.01%	67.04%	70.69%	60.78%	65.33%	69.58%
Scenario 10	275	**67.74%**	**70.28%**	**72.12%**	56.41%	62.43%	65.42%	56.09%	61.12%	64.84%
Scenario 11	300	**66.21%**	**67.80%**	**69.13%**	51.52%	57.72%	65.97%	53.42%	58.75%	64.54%
Scenario 12	325	**66.07%**	**68.56%**	**71.22%**	57.45%	60.86%	64.17%	50.51%	58.84%	63.63%
Scenario 13	350	**63.06%**	**64.80%**	**66.29%**	52.57%	57.53%	62.28%	51.67%	54.34%	58.59%
Scenario 14	375	**57.54%**	**60.96%**	**62.87%**	45.79%	50.85%	54.69%	49.83%	52.80%	56.16%
Scenario 15	400	**58.14%**	**59.90%**	**61.60%**	50.14%	53.71%	58.29%	40.53%	51.70%	56.25%

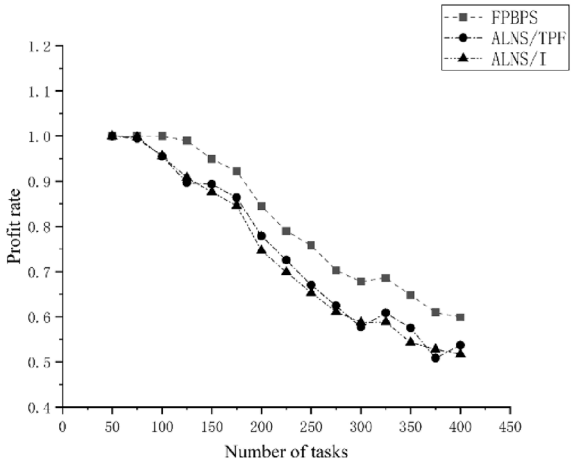

Fig. 7. Values of profit rate.

$$P_R^i = P_R^i * (1-\partial) + \frac{n_R^i}{n_P} * \partial, i \in \text{Removal operator pool} \tag{19}$$

$$P_I^i = P_I^i * (1-\partial) + \frac{n_I^i}{n_P} * \partial, i \in \text{Sorting operator pool} \tag{20}$$

Table 4
Computation time with different algorithms.

ID	Number of tasks	Computation time (s)		
		FPBPS	ALNS/TPF	ALNS/I
Scenario 1	50	2.7	**0.1**	**0.1**
Scenario 2	75	5.8	1.1	**1.0**
Scenario 3	100	21.0	**10.0**	13.3
Scenario 4	125	43.0	34.9	**29.2**
Scenario 5	150	**56.3**	149.2	84.6
Scenario 6	175	**71.6**	122.2	119.1
Scenario 7	200	**85.3**	114.5	119.8
Scenario 8	225	**107.2**	120.9	167.2
Scenario 9	250	**137.2**	189.0	192.5
Scenario 10	275	**143.8**	219.3	200.3
Scenario 11	300	**162.9**	242.7	265.2
Scenario 12	325	**184.5**	329.3	310.9
Scenario 13	350	**211.6**	378.2	373.9
Scenario 14	375	**249.9**	391.2	454.3
Scenario 15	400	**249.4**	473.1	422.5

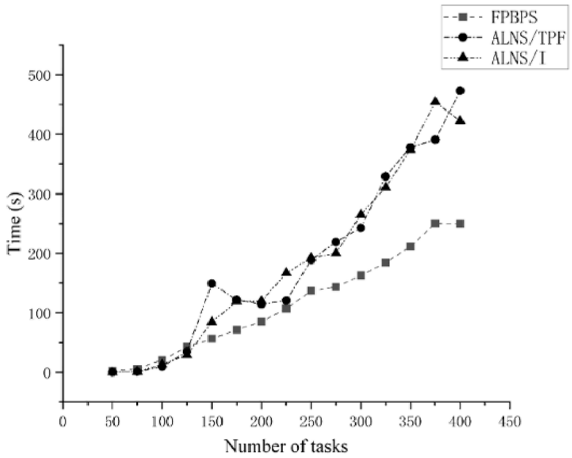

Fig. 8. Computation time.

where ∂ is the rate of historical probability, which prevents algorithms and operators from performing poorly in this round and not participating in subsequent iterations. Initially, each algorithm and operator have the same probability.

3.4. New solutions construction method based on frequent pattern

New solutions construction based on frequent pattern is the last part in FPBPS. The frequent pattern mining method in this paper is applied to mine specific patterns that often occur in high-quality solutions simultaneously. Then, new solutions are constructed based on mined patterns [3536]. The main process includes the following three steps [37].

Step 1: Data transform. In AEOS task scheduling problem, the execution time of a task is depend on the previous task. Therefore, a frequent pattern is defined, which is combination of adjacent tasks. The transform procedure is shown in Fig. 5.

Step 2: mining the frequent patterns. In this paper, we use the FP-growth to mine the knowledge with respect to the AEOS scheduling. The main characteristic of FP-growth is to use the tree structure to store data, which is named by frequent pattern tree (FP-tree). This structure can speed up the algorithm. Fig. 6 shows the construction of FP-tree.

Step 3: Constructing the new solutions based on the selected frequent patterns. We use the roulette strategy to select the high-quality solutions to guide the construction process. Specifically, tasks in this solution that are not in the frequent pattern and unscheduled tasks are recognized, and the FI method (Section 3.2.2) is applied to insert the remaining tasks in turn until they cannot be inserted. This process is repeated until the number of new solutions is equal to the number of threads. Finally, the constructed new solutions are improved by parallel local search.

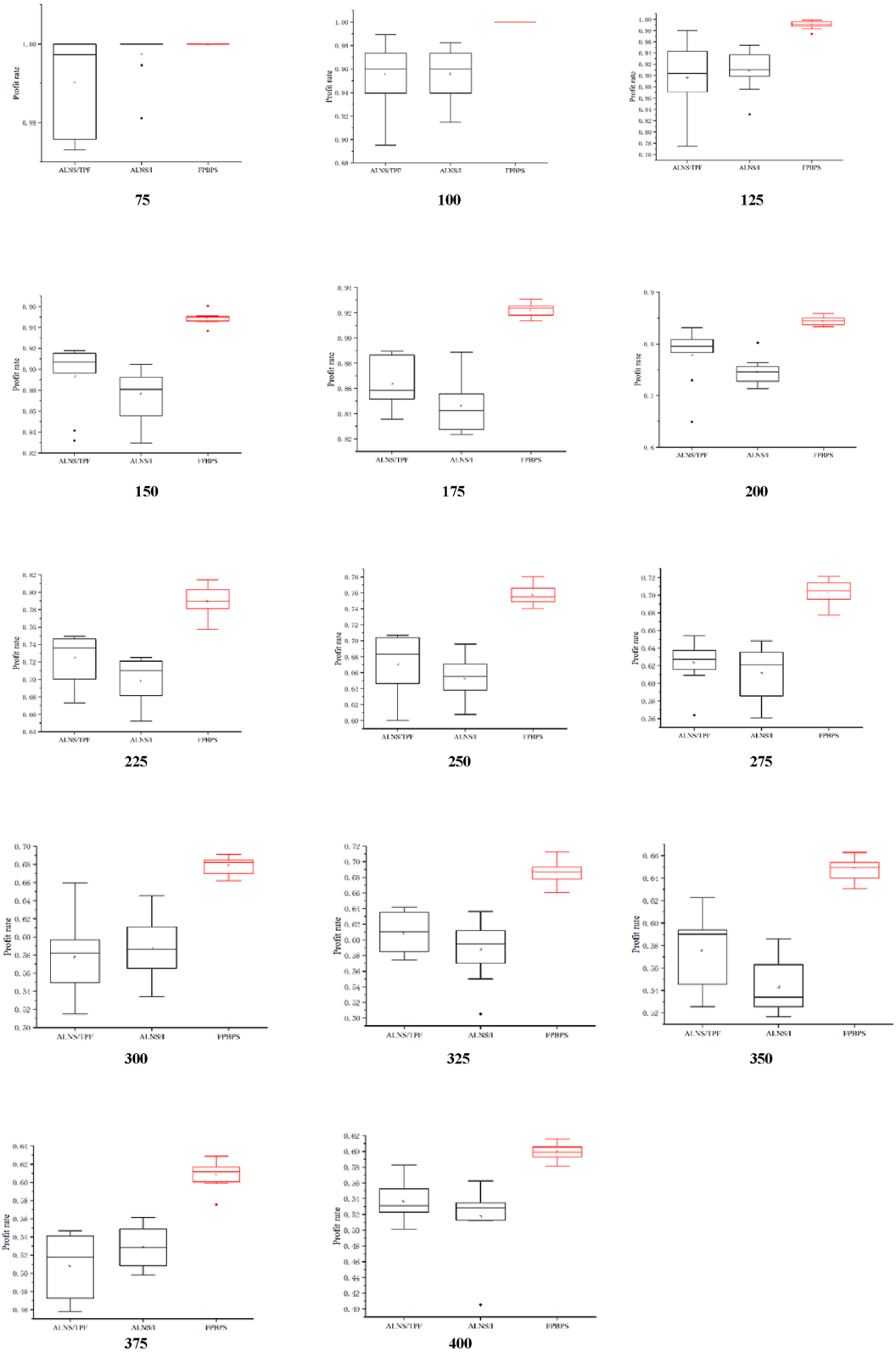

Fig. 9. Stability of different algorithms.

Table 5
The standard deviation of results.

ID	Number of tasks	Standard deviation		
		FPBPS	ALNS/TPF	ALNS/I
Scenario 2	75	**0**	0.012190787	0.006020437
Scenario 3	100	**0**	0.027414652	0.020956973
Scenario 4	125	**0.007275179**	0.060201639	0.035676135
Scenario 5	150	**0.00663428**	0.03282644	0.024651716
Scenario 6	175	**0.005439483**	0.018379682	0.020608792
Scenario 7	200	**0.00855887**	0.053018271	0.02524118
Scenario 8	225	**0.018264195**	0.026294905	0.027218219
Scenario 9	250	**0.012866734**	0.039045463	0.030635282
Scenario 10	275	**0.013857893**	0.02495211	0.02945132
Scenario 11	300	**0.009533829**	0.041415634	0.034567259
Scenario 12	325	**0.014488327**	0.024872544	0.038667443
Scenario 13	350	**0.0095685**	0.032790317	0.023267182
Scenario 14	375	**0.015483341**	0.036944235	0.02214491
Scenario 15	400	**0.010384213**	0.026131903	0.042041882

Table 6
The influence of frequent pattern mining on profit rate.

ID	Number of tasks	Profit rate (%)	
		PLS	FPBPS
Scenario 1	50	**100%**	**100%**
Scenario 2	75	**100%**	**100%**
Scenario 3	100	98.29%	**100%**
Scenario 4	125	95.40%	**99.00%**
Scenario 5	150	91.38%	**94.89%**
Scenario 6	175	88.05%	**92.24%**
Scenario 7	200	79.19%	**84.47%**
Scenario 8	225	73.43%	**79%**
Scenario 9	250	69.67%	**75.78%**
Scenario 10	275	64.73%	**70.28%**
Scenario 11	300	62.94%	**67.80%**
Scenario 12	325	62.42%	**68.59%**
Scenario 13	350	58.55%	**64.80%**
Scenario 14	375	57.54%	**60.96%**
Scenario 15	400	56.42%	**59.90%**

4. Computational experiments

To evaluate the FPBPS algorithm, we first design the multiple experimental scenarios and set the values of parameters in the FPBPS algorithm. Then, the FPBPS algorithm is compared with two state-of-the-art algorithms in the field of AEOS task scheduling, and the performance of each algorithm is given in profit rate, running time and robustness. Moreover, we further analyse the impact of parallel local search, frequent pattern mining and competition on the final solutions.

4.1. Experimental environment

Due to the differences arising from task, design and adopted technologies, the satellites used in different countries differ significantly in terms of capabilities, constraints and management model, providing no benchmark for AEOS scheduling. We use the same method proposed by Liu to design the AEOS scheduling scenarios [14]. In all scenarios, targets are distributed at 3^{o}N-53^{o}N and 74^{o}E-133^{o}E. The scheduling period is from 2013/04/20/00:00:00 to 2013/04/20/23:59:59, and the number of orbits is sixteen. Fifteen scenarios are designed, in which the number of tasks is ranges from 50 to 400 in steps of 25.

The values of parameters in FPBPS algorithm and two comparison algorithms are listed in Table 2. The two comparison algorithms are ALNS/I [14] and ALNS/TPF [38], which had been proven to be effective in solving the AEOS scheduling problem. Due to the parallel strategy is used in the FPBPS algorithm, the assignment of parameters is the same except for the maximum number of generations. To show the comparison results more objectively, we use the same number of evaluations. The maximum number of generations of ALNS/TPF and ALNS/I are set to 500, and the maximum number of generations of FPBPS are set to 100 because the number of threads is 5. All results are obtained by running the algorithm 10 times independently. All algorithms are coded in Python 3.7.6, and the experiments are conducted using an Intel (R) Core (TM) i5-8250U CPU 1.80 GHz under Windows Server 2010 with 8 GB RAM.

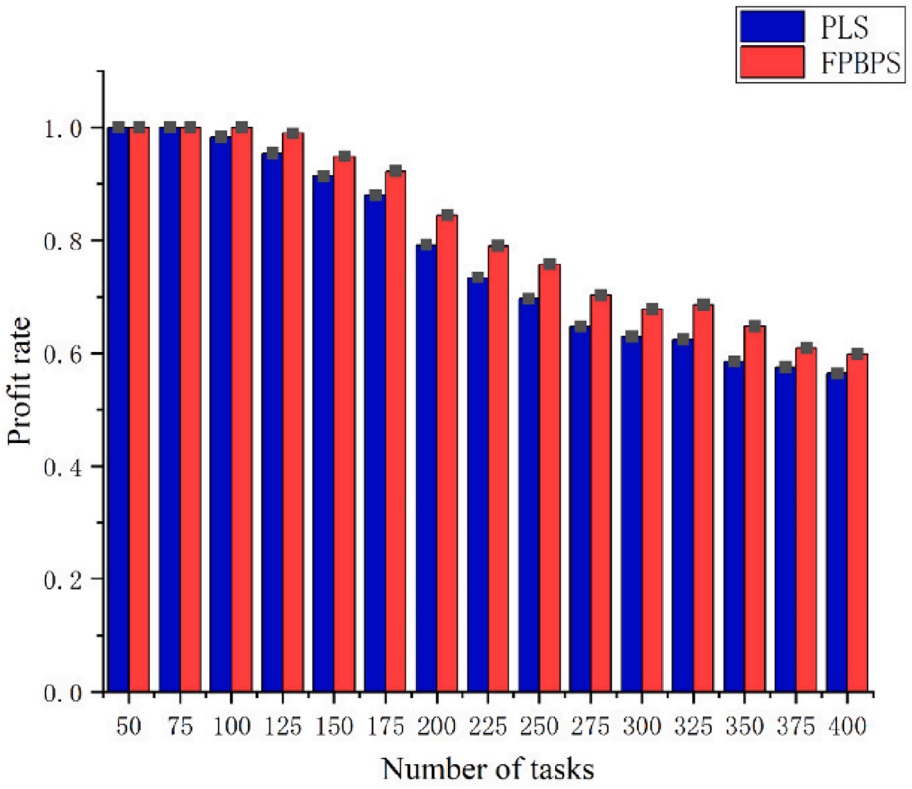

Fig. 10. The influence of frequent pattern mining on profit rate.

4.2. Experimental results

The solution quality of three algorithms is shown in Table 3 and Fig. 7. The profit rate is proposed to evaluate the quality of solution, which is the profit of scheduled tasks divided by the total profit. In Table 3, this paper represents the minimum value, average value and maximum value of each algorithm. When the scale of tasks is 50 or 100, there is no significant difference in the profit rate. As the scale increases, the gap between algorithms is becoming more and more obvious. The FPBPS algorithm can get the best profit rate in all algorithms. This is due to the parallel local search and frequent pattern mining are integrated into the FPBPS algorithm, which can provide a competitive result on the larger-scale task scenarios. The influence of parallel local search and frequent pattern mining on the experimental results will be discussed below. In conclusion, we can find that the performance of the FPBPS algorithm is very well in solving the AEOS scheduling problem.

The computation time of each algorithm is shown in Table 4 and Fig. 8. Since the FPBPS algorithm adopts a parallel architecture, the interaction between threads takes some time. As a result, when the task scale is small, the advantage of the proposed approach in running time is not outstanding. But, when the number of tasks is 150, the computation time of the FPBPS algorithm is shorter than other algorithms. The computation time of the FPBPS algorithm increases slowly with the increasing number of tasks. One of the most important reasons is the frequent pattern mining procedure, which can help the FPBPS algorithm converge quickly.

Another advantage of the FPBPS is that it has good robustness. In Fig. 9, we can find that the proposed FPBPS algorithm is more stable than other algorithms in most scenarios. The standard deviation of profit rate is shown in Table 5, the FPBPS has the minimum value of standard deviation in all scenarios.

In summary, the FPBPS takes the shortest time and has the highest-quality solution in these three algorithms. Meanwhile, when the scale of tasks increases, the advantages of the FPBPS algorithm become more obvious. Meanwhile, the FPBPS has good robustness. Next, we will analyse the role of parallel local search and frequent pattern mining in the FPBPS.

4.3. Further analysis

The parallel local search, new solutions construction method based on frequent pattern and the competition are analysed in detail. Since the difference is not obvious on small-scale tasks, we analyse only the situation when the task scale exceeds 200.

We compare the results of the FPBPS algorithm and the PLS in profit rate and robustness in Table 6. From Table 6 and Fig. 10, we can see that frequent pattern mining can significantly improve the value of profit rate, and the maximum increase is approximately 6%. Meanwhile, the robustness of the algorithm is also improved as shown in Fig. 11. The most prominent feature of AEOS task scheduling

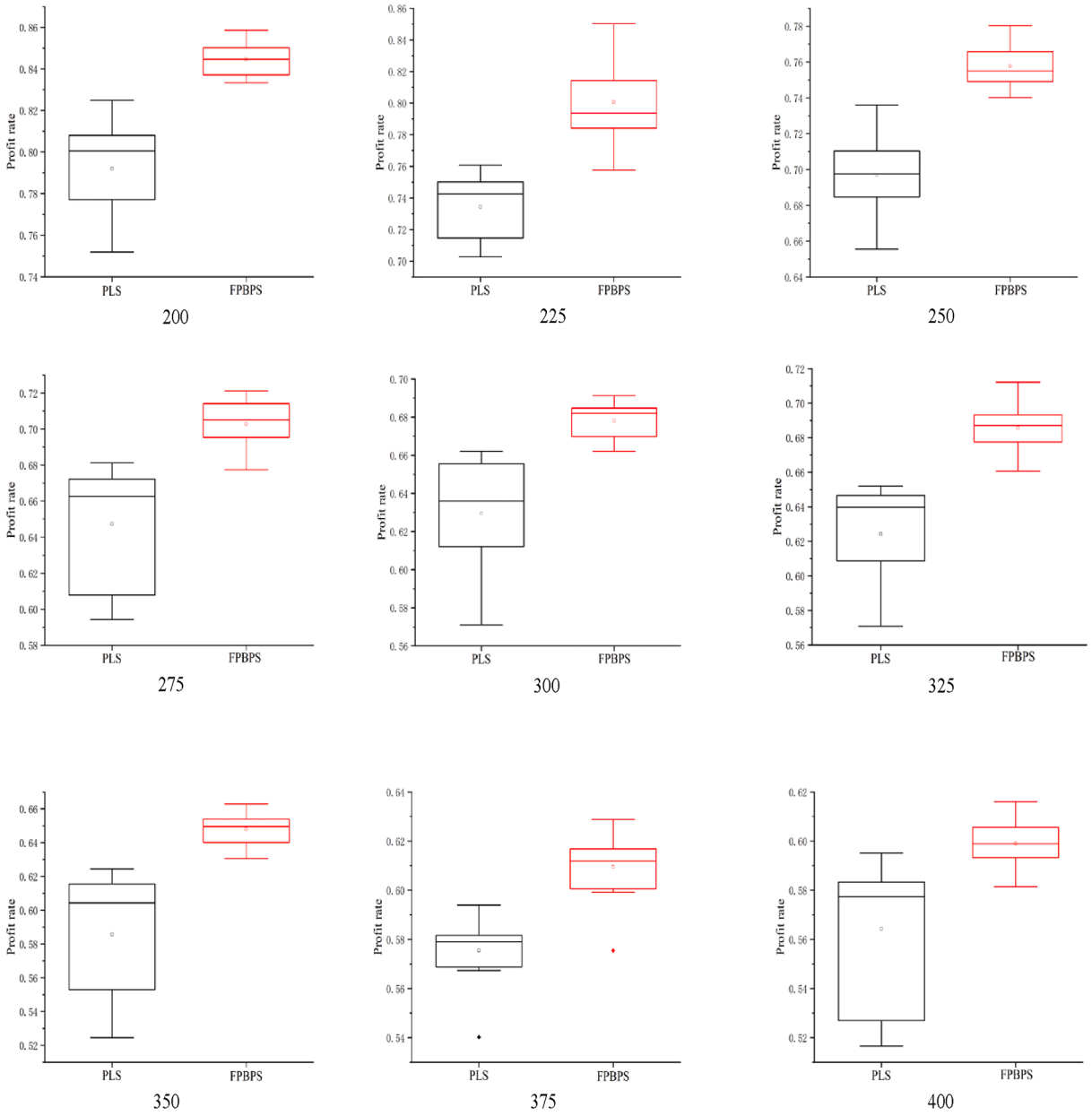

Fig. 11. The influence of frequent pattern mining on robustness.

problem is time-dependent, that is, the execution of current task depends on the previous task to a large extent. The prior experiment proved that the partial task sequences often occur in high-quality solutions simultaneously, which is known as frequent patterns in data mining. The role of frequent patterns is to construct new solutions and guide the search of algorithm. Further, to directly reflect the influence of frequent pattern-based new solution on the final result, the index *RI* is designed, which is calculated according to the following formula.

$$RI = \frac{num_{FP \cap FI}}{num_{FP}} \tag{21}$$

Where $num_{FP \cap FI}$ is the number of common tasks that appearing in the final solution and frequent patterns. num_{FP} is the number of tasks in the frequent pattern. Fig. 12 shows the value of *RI* in each scenario. From Fig. 12, we can find that the value of *RI* is high in each scenario, which indicates that the proportion of the mined frequent patterns in the final solution is also very high. Since the new solution is constructed based on the selected frequent patterns, the algorithm avoids invalid searches and the computational efficiency is improved. Meanwhile, the final solution is highly similar to the frequent pattern, so the algorithm has high robustness.

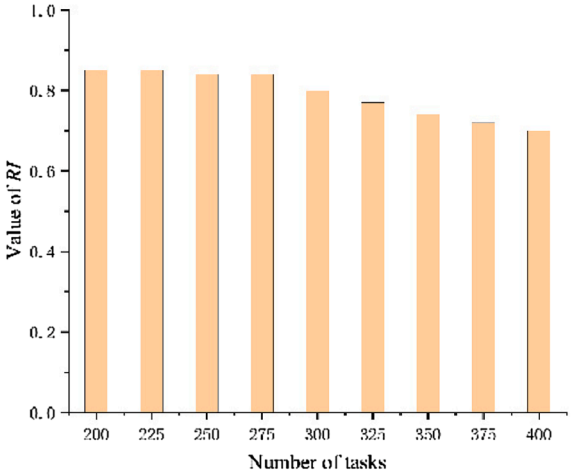

Fig. 12. The value of *RI*

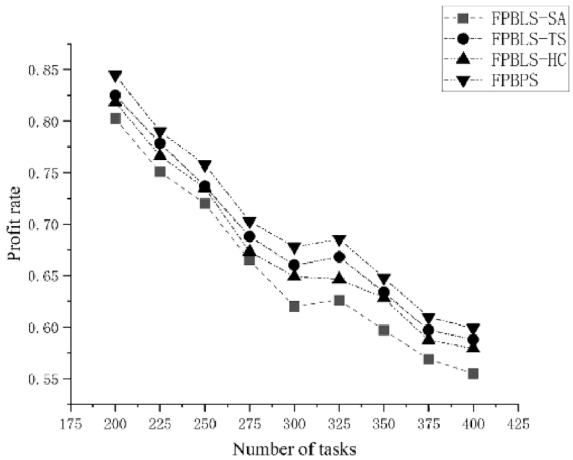

Fig. 13. The influence of parallel architecture on profit rate.

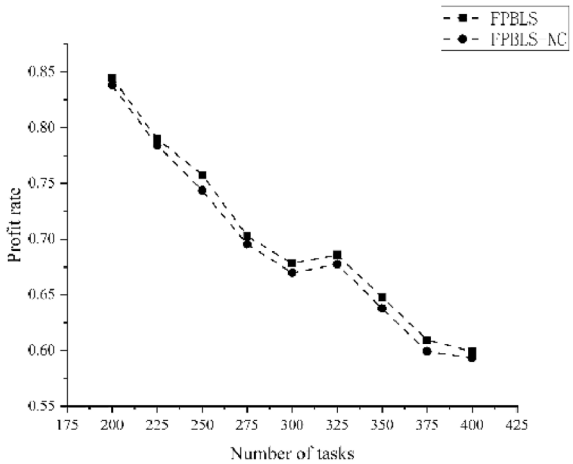

Fig. 14. The influence of competition strategy on profit rate.

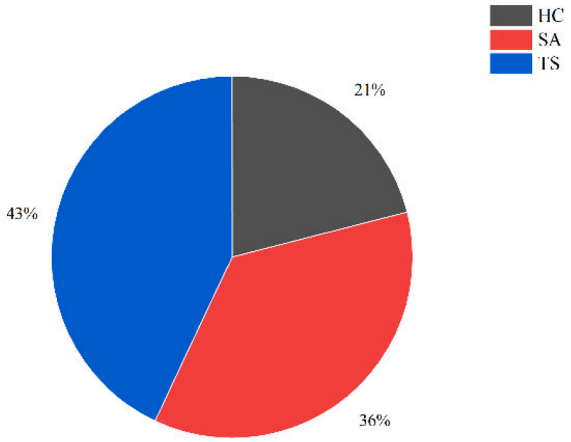

Fig. 15. The probability of each algorithm.

In the second part, we discuss the influence of the parallel architecture on the algorithm. We compare FPBPS with frequent pattern-based local search-hill climbing (FPBLS-HC), frequent pattern-based local search-tabu search (FPBLS-TS) and frequent pattern-based local search-simulated annealing (FPBLS-SA). Fig. 13 shows that FPBPS obtains a better profit rate than the other algorithms. This is due to the local search algorithms running in parallel, which greatly make full play to the local optimization ability. Meanwhile, parallel local search can obtain diverse and high-quality solutions through different search trajectories and provide better data for data mining.

In the last part, the influence of the competition procedure on the quality of final solution is discussed. We compare FPBPS with frequent pattern-based local search-no competition (FPBLS-NC). Fig. 14 shows that FPBPS obtains a better profit rate than the FPBLS-NC in all scenarios. The competition procedure meets the algorithm's requirements for self-adaptive ability by updating the probability regularly. In the process of search, this procedure uses appropriate algorithms and operators, which provides an important guarantee for the quality of final solutions. Fig. 15 shows the overall probability of each algorithm.

5. Conclusion and future work

In this paper, we propose the FPBPS algorithm to solve the AEOS scheduling problem. Firstly, different local algorithms run in parallel manner. As a result, some high-quality and diverse solutions are obtained in a short time. Then, we regularly evaluate the contribution of algorithms and operators and update the probability, which improves the self-adaptive of the FPBPS algorithm. Next, the frequent pattern mining method is applied to extract knowledge with respect to AEOS scheduling and construct new solutions. Finally, the experimental results prove that the proposed FPBPS algorithm can achieve better results than state-of-the-art comparison algorithms at different task scales.

The current work can be expanded in two aspects: 1) integrating more *meta*-heuristic algorithms and machine learning algorithms into the FPBPS algorithm; and 2) trying to solve different time/order dependent problems to verify the effectiveness of the FPBPS algorithm.

CRediT authorship contribution statement

Jian Wu: Conceptualization, Methodology, Software. **Feng Yao:** Formal analysis. **Yanjie Song:** Formal analysis. **Lei He:** Validation. **Fang Lu:** Validation. **Yonghao Du:** Validation. **Jungang Yan:** Validation. **Yuning Chen:** Validation. **Lining Xing:** Project administration. **Junwei Ou:** Formal analysis.

Declaration of Competing Interest

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

Data availability

No data was used for the research described in the article.

Acknowledgements

This study is supported by the National Natural Science Foundation of China (71701203, 71901213, 72001212, 72271240),

Natural Science Foundation of Hunan (2022JJ30671), Special Project in Major Fields of Guangdong Universities (Grant No. 2021ZDZX1019), the National Natural Science Fund for Distinguished Young Scholars of China (61525304) and the Hunan Postgraduate Research Innovation Project (CX20210031). It is also supported by the Science and Technology Innovation Team of Shaanxi Province (2023-CX-TD-07).

References

[1] He, Y., Xing, L., Chen, Y., Pedrycz, W., Wang, L., & Wu, G. (2020). A generic Markov decision process model and reinforcement learning method for scheduling agile earth observation satellites. IEEE Transactions on Systems, Man, and Cybernetics: Systems.
[2] G. Peng, G. Song, L. Xing, A. Gunawan, P. Vansteenwegen, An exact algorithm for agile earth observation satellite scheduling with time-dependent profits, Computers & Operations Research 120 (2020), 104946.
[3] Y. Du, T. Wang, B. Xin, L. Wang, Y. Chen, L. Xing, A data-driven parallel scheduling approach for multiple agile earth observation satellites, IEEE Transactions on Evolutionary Computation 24 (4) (2019) 679–693.
[4] G. Peng, R. Dewil, C. Verbeeck, A. Gunawan, L. Xing, P. Vansteenwegen, Agile earth observation satellite scheduling: An orienteering problem with time-dependent profits and travel times, Computers & Operations Research 111 (2019) 84–98.
[5] Chien, S. , Sherwood, R. , Tran, D. , Castano, R. , Cichy, B. , & Davies, A. , et al. (2003). Autonomous Science on the EO-1 Mission Abstract.
[6] H. Qiu, B. Zhao, W. Gu, R. Bo, Bi-level two-stage robust optimal scheduling for AC/DC hybrid multi-microgrids, IEEE Transactions on Smart Grid 9 (5) (2018) 5455–5466.
[7] Chien, S., Tran, D., Rabideau, G., Schaffer, S., Mandl, D., & Frye, S. (2009). Planning Operations of the Earth Observing Satellite EO-1: Representing and reasoning with spacecraft operations constraints. In Proc. 6th Int. Workshop Plan. Scheduling Space (IWPSS) (pp. 1-8).
[8] Cichy, B. , Chien, S. , Rabideau, G. , & Tran, D. . (2004). Validating the autonomous EO-1 science agent. International Workshop on Planning and Scheduling for Space. Pasadena, CA : Jet Propulsion Laboratory, National Aeronautics and Space Administration, 2004.
[9] B. Wille, M.T. Wörle, C. Lenzen, Vamos – verification of autonomous mission planning on-board a spacecraft, IFAC Proceedings Volumes 46 (19) (2013) 382–387.
[10] K.A.M. Goetz, F. Huber, M.V. Schoenermark, Vimos - autonomous image analysis on board of biros, IFAC Proceedings Volumes 46 (19) (2013) 423–428.
[11] N. Bianchessi, J.F. Cordeau, J. Desrosiers, G. Laporte, V. Raymond, A heuristic for the multi-satellite, multi-orbit and multi-user management of earth observation satellites, European Journal of Operational Research 177 (2) (2007) 750–762.
[12] X. Chu, Y. Chen, Y. Tan, A branch and bound algorithm for agile earth observation satellite scheduling, Advances in Space Research 2017 (9) (2017) 1–15.
[13] G. Wu, Q. Luo, X. Du, Y. Chen, P.N. Suganthan, X. Wang, Ensemble of Metaheuristic and Exact Algorithm Based on the Divide-and-Conquer Framework for Multisatellite Observation Scheduling, IEEE Transactions on Aerospace and Electronic Systems 58 (5) (2022) 4396–4408.
[14] X. Liu, G. Laporte, Y. Chen, R. He, An adaptive large neighborhood search metaheuristic for agile satellite scheduling with time-dependent transition time, Computers & Operations Research 86 (2017) 41–53.
[15] L. He, X. Liu, G. Laporte, Y. Chen, Y. Chen, An improved adaptive large neighborhood search algorithm for multiple agile satellites scheduling, Computers & Operations Research 100 (2018) 12–25.
[16] Peng, G., Song, G., He, Y., Yu, J., Xiang, S., Xing, L., & Vansteenwegen, P. (2020). Solving the agile earth observation satellite scheduling problem with time-dependent transition times. IEEE Transactions on Systems, Man, and Cybernetics: Systems.
[17] Han, C., Gu, Y., Wu, G., & Wang, X. (2022). Simulated Annealing-Based Heuristic for Multiple Agile Satellites Scheduling Under Cloud Coverage Uncertainty. IEEE Transactions on Systems, Man, and Cybernetics: Systems.
[18] X. Wang, Y. Gu, G. Wu, J.R. Woodward, Robust scheduling for multiple agile Earth observation satellites under cloud coverage uncertainty, Computers & Industrial Engineering 156 (2021), 107292.
[19] L. Wei, L. Xing, Q. Wan, Y. Song, Y. Chen, A multi-objective memetic approach for time-dependent agile earth observation satellite scheduling problem, Computers & Industrial Engineering 159 (2021), 107530.
[20] Y. Du, L. Xing, J. Zhang, Y. Chen, Y. He, MOEA based memetic algorithms for multi-objective satellite range scheduling problem, Swarm and Evolutionary Computation 50 (2019), 100576.
[21] J. Zhang, L. Xing, G. Peng, F. Yao, C. Chen, A large-scale multiobjective satellite data transmission scheduling algorithm based on SVM+ NSGA-II, Swarm and Evolutionary Computation 50 (2019), 100560.
[22] F. Wang, X. Wang, S. Sun, A reinforcement learning level-based particle swarm optimization algorithm for large-scale optimization, Information Sciences 602 (2022) 298–312.
[23] K. Li, T. Zhang, R. Wang, Deep reinforcement learning for multiobjective optimization, IEEE transactions on cybernetics 51 (6) (2020) 3103–3114.
[24] Lu, H., Zhang, X., & Yang, S. (2019, September). A learning-based iterative method for solving vehicle routing problems. In International conference on learning representations.
[25] Y. Wei, M. Zhao, A reinforcement learning-based approach to dynamic job-shop scheduling, Acta Automatica Sinica 31 (5) (2005) 765.
[26] W. Usaha, J.A. Barria, Reinforcement learning for resource allocation in LEO satellite networks, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 37 (3) (2007) 515–527.
[27] W.A.N.G. Haijiao, Y.A.N.G. Zhen, Z.H.O.U. Wugen, L.I. Dalin, Online scheduling of image satellites based on neural networks and deep reinforcement learning, Chinese Journal of Aeronautics 32 (4) (2019) 1011–1019.
[28] E.G. Talbi, Machine learning into metaheuristics: a survey and taxonomy, ACM Computing Surveys 54 (6) (2021) 1–32.
[29] M. Karimi-Mamaghan, et al., Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art, European Journal of Operational Research 296 (2) (2022) 393–422. “.
[30] F. Arnold, K. Sörensen, What makes a VRP solution good? The generation of problem-specific knowledge for heuristics, Computers & Operations Research 106 (2019) 280–288.
[31] Kazimipour, B., Li, X., & Qin, A. K. (2014, July). A review of population initialization techniques for evolutionary algorithms. In 2014 IEEE congress on evolutionary computation (CEC) (pp. 2585-2592). IEEE.
[32] M. Guerine, I. Rosseti, A. Plastino, Extending the hybridization of metaheuristics with data mining: Dealing with sequences, Intelligent Data Analysis 20 (5) (2016) 1133–1156.
[33] M.M. Drugan, Reinforcement learning versus evolutionary computation: A survey on hybrid algorithms, Swarm and evolutionary computation 44 (2019) 228–246.
[34] Yu, S., Aleti, A., Barca, J. C., & Song, A. (2018, June). Hyper-heuristic online learning for self-assembling swarm robots. In International Conference on Computational Science (pp. 167-180). Springer, Cham.
[35] Zhou, Y., Hao, J. K., & Duval, B. (2020). Frequent pattern-based search: a case study on the quadratic assignment problem. IEEE Transactions on Systems, Man, and Cybernetics: Systems.
[36] D. Martins, G.M. Vianna, I. Rosseti, S.L. Martins, A. Plastino, Making a state-of-the-art heuristic faster with data mining, Annals of Operations Research 263 (1) (2018) 141–162.
[37] J. Wu, B. Song, G. Zhang, J. Ou, Y. Chen, F. Yao, L. Xing, A data-driven improved genetic algorithm for agile earth observation satellite scheduling with time-dependent transition time, Computers & Industrial Engineering 174 (2022), 108823.
[38] L. He, M. de Weerdt, N. Yorke-Smith, Time/sequence-dependent scheduling: the design and evaluation of a general purpose tabu-based adaptive large neighbourhood search algorithm, Journal of Intelligent Manufacturing 31 (4) (2020) 1051–1078.